Райан Бенедетти Ронан Крэнли

Изучаем работу с jQuery

Создавай удобные HTML-формы

Пиши свои собственные фукции jQuery

Добавь интерактивности своему сайту с помощью всего нескольких строчек кода

Добавляй возможности использования анимации и AJAX на сайте

Разрабатывай веб-приложения без оглядки на ограничения браузеров

O'REILLY®

ПИТЕР®

О других книгах серии *Head First*

«В книге „*Изучаем объектно-ориентированный анализ и проектирование*“ используется новый подход к теме ООАП. Эта книга отличается от других тем, что она ориентирована на обучение. Авторы сделали материал ООАП доступным и полезным для программиста-практика».

— Айвар Джейкобсон, Ivar Jacobson Consulting

«Я только что закончил читать „*Изучаем объектно-ориентированный анализ и проектирование*“ и остался в полном восторге! Больше всего мне понравилось то, что книга ориентирована на необходимость применения ООАП для написания хороших программ».

— Кайл Браун, заслуженный инженер, IBM

«За смешными картинками и вычурными шрифтами скрывается серьезное, разумное, исключительно качественно проработанное изложение объектно-ориентированного анализа и проектирования. Во время чтения книги мне казалось, что я стою за спиной опытного проектировщика, который объясняет мне, какие аспекты важны на каждом этапе проектирования, и почему».

— Эдвард Сьоре, адьюнкт-профессор, факультет компьютерных технологий, Бостонский колледж

«В целом „*Управление разработкой ПО*“ представляет собой замечательный источник информации для всех, кто хочет привести в порядок свои навыки программирования — причем подход к изложению материала постоянно привлекает внимание читателя на многих уровнях».

— Энди Хадсон, Linux Format

«Программистам-новичкам „*Управление разработкой ПО*“ поможет сразу встать на правильный путь. Да и опытные разработчики найдут здесь для себя немало полезного».

— Томас Дафф, Duffbert's Random Musings

«Вместо изложения материала в стиле традиционных учебников „*Программируем для iPhone и iPad*“ предлагает читателю живую, увлекательную и даже приятную методику обучения программированию для iOS. Материал подобран умело и качественно: в книге рассматриваются многие ключевые технологии, включая Core Data, и даже такие важные аспекты, как проектирование интерфейса. И где еще можно прочитать, как UIWebView и UITextField беседуют у камина?»

— Шон Мерфи, проектировщик и разработчик приложений для iOS

О других книгах серии *Head First*

«Книга „*Программируем для iPhone и iPad*“ объясняет принципы разработки приложений iOS с самого начала. Основные изменения по сравнению с первым изданием относятся к iOS 4, Xcode 4 и написанием приложений для iPad. Благодаря пошаговым описаниям с визуальным стилем изложения материала эта книга становится отличным средством изучения программирования для iPhone и iPad во всех аспектах, от простейших до нетривиальных».

— Рич Розен, программист и соавтор книги Mac OS X for Unix Geeks

«Главное достоинство книги — простые, пошаговые описания. Она не пытается научить читателя всему сразу, а знакомит его с построением приложений для iOS на уровне дружественного, разговорного общения. Эта книга идеально подходит для людей, которые уже умеют программировать и хотят поскорее заняться построением приложений для iOS».

— Эрик Шеферд, владелец Syndicomm

«Книга „*Программируем для iPhone и iPad*“ написана специально для того, чтобы научить вас создавать приложения, изучать и использовать технологии iOS без предварительного опыта работы со средствами разработки на Macintosh».

— Джо Хек, основатель Seattle Xcoders

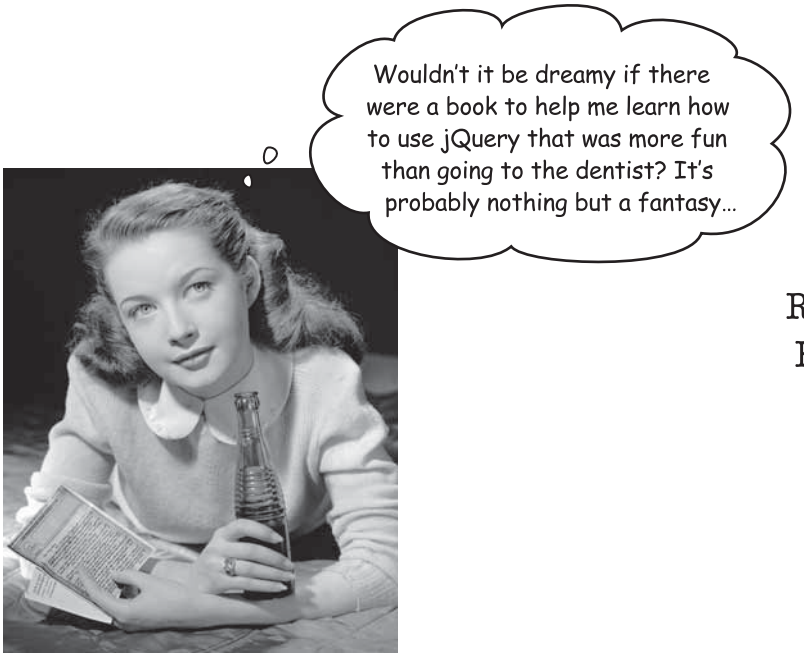
«Эта книга способна вывести из себя! Некоторым из нас пришлось изучать программирование для iOS „классическим“ способом, а теперь вдруг выясняется, что все мучения были излишними».

— Майк Моррисон, основатель Stalefish Labs

«Книга „*Программируем для iPhone и iPad*“ продолжает нарастающую тенденцию к изложению сложных технических тем на доступном уровне, но без ущерба для глубины и масштабности материала. Программирование для iOS в любом случае придется покорять, как горную вершину, но с книгой „Программируем для iPhone и iPad“ вы будете взбираться на эту „гору“ по заранее проложенным маршрутам, со страховкой и в компании опытного проводника! Я рекомендую эту книгу каждому, кто хочет быстро освоить программирование для этой непростой и интересной платформы».

— Крис Пилсор, snogboggin.com

Head First jQuery



Ryan Benedetti
Ronan Cranley

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Изучаем работу с jQuery

Райан Бенедетти
Ронан Крэнли

ПИТЕР®

Москва · Санкт-Петербург · Нижний Новгород · Воронеж
Ростов-на-Дону · Екатеринбург · Самара · Новосибирск
Киев · Харьков · Минск
2012

ББК 32.973.2-018.1
УДК 004.434
Б41

Бенедетти Р., Крэнли Р.
Б41 Изучаем работу с jQuery. — СПб.: Питер, 2012. — 528 с.: ил.

ISBN 978-5-459-00896-8

Хотите добавить интерактивности своему интернет-сайту? Узнайте, как jQuery позволит вам создать целый набор скриптов, используя всего несколько строчек кода! С помощью этого издания вы максимально быстро научитесь работать с jQuery — этой удивительной библиотекой JavaScript, использование которой сегодня стало необходимостью для разработки современных веб-сайтов и RIA-приложений. jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, а также предоставляет богатые возможности по взаимодействию с AJAX.

Особенностью данного издания является уникальный способ подачи материала, выделяющий серию «Head First» издательства O'Reilly в ряду множества скучных книг, посвященных программированию.

ББК 32.973.2-018.1
УДК 004.434

ISBN 978-1449393212 англ.

ISBN 978-5-459-00896-8

Мы посвящаем эту книгу магистрам JavaScript: Джону Резигу (создателю и ведущему разработчику библиотеки jQuery), Дугласу Крокфорду, Дэвиду Фленагану и Брэндону Эйку.

Моим трем феям: Джози, Вин и Шонне.

– Райан

Кейтлин и Боно: спасибо вам за все!

– Ронан

Райан

Райан Бенедетти — обладатель степени магистра искусств в области писательского мастерства (университет Монтаны); работает веб-разработчиком и специалистом по мультимедиа в университете Портленда. В своей работе использует jQuery, Flash, ActionScript, Adobe Creative Suite, Liferay Portal, Apache Jakarta Velocity и Drupal.

В течение семи лет Райан руководил факультетом информационных и компьютерных технологий в колледже Салиш-Кутени. До этого он работал редактором и специалистом по информационным системам в программе исследования рек, ручьев и сильно увлажненных земель в школе лесоводства при университете Монтаны.

Стихотворения Райана опубликованы в поэтических сборниках. В свободное время занимается рисованием, играет на губной гармошке и практикует сидячую медитацию. Свои лучшие моменты жизни Райан проводит с дочерью, сыном и своей любимой Шонной в Портленде, штат Орегон. Также любит выбираться на природу со своими четвероногими друзьями: Роки, Манчем, Фестером и Тазом.

Ронан

Ронан Крэнли работал в университете Портленда с момента своего переезда из Дублина, Ирландия, в 2006 году — с должности веб-разработчика до старшего веб-разработчика/инженера-системотехника и заместителя директора по веб-системам и администрированию.

Ронан получил степень бакалавра в области компьютерных технологий в технологическом институте Дублина, который он закончил с отличием в 2003 году. Во время учебы, а также на своей предыдущей должности в ESB International (Дублин) и текущей в университете Портленда Ронан работал над разнообразными проектами на языках PHP, VB.NET, C# и Java. Среди этих проектов была клиентская система GIS, система управления контентом, система календарного планирования и гибридное приложение jQuery/Google Maps.

Помимо проектирования и построения веб-приложений, Ронан также занимается администрированием баз данных SQL Server. Свободное время Ронан проводит на футбольном поле, на площадке для гольфа, а также со своей женой Кейтлин и английским бульдогом Боно, наслаждаясь всей полнотой жизни на северо-западном Тихоокеанском побережье.

Содержание (сводка)

Содержание (настоящее)

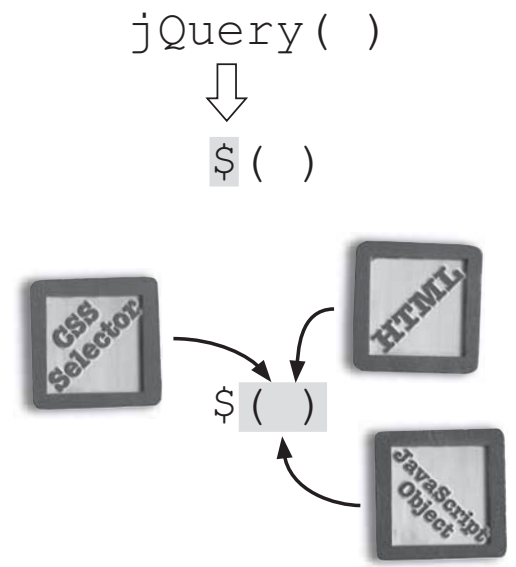
Введение

Ваш мозг думает о jQuery. Вы сидите за книгой и пытаетесь что-нибудь выучить, но ваш мозг считает, что вся эта писанина не нужна. Ваш мозг говорит: «Выгляни в окно! На свете есть более важные вещи, например сноуборд». Как заставить мозг думать, что ваша жизнь действительно зависит от jQuery?

1 Знакомство с jQuery

Живые веб-страницы

Вы хотите расширить возможности своих веб-страниц. Вы уже знаете HTML и CSS и хотите включить в свой арсенал сценарное программирование, но вам совершенно не хочется проводить свою жизнь за написанием сотен строк кода. Нужна библиотека сценариев, которая позволяла бы изменять веб-страницы «на ходу». И если на то пошло, как насчет поддержки AJAX и PHP? И чтобы в трех строках кода можно было сделать то, для чего в большинстве клиентских языков потребуется 15? Пустые мечты, скажете вы... А вот и нет! На помощь приходит jQuery.



2 Селекторы и методы

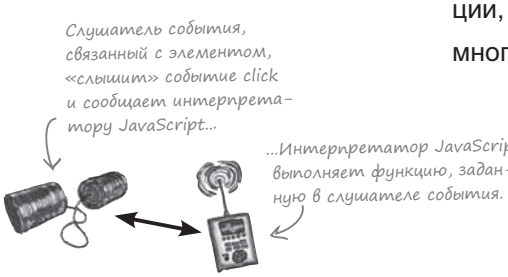
Хватай и действуй

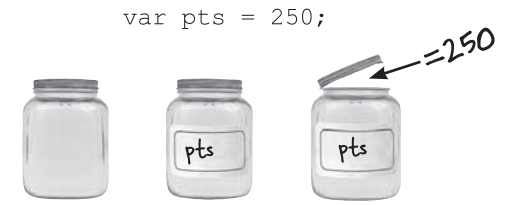
jQuery помогает выбирать элементы веб-страниц и выполнять с ними всевозможные операции. В этой главе более подробно рассмотрены селекторы и методы jQuery. Селекторы jQuery выбирают элементы страницы, а методы выполняют операции с этими элементами. Библиотека jQuery, словно сборник магических заклинаний, позволяет изменять окружающую реальность. Вы можете заставить изображение исчезнуть или появиться из ниоткуда или же выбрать фрагмент текста и анимировать изменение его размера шрифта... Но довольно разговоров — хватайте элементы веб-страниц и действуйте!

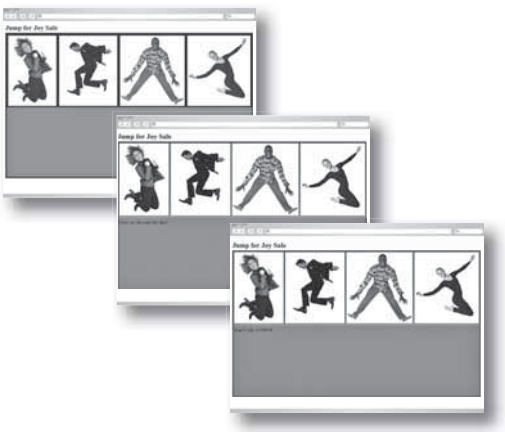
3 События и функции jQuery

Страница в центре событий

jQuery позволяет легко включить в любую веб-страницу поддержку действий и интерактивности. В этой главе вы узнаете, как заставить вашу страницу реагировать на действия, выполняемые пользователем. Возможность выполнения кода в ответ на действия пользователя поднимает сайт на совершенно новый уровень. Также в этой главе рассказано, как создавать функции, чтобы однократно написанный код можно было использовать много раз.







Операции со структурой страниц в jQuery

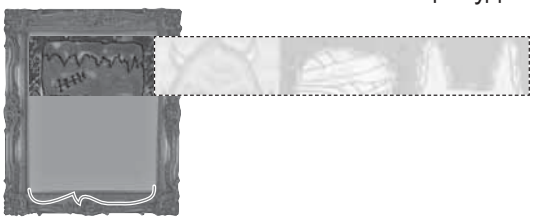
4 Изменение DOM

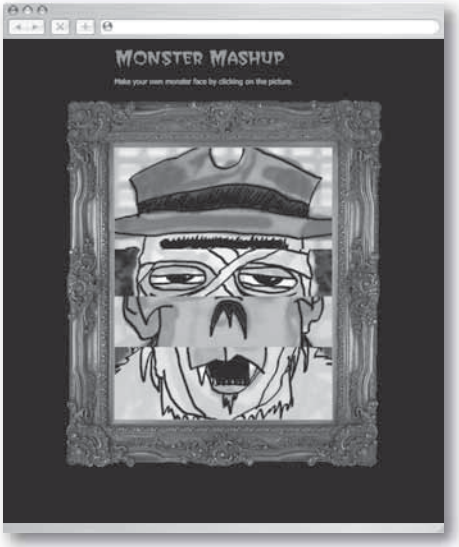
Завершение загрузки страницы еще не означает, что ее структура останется неизменной. В главе 1 было показано, как в процессе загрузки страницы строится модель DOM, определяющая ее структуру. В этой главе вы научитесь перемещаться по дереву DOM, работать с иерархией элементов и отношениями «родитель/потомок» для изменения структуры страницы «на ходу» средствами jQuery.

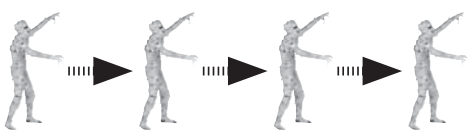
5 jQuery эффекты и анимация

Плавно и изящно

Реализация всяких интересных возможностей — дело замечательное, но если ваша страница не будет хорошо смотреться, люди не станут приходить на сайт. И здесь на первый план выходят визуальные эффекты и анимация jQuery. Вы научитесь организовывать переходы, скрывать и отображать нужные части элементов, изменять размеры элементов на странице — и все это на глазах у ваших посетителей! Вы научитесь планировать выполнение анимаций, чтобы они происходили с различными интервалами, отчего ваша страница будет выглядеть исключительно динамично.



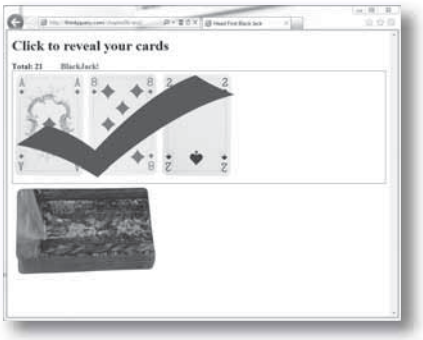


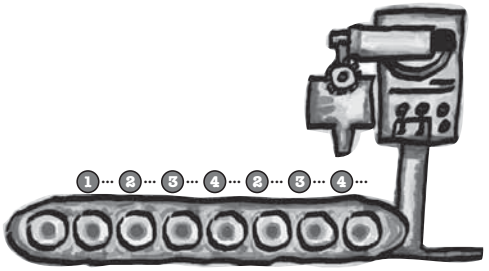


6 jQuery и JavaScript

~~Люк~~ jQuery, я твой отец!

Силы jQuery небезграничны. Хотя эта библиотека написана на JavaScript, к сожалению, она позволяет сделать не все, на что способен язык-родитель. В этой главе мы рассмотрим некоторые возможности JavaScript, необходимые для создания современных сайтов, и их применение в jQuery для создания пользовательских списков и объектов, а также средства перебора списков и объектов, которые существенно упростят вашу работу.

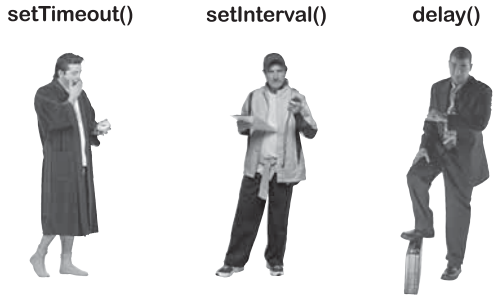




7 Пользовательские функции для пользовательских эффектов

Что будем делать?

От объединения пользовательских эффектов jQuery с функциями JavaScript ваш код (и ваши веб-приложения) становится более эффективным и более мощным. В этой главе мы займемся совершенствованием эффектов jQuery посредством обработки событий браузера, использования временных функций и улучшения общей структуры и возможностей повторного использования пользовательских функций JavaScript.

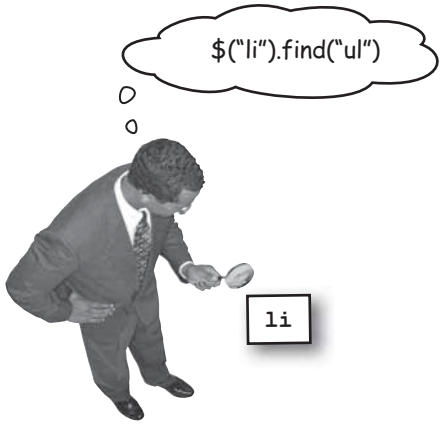


8

jQuery и Ajax

Пожалуйста, передайте данные

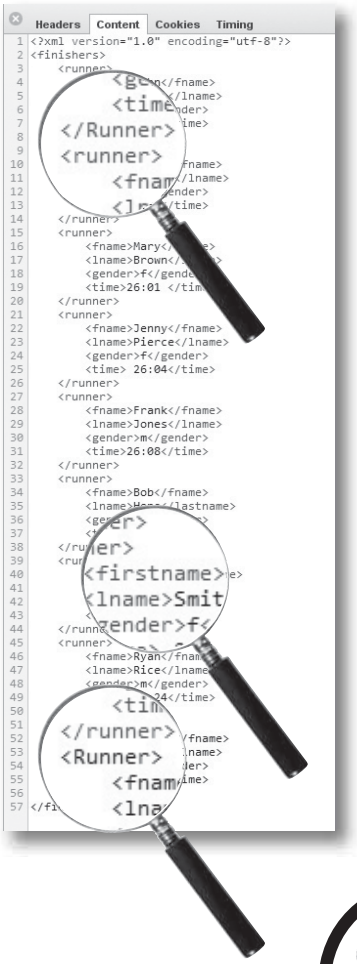
Использовать jQuery для всяких фокусов с CSS и DOM довольно весело, но при программировании веб-приложений необходимо получать данные с сервера и отображать их на странице. Возможно, вам даже захочется обновлять небольшие фрагменты страницы без полной перезагрузки страницы. Технология Ajax в сочетании с jQuery и JavaScript позволяет решить эту задачу. В этой главе вы узнаете, как в jQuery реализуются обращения Ajax к серверу и что можно сделать с полученной информацией.



9 Данные JSON

Клиент встречается с сервером

Возможность чтения данных из файлов XML безусловно полезна, но иногда этого оказывается недостаточно. Другой, более эффективный формат передачи данных JSON (JavaScript Object Notation) упрощает получение данных со стороны сервера. Кроме того, данные JSON проще генерируются и читаются, чем данные XML. При помощи jQuery, PHP и SQL можно создать базу данных для хранения информации, которая позднее читается с использованием JSON и отображается на экране средствами jQuery. Вот она, истинная мощь веб-приложений!



10 jQuery UI

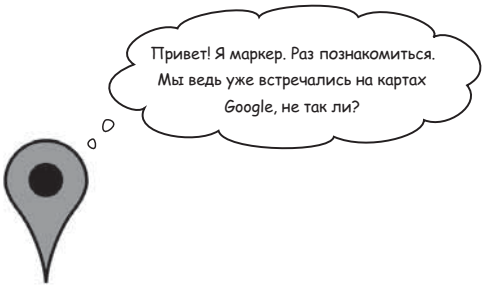
Переработка форм

Пользователи и их данные — жизнь и смерть веб-приложений. Ввод данных пользователем — серьезная задача, которая может отнять много времени у веб-разработчика. Вы уже видели, как jQuery упрощает построение веб-приложений, использующих Ajax, PHP и MySQL. Теперь давайте посмотрим, как jQuery упрощает построение пользовательского интерфейса форм для ввода данных пользователем. Заодно вы узнаете много полезного о jQuery UI — официальной библиотеке пользовательского интерфейса для jQuery.

11 jQuery и APIs

Объекты, сплошные объекты

Даже самый талантливый разработчик не сможет сделать всю работу в *одиночку*... Мы уже видели, как включать расширения jQuery (такие как jQuery UI или вкладки) для повышения уровня функциональности приложения. Чтобы поднять наше приложение на следующий уровень — использовать Интернет и информацию из крупных информационных систем типа Google, Twitter или Yahoo! — понадобится... нечто большее. Эти компании предоставляют вам программные интерфейсы (API, Application Programming Interface). В этой главе мы рассмотрим основы работы с API, а также используем очень распространенный сервис Google Maps API.



Остатки

Десять важных вещей (которые мы не рассмотрели)

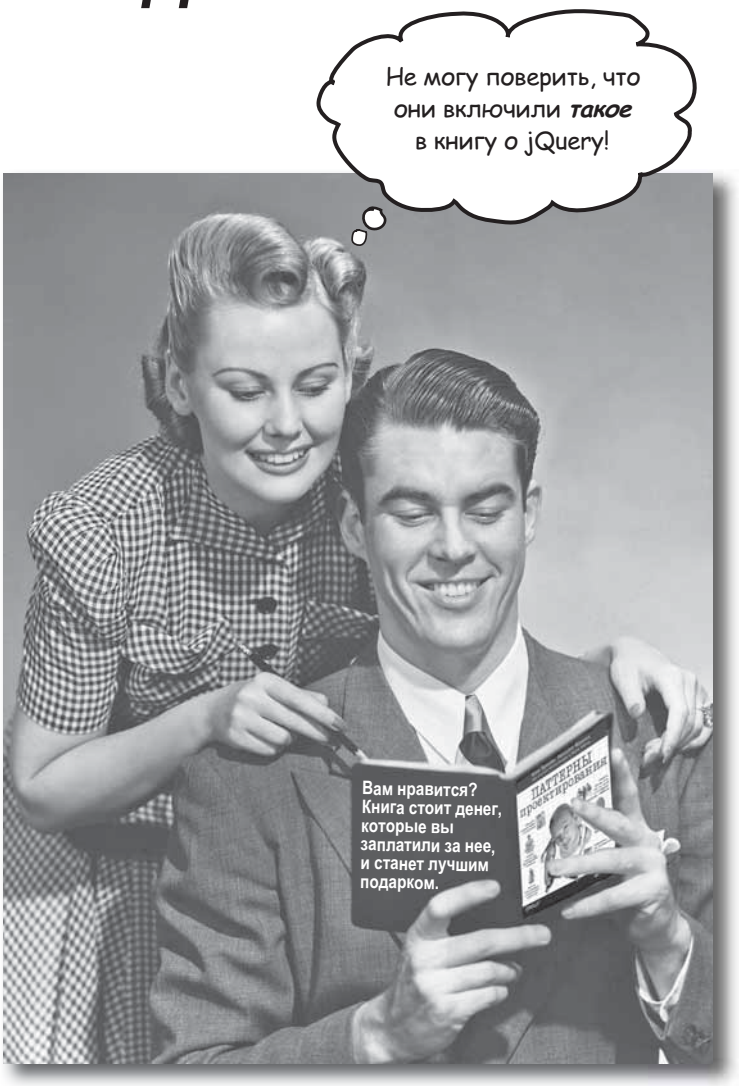
Даже после всего сказанного многое осталось «за кадром». Существует масса других полезных возможностей jQuery и JavaScript, которые нам не удалось вместить в книгу. Было бы неправильно даже не упомянуть о них. Мы хотим, чтобы вы были готовы к любому аспекту jQuery, с которым вы можете столкнуться во время самостоятельных исследований.

Настройка среды разработки

Готовимся к великим свершениям

Вам понадобится среда, в которой вы сможете потренировать свои навыки PHP, но так, чтобы ваши данные не стали уязвимыми для внешних атак. Разработку приложений PHP всегда желательно начинать с безопасной среды и только потом открывать доступ для внешнего мира. В этом приложении приведены инструкции по установке веб-сервера, MySQL и PHP. После их выполнения в вашем распоряжении появится безопасная среда для работы и экспериментов.

Введение



В этом разделе мы ответим на насущный вопрос: «Так почему они включили ТАКОЕ в книгу о jQuery?»

Для кого написана эта книга?

Если вы ответите «да» на все следующие вопросы:

1. У вас уже имеется опыт веб-разработки или веб-дизайна?

2. Вы хотите **изучить, запомнить, понять и научиться применять важнейшие** концепции jQuery и JavaScript, чтобы сделать ваши страницы более интерактивными и интересными?

3. Вы предпочитаете **оживленную беседу сухим, скучным академическим лекциям**?

Безусловно, навыки сценарного программирования тоже пригодятся. Опыт использования JavaScript полезен, но необязателен.

...то эта книга для вас.

Кому эта книга не подойдет?

Если вы ответите «да» на любой из следующих вопросов:

1. Вы **абсолютно не разбираетесь** в веб-программировании?

2. Вы уже занимаетесь разработкой веб-приложений, и ищете ***справочник*** по jQuery?

3. Вы **боитесь попробовать что-нибудь новое**? Скорее пойдете к зубному врачу, чем наденете полосатое с клетчатым? Считаете, что техническая книга со снежным человеком серьезной быть не может?

Обратитесь к книге «Изучаем HTML, XHTML и CSS» (Питер, 2012) — в ней приведен отличный вводный курс веб-программирования. А потом возвращайтесь к нам!

...эта книга не для вас.

[Заметка от отдела продаж: вообще-то эта книга для любого, у кого есть деньги.]

Мы знаем, о чем вы думаете

«Разве серьезные книги по программированию такие?»

«И почему здесь столько рисунков?»

«Можно ли так чему-нибудь научиться?»

И мы знаем, о чем думает ваш мозг

Мозг жаждет новых впечатлений. Он постоянно ищет, анализирует, *ожидает* чего-то необычного. Он так устроен, и это помогает нам выжить.

Как наш мозг поступает со всеми обычными, повседневными вещами? Он всеми силами пытается оградиться от них, чтобы они не мешали его *настоящей* работе — сохранению того, что действительно важно. Мозг не считает нужным сохранять скучную информацию. Она не проходит фильтр, отсекающий «очевидно несущественное».

Но как же мозг *узнает*, что важно? Представьте, что вы выехали на прогулку и вдруг прямо перед вами появляется тигр. Что происходит в вашей голове и теле?

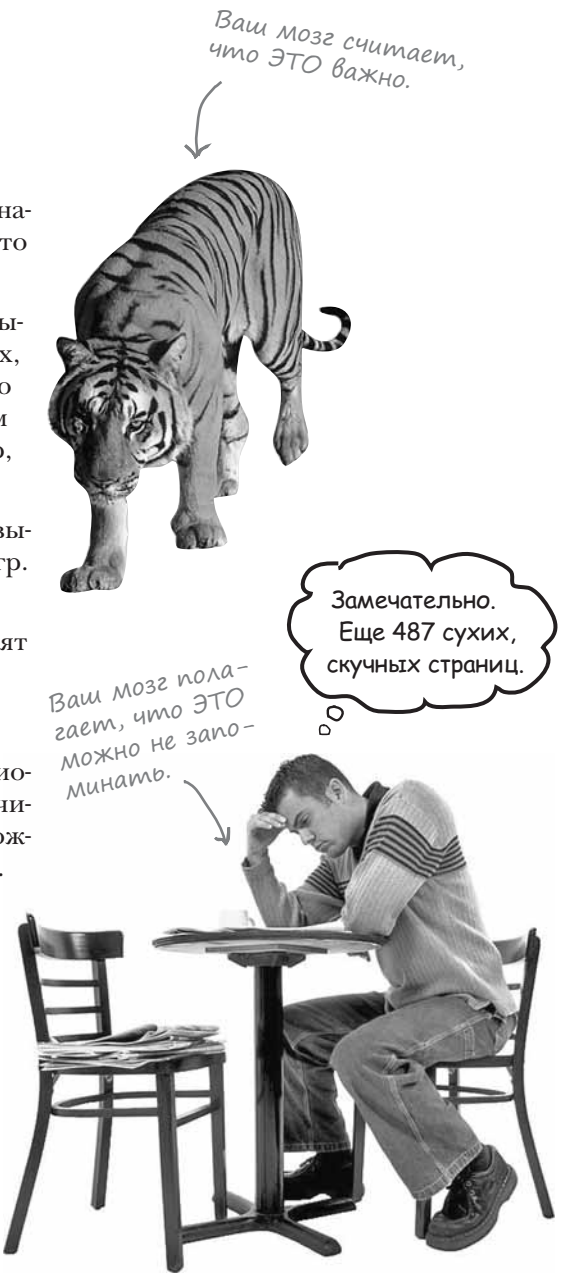
Активизируются нейроны. Вспыхивают эмоции. Происходят химические реакции. И тогда ваш мозг понимает...

Конечно, это важно! Не забывать!

А теперь представьте, что вы находитесь дома или в библиотеке, в теплом, уютном месте, где тигры не водятся. Вы учитесь — готовитесь к экзамену. Или пытаетесь освоить сложную техническую тему, на которую вам выделили неделю... максимум десять дней.

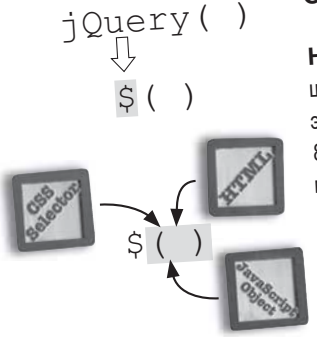
И тут возникает проблема: ваш мозг пытается оказать вам услугу. Он старается сделать так, чтобы на эту *очевидно* несущественную информацию не тратились драгоценные ресурсы. Их лучше потратить на что-нибудь важное. На тигров, например. Или на то, что к огню лучше не прикасаться. Или что на лыжах не стоит кататься в футболке и шортах.

Нет простого способа сказать своему мозгу: «Послушай, мозг, я тебе, конечно, благодарен, но какой бы скучной ни была эта книга и пусть мой датчик эмоций сейчас на нуле, я *хочу* запомнить то, что здесь написано».



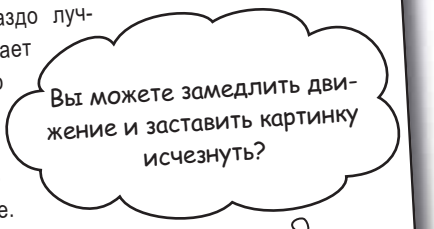
Эта книга для тех, кто хочет учиться.

Как мы что-то узнаем? Сначала нужно это «что-то» *понять*, а потом не забыть. Затолкать в голову побольше фактов недостаточно. Согласно новейшим исследованиям в области когнитивистики, нейробиологии и психологии обучения, для *усвоения материала* требуется что-то большее, чем простой текст на странице. Мы знаем, как заставить ваш мозг работать.



Основные принципы серии Head First:

Наглядность. Графика запоминается гораздо лучше, чем обычный текст, и значительно повышает эффективность восприятия информации (до 89 % по данным исследований). Кроме того, материал становится более понятным. Текст размещается на рисунках, к которым он относится, а не под ними или на соседней странице.



Разговорный стиль изложения. Недавние исследования показали, что при разговорном стиле изложения материала (вместо формальных лекций) улучшение результатов на итоговом тестировании достигает 40 %. Рассказывайте историю, вместо того чтобы читать лекцию. Не относитесь к себе слишком серьезно. Что привлечет ваше внимание: занимательная беседа за столом или лекция?

Активное участие читателя. Пока вы не начнете напрягать извилины, в вашей голове ничего не произойдет. Читатель должен быть заинтересован в результате; он должен решать задачи, формулировать выводы и овладевать новыми знаниями. А для этого необходимы упражнения и каверзные вопросы, в решении которых задействованы оба полушария мозга и разные чувства.

Привлечение (и сохранение) внимания читателя. Ситуация, знакомая каждому: «Я очень хочу изучить это, но засыпаю на первой странице». Мозг обращает внимание на интересное, странное, притягательное, неожиданное. Изучение сложной технической темы не обязано быть скучным. Интересное узнается намного быстрее.

Обращение к эмоциям. Известно, что наша способность запоминать в значительной мере зависит от эмоционального сопереживания. Мы запоминаем то, что нам небезразлично. Мы запоминаем, когда что-то чувствуем. Нет, сентименты здесь ни при чем: речь идет о таких эмоциях, как удивление, любопытство, интерес, и чувстве «Да я крут!» при решении задачи, которую окружающие считают сложной, — или когда вы понимаете, что разбираетесь в теме лучше, чем всезнайка-Боб из технического отдела.

Метапознание: наука о мышлении

Если вы действительно хотите быстрее и глубже усваивать новые знания — задумайтесь над тем, как вы задумываетесь. Учитесь учиться.

Мало кто из нас изучает теорию метапознания во время учебы. Нам *положено* учиться, но нас редко этому *учат*.

Но раз вы читаете эту книгу, то, вероятно, вы хотите освоить jQuery, и по возможности быстрее. Вы хотите *запомнить* прочитанное, а для этого абсолютно необходимо сначала *понять* прочитанное.

Чтобы извлечь максимум пользы из учебного процесса, нужно заставить ваш мозг воспринимать новый материал как Нечто Важное. Критичное для вашего существования. Такое же важное, как тигр. Иначе вам предстоит бесконечная борьба с вашим мозгом, который всеми силами уклоняется от запоминания новой информации.

Как же УБЕДИТЬ мозг, что программирование для jQuery не менее важно, чем тигр?

Есть способ медленный и скучный, а есть быстрый и эффективный. Первый основан на тупом повторении. Всем известно, что даже самую скучную информацию *можно* запомнить, если повторять ее снова и снова. При достаточном количестве повторений ваш мозг прикидывает: «*Вроде бы* несущественно, но раз одно и то же повторяется *столько раз*... Ладно, уговорил».

Быстрый способ основан на **повышении активности мозга** и особенно сочетании разных ее *видов*. Доказано, что все факторы, перечисленные на предыдущей странице, помогают вашему мозгу работать на вас. Например, исследования показали, что размещение слов *внутри* рисунков (а не в подписях, в основном тексте и т. д.) заставляет мозг анализировать связи между текстом и графикой, а это приводит к активизации большего количества нейронов. Больше нейронов — выше вероятность того, что информация будет сочтена важной и достойной запоминания.

Разговорный стиль тоже важен: обычно люди проявляют больше внимания, когда они участвуют в разговоре, так как им приходится следить за ходом беседы и высказывать свое мнение. Причем мозг совершенно не интересует, что вы «разговариваете» с книгой! С другой стороны, если текст сух и формален, то мозг чувствует то же, что чувствуете вы на скучной лекции в роли пассивного участника. Его клонит в сон.

Но рисунки и разговорный стиль — это только начало.

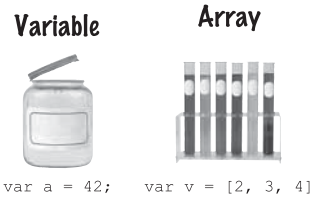
Вот что сделали МЫ:

Мы использовали **рисунки**, потому что мозг лучше приспособлен для восприятия графики, чем текста. С точки зрения мозга рисунок действительно стоит тысячи слов. А когда текст комбинируется с графикой, мы внедряем текст прямо в рисунки, потому что мозг при этом работает эффективнее.

Мы используем **избыточность**: повторяем одно и то же несколько раз, применяя разные средства передачи информации, обращаемся к разным чувствам — и все для повышения вероятности того, что материал будет закодирован в нескольких областях вашего мозга.

Мы используем концепции и рисунки несколько **неожиданным** образом, потому что мозг лучше воспринимает новую информацию. Кроме того, рисунки и идеи обычно имеют ***эмоциональное*** *содержание*, потому что мозг обращает внимание на биохимию эмоций. То, что заставляет нас *чувствовать*, лучше запоминается — будь то ***шутка***, ***удивление*** или ***интерес***.

Мы используем ***разговорный стиль***, потому что мозг лучше воспринимает информацию, когда вы участвуете в разговоре, а не пассивно слушаете лекцию. Это происходит и при *чтении*.



В книгу включены многочисленные упражнения, потому что мозг лучше запоминает, когда вы что-то делаете. Мы постарались сделать их непростыми, но интересными — то, что предпочитает большинство читателей.

Мы совместили ***несколько стилей обучения***, потому что одни читатели предпочитают пошаговые описания, другие стремятся сначала представить «общую картину», а третьим хватает фрагмента кода. Независимо от ваших личных предпочтений полезно видеть несколько вариантов представления одного материала.

Мы постарались задействовать ***оба полушария вашего мозга***; это повышает вероятность усвоения материала. Пока одна сторона мозга работает, другая часто имеет возможность отдохнуть; это повышает эффективность обучения в течение продолжительного времени.

А еще в книгу включены ***истории*** и упражнения, отражающие другие точки зрения. Мозг глубже усваивает информацию, когда ему приходится оценивать и выносить суждения.

В книге часто встречаются ***вопросы***, на которые не всегда можно дать простой ответ, потому что мозг быстрее учится и запоминает, когда ему приходится что-то делать. Невозможно накачать *мышцы*, наблюдая за тем, как занимаются *другие*. Однако мы позаботились о том, чтобы усилия читателей были приложены в *верном* направлении. Вам не придется ломать голову над невразумительными примерами или разбираться в сложном, перенасыщенном техническим жаргоном или слишком лаконичном тексте.

В историях, примерах, на картинках используются ***люди*** — потому что вы тоже ***человек***. И ваш мозг обращает на людей больше внимания, чем на неодушевленные ***предметы***.

Вырежьте и прикрепите на холодильник.

Что можете сделать ВЫ, чтобы заставить свой мозг повиноваться

Мы свое дело сделали. Остальное за вами. Эти советы станут отправной точкой; прислушайтесь к своему мозгу и определите, что вам подходит, а что не подходит. Пробуйте новое.

1 Не торопитесь. Чем больше вы поймете, тем меньше придется запоминать.

Просто читать недостаточно. Когда книга задает вам вопрос, не переходите к ответу. Представьте, что кто-то *действительно* задает вам вопрос. Чем глубже ваш мозг будет мыслить, тем скорее вы поймете и запомните материал.

2 Выполняйте упражнения, делайте заметки.

Мы включили упражнения в книгу, но выполнять их за вас не собираемся. И не *разглядывайте* упражнения. Берите карандаш и пишите. Физические действия *во время* учения повышают его эффективность.

3 Читайте врезки.

Это значит: читайте все. ***Врезки – часть основного материала!*** Не пропускайте их.

4 Не читайте другие книги после этой перед сном.

Часть обучения (особенно перенос информации в долгосрочную память) происходит после того, как вы откладываете книгу. Ваш мозг не сразу усваивает информацию. Если во время обработки поступит новая информация, часть того, что вы узнали ранее, может быть потеряна.

5 Пейте воду. И побольше.

Мозг лучше всего работает в условиях высокой влажности. Дегидратация (которая может наступить еще до того, как вы почувствуете жажду) снижает когнитивные функции.

6 Говорите вслух.

Речь активизирует другие участки мозга. Если вы пытаетесь что-то понять или получше запомнить, произнесите вслух. А еще лучше, попробуйте объяснить кому-нибудь другому. Вы будете быстрее усваивать материал и, возможно, откроете для себя что-то новое.

7 Прислушивайтесь к своему мозгу.

Следите за тем, когда ваш мозг начинает уставать. Если вы начинаете поверхностно воспринимать материал или забываете только что прочитанное — пора сделать перерыв. С определенного момента попытки «затолкать» в мозг дополнительную информацию не только не ускоряют обучение, а скорее идут во вред ему.

8 Чувствуйте!

Ваш мозг должен знать, что материал книги действительно *важен*. Переживайте за героев наших историй. Придумывайте собственные подписи к фотографиям. Поморщиться над неудачной шуткой все равно лучше, чем не почувствовать ничего.

9 Творите!

Попробуйте применить новые знания в своей повседневной работе. Просто сделайте хоть что-нибудь, чтобы приобрести практический опыт за рамками упражнений. Все, что для этого нужно — это карандаш и подходящая задача... задача, в которой изучаемые методы и инструменты могут принести пользу.

Примите к сведению

Это учебник, а не справочник. Мы намеренно убрали из книги все, что могло бы помешать изучению материала, над которым вы работаете. И при первом чтении книги начинать следует с самого начала, потому что книга предполагает наличие у читателя определенных знаний и опыта.

Предполагается, что вы уже знаете HTML и CSS.

Если вы еще не знаете HTML и CSS, то для начала найдите книгу *Head First HTML with CSS & XHTML*. В этой книге мы напомним вам, как работают селекторы CSS, но не ждите, что здесь будет рассказано все, что необходимо знать о CSS.

От вас не требуется знание JavaScript.

Знаем, знаем — кто-то с нами не согласится, но мы считаем, что jQuery можно изучать без предварительного изучения JavaScript. Конечно, для написания кода jQuery необходимо знать некоторые концепции JavaScript, но мы представим их одновременно с описанием кода jQuery. Мы глубоко и искренне верим в девиз jQuery: Меньше Кода, Больше Дела.

Мы рекомендуем использовать разные браузеры.

Страницы желательно тестировать по крайней мере в трех современных браузерах. Так вы узнаете, чем отличаются разные браузеры, и научитесь создавать страницы, нормально работающие в разных браузерах.

Эта книга не о браузерных средствах разработчика...

...но предполагается, что вы умеете пользоваться ими. Мы настоятельно рекомендуем использовать браузер Google Chrome, который можно загрузить по адресу *http://www.google.com/chrome*. Информация о браузерах и их средствах разработчика представлена на следующих сайтах:

Google Chrome	*http://code.google.com/chrome/devtools/docs/overview.html*
Firefox Firebug	*http://getfirebug.com/wiki/index.php/FAQ*
Safari	*http://www.apple.com/safari/features.html#developer*
Internet Explorer 8	*http://msdn.microsoft.com/en-us/library/dd565628(v=vs.85).aspx*
Internet Explorer 9	*http://msdn.microsoft.com/en-us/ie/aa740478*
Opera Dragonfly	*http://www.opera.com/dragonfly/*

Мы надеемся, что вы не ограничитесь чтением книги.

Лучшее, что можно сделать при изучении чего-то нового — присоединиться к сообществу изучающих. Мы считаем, что сообщество jQuery — одно из лучших, самых активных сообществ в мире современных технологий. Дополнительная информация приводится на сайте *http://www.jquery.com*.

Упражнения ОБЯЗАТЕЛЬНЫ.

Упражнения являются частью основного материала книги. Одни упражнения способствуют запоминанию материала, другие помогают лучше понять его, третьи ориентированы на его практическое применение. Не пропускайте упражнения.

Повторение применяется намеренно.

У книг этой серии есть одна принципиальная особенность: мы хотим, чтобы вы действительно хорошо усвоили материал. И чтобы вы запомнили все, что узнали. Большинство справочников не ставит своей целью успешное запоминание, но это не справочник, а учебник, поэтому некоторые концепции излагаются в книге по нескольку раз.

Упражнения «Мозговой штурм» не имеют ответов.

В некоторых из них правильного ответа вообще нет, в других вы должны сами решить, насколько правильны ваши ответы (это является частью процесса обучения). В некоторых упражнениях «Мозговой штурм» приводятся подсказки, которые помогут вам найти нужное направление.

Требования к программному обеспечению

Для написания кода jQuery вам понадобится текстовый редактор, браузер, веб-сервер (он может работать локально на вашем персональном настольном компьютере) и библиотека jQuery.

В системе Windows мы рекомендуем использовать редакторы PSPad, TextPad и EditPlus (хотя при желании можно работать и в Блокноте). Для Mac можно воспользоваться редактором TextWrangler. В системе Linux имеется много встроенных текстовых редакторов; вероятно, пользователям этой системы не нужно рассказывать про них.

Для разработки веб-приложений понадобится веб-сервер. Для последних глав книги (9, 10 и 11) вам стоит обратиться к приложению с описанием установки PHP, MySQL и веб-сервера (Apache или IIS) и выполнить приведенные там инструкции. Сделайте это прямо сейчас. Нет, серьезно — откройте приложение, выполните инструкции и возвращайтесь, когда все необходимое будет установлено.

Вам также понадобится браузер и средства разработчика (см. предыдущую страницу). Научитесь пользоваться консолью JavaScript в Google Chrome Dev Tools — вы не пожалеете о потраченном времени. Считайте, что это домашнее задание, которое вы должны выполнить самостоятельно.

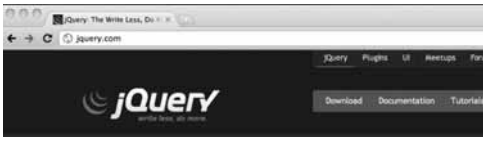
Наконец, вам понадобится сама библиотека jQuery; о том, где ее взять, рассказано на следующей странице.

Загрузка jQuery

Пора браться за дело. Откройте сайт jQuery и загрузите копию библиотеки, которую вы будете использовать в книге.

Шаг 1:

Запустите браузер и введите в нем адрес *http://www.jquery.com*.



Шаг 2:

Найдите раздел «Grab the Latest Version» и установите флажок рядом со строкой «Production».

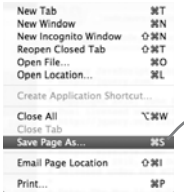
Шаг 3:

Щелкните на кнопке «Download jQuery».

Шаг 4:

Открывается следующая страница, которая выглядит примерно так.

Сохраните ее в папке с именем *scripts*.



jquery-1.6.2.min.js

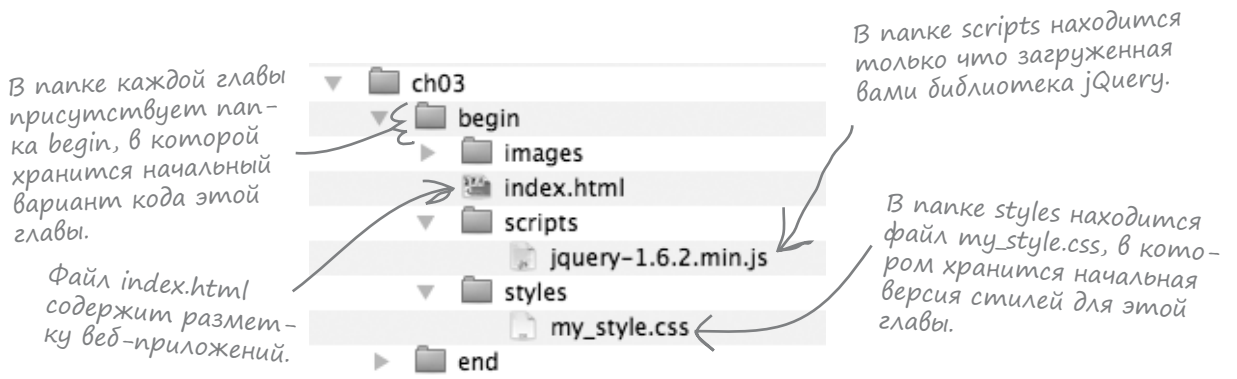
scripts

Чем рабочая (Production) версия отличается от версии для разработчика (Development)?

Рабочая версия оптимизирована для максимальной скорости выполнения на веб-сервере. Версия для разработчика предназначена для изучения внутреннего строения библиотеки jQuery. Если вы любите «покопаться во внутренностях» библиотек, загрузите обе версии.

Структура папок

После загрузки и распаковки кода с сайта Head First Labs (*http://www.headfirstlabs.com/books/hfjquery*) вы увидите, что для каждой главы книги в структуре кода создана отдельная папка. Для примера возьмем папку *ch03*:



Папка *end* в папке каждой главы содержит итоговую версию кода этой главы. Используйте папку *end* только для того, чтобы сверить ее с вашей версией кода.

Вы можете свободно использовать библиотеку jQuery в любых своих проектах. Для удобства мы включили jQuery в папки с кодом книги, однако вы должны знать, где загрузить jQuery для ваших будущих проектов, а также при выходе новой версии — библиотека jQuery регулярно обновляется.

Научные редакторы

Линдси Скурас

Билл Мителски

Пол Барри

Джим Доран

Джим Доран работает программистом в университете Джона Хопкинса в Балтиморе, штат Мэриленд. Он преподает JavaScript в муниципальном колледже округа Балтимор, а также выступает с лекциями о jQuery на веб-конференциях. В остальное время Джим ведет блог по адресу *http://jimdoran.net* и катается на роликовых коньках.

Билл Мителски был научным редактором нескольких книг из серии Head First. В настоящее время он работает программистом в крупном национальном медицинском центре в окрестностях Чикаго, где занимается биостатистическими исследованиями. Когда он не занят сбором и обработкой данных, его можно найти на поле для гольфа, где он гоняет маленький белый мячик.

Линдси Скурас — юрист из Вашингтона. Она самостоятельно научилась программировать в свободное время по книгам серии Head First. Ее другие увлечения — чтение, рукоделие, посещение музеев и проведение свободного времени с мужем и собаками.

Пол Барри читает лекции по компьютерным дисциплинам в технологическом институте Карлоу (Ирландия). Пол пишет для журнала *Linux Journal*, а также опубликовал несколько технических книг. Он является автором *Head First Python* и соавтором *Head First Programming*. В свободное время Пол консультирует малые и средние предприятия, а также начинающие фирмы по ведению программных проектов.

Благодарности


Кортни Нэш

Нашему редактору:

Спасибо (и наши поздравления!) Кортни Нэш, которая заставила нас выложиться «на полную» в работе над книгой. Она справилась с огромным потоком сообщений, вопросов, жалоб и наших капризов. Она постоянно была рядом с нами и доверяла нам — а мы, естественно, доверяли ей.

Группе O'Reilly:

Спасибо **Лу Барр** за быструю, превосходную и просто волшебную работу. Благодаря ей эта книга приняла свою окончательную форму и стала такой красивой.

Спасибо **Лори Петрики** за то, что она дала «зеленый свет» нашему проекту. Райан сохранил теплые воспоминания о программе обучения авторов HF в Бостоне и никогда не забудет классной, семейной атмосферы, которую здесь создала Лори.

Спасибо **Карен Шанер**. Спасибо всем специалистам из группы технического рецензирования.

Райан никогда не забудет тот день, когда он обнаружил первую книгу серии Head First в книжном магазине. Спасибо Кэти Сьерра и Берту Бэйтсу за то, что они помогли расшевелить нейроны умников по всему миру. Спасибо Берту за то, что он слушал наши жалобы, помогал избавиться от нашей ограниченности и сохранять объективный взгляд на мир ;)

Спасибо Тиму О'Рейли за лучшее издательство технической литературы!


Лу Барр

Друзьям и семье Ронана:

Я особенно благодарен своей жене Кейтлин — ее фантастические способности к дизайну и знание всех тонкостей Adobe позволили этой книге воплотиться в жизни. А еще спасибо за ее терпение — без тебя у меня бы ничего не получилось! Огромное спасибо всем, кто поддерживал нас в этой работе: моим соседям, моим коллегам по университету Портленда, моей футбольной команде и товарищам по гольфу. Спасибо моей ирландской семье за поддержку и помощь. И самое главное — спасибо Райану Бенедетти, моему замечательному соавтору, коллеге и другу. Спасибо, что вывел меня в этот путь. Это было незабываемо!

Друзьям и семье Райана:

Спасибо моей дочери Джози, моему сыну Винни и моей невесте Шонне — они верили в меня и ежедневно поддерживали в работе над книгой. *Ti amo, i miei tre miracoli.* Я люблю вас всех, вы мои три чуда!

Спасибо маме и папе; моему брату Джеффу; моим племянницам Клэр и Квинн. Спасибо моим коллегам и группе WAS из университета Портленда — а именно Дженни Уолш, Джейкобу Канипароли и «вторниковой» технической группе (сами знаете, о ком я). Спасибо Кейтлин Пирс-Крэнли за ее выдающиеся навыки дизайна. Спасибо моему приятелю «ирландскому ниндзя» (также известному как Ронан Крэнли) за его мастерство программирования jQuery, JavaScript и PHP; его чувство юмора и невероятную трудовую дисциплину в работе над книгой.

Живые веб-страницы

Вы хотите расширить возможности своих веб-страниц. Вы уже знаете HTML и CSS и хотите включить в свой арсенал сценарное программирование, но вам совершенно не хочется проводить свою жизнь за написанием сотен строк кода. Нужна библиотека сценариев, которая позволяла бы изменять веб-страницы «на ходу». И если на то пошло, как насчет поддержки AJAX и PHP? И чтобы в трех строках кода можно было сделать то, для чего в большинстве клиентских языков потребуется 15? Пустые мечты, скажете вы... А вот и нет! На помощь приходит jQuery.

Новые возможности веб-страниц

Вы уже умеете строить замечательные веб-страницы из чистого, синтаксически правильного кода HTML и CSS. Но статические веб-страницы сегодня никому не нужны – пользователи требуют динамики, анимации, интерактивности и современных эффектов.

Возьми в руку карандаш

Хотите управлять своими веб-страницами и сделать их более удобными для посетителей? Отметьте все нужные пункты в следующем списке.

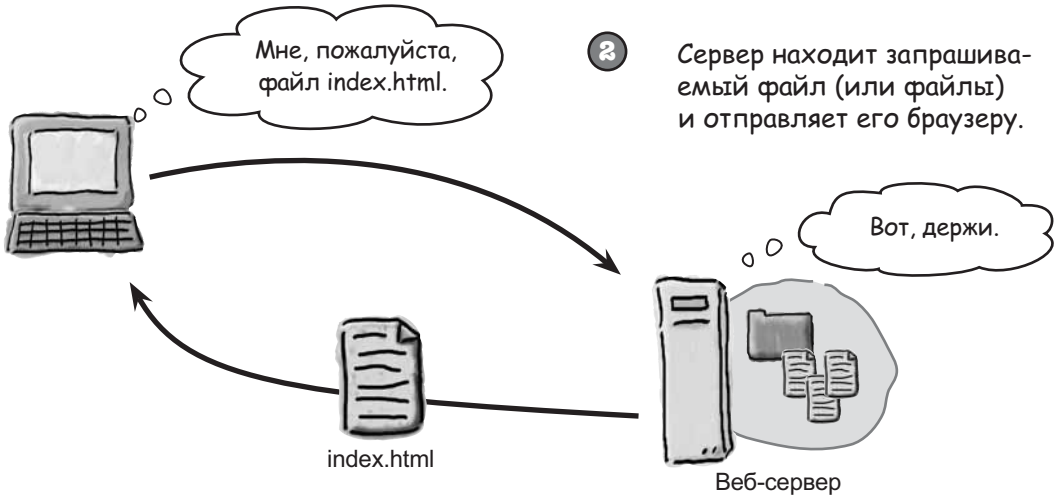
- ☐ Динамическое включение элементов в веб-страницу без ее полной перезагрузки.
- ☐ Изменение элементов меню при наведении на них указателя мыши.
- ☐ Оповещение пользователя о незаполненных полях формы.
- ☐ Поддержка движения и переходов в тексте и графике.
- ☐ Загрузка данных с сервера в тот момент, когда они понадобятся пользователю.

HTML и CSS — это, конечно, хорошо, но...

Старые добрые технологии HTML и CSS определяют структуру и стилевое оформление веб-страницы. Готовая страница прекрасно отображается в браузере, но она остается *статической*.

А если понадобится изменить ее внешний вид, что-то добавить или удалить? Приходится либо идти на немыслимые выкрутасы с CSS, либо просто загружать новую страницу. Оба варианта оставляют желать лучшего. Почему? Да потому что HTML и CSS предназначены для управления внешним видом страницы.

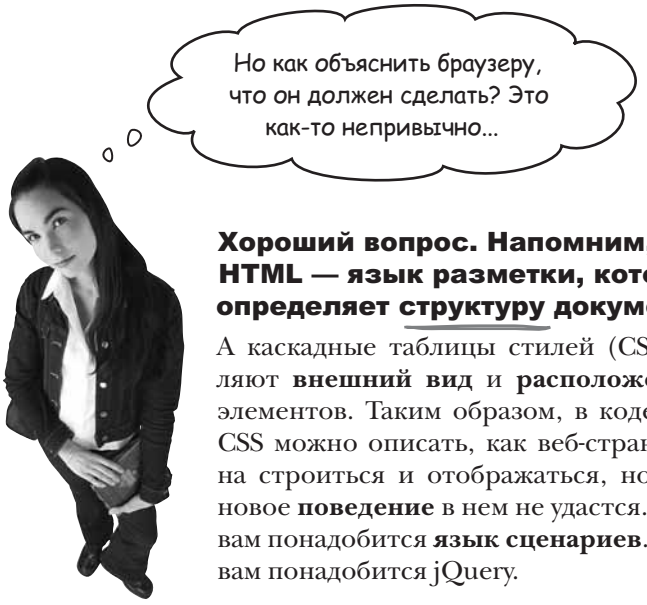
1 Когда пользователь вводит веб-адрес в адресной строке, браузер запрашивает веб-страницу у сервера.



3 Браузер отображает страницу HTML на основании файла, полученного от сервера.

...без сценариев не обойтись

Чтобы изменять веб-страницы «на ходу», без полной перезагрузки, необходимо отдать соответствующую команду браузеру. Как это сделать? При помощи тега HTML `<script>`.



Хороший вопрос. Напомним, что HTML — язык разметки, который определяет структуру документа.

А каскадные таблицы стилей (CSS) определяют **внешний вид** и **расположение** этих элементов. Таким образом, в коде HTML и CSS можно описать, как веб-страница должна строиться и отображаться, но добавить новое **поведение** в нем не удастся. Для этого вам понадобится **язык сценариев**. Для этого вам понадобится jQuery.

Знакомьтесь: jQuery (и JavaScript)!

Язык, используемый для передачи инструкций браузеру, называется JavaScript. В каждом браузере имеется встроенный интерпретатор JavaScript, который получает инструкции из тегов <script> и преобразует их в операции с веб-страницей.

Чтобы передать инструкции интерпретатору, необходимо знать JavaScript. Не беспокойтесь! jQuery вам в этом поможет. jQuery — **библиотека** JavaScript, предназначенная для изменения документов веб-страниц «на ходу». Рассмотрим пример использования jQuery.

jQuery — библиотека JavaScript, предназначенная для изменения документов веб-страниц «на ходу».

Возьми в руку карандаш

Следующий сценарий динамически изменяет веб-страницу. Прочитайте каждую строку и попробуйте предположить, что она делает, руководствуясь своими знаниями HTML и CSS. Затем запишите краткое описание каждой строки справа. Если в чем-то не уверены — попытайтесь угадать, это абсолютно нормально. Описание одной строки мы заполнили за вас.

```
<script>
$(document).ready(function(){

  $("button").click(function(){
      $("h1").hide("slow");
      $("h2").show("fast");
      $("img").slideUp();
  });
});
</script>
```

Когда документ веб-страницы будет готов, выполнить следующие действия.

Возьми в руку карандаш
Решение

Следующий сценарий динамически изменяет веб-страницу. Прочитайте каждую строку и попробуйте предположить, что она делает, руководствуясь своими знаниями HTML и CSS. Затем запишите краткое описание каждой строки справа. Если в чем-то не уверены — попытайтесь угадать, это абсолютно нормально. Ниже приведено наше решение.

```
<script>
$(document).ready(function(){

  $("button").click(function(){

      $("h1").hide("slow");

      $("h2").show("fast");

      $("img").slideUp();
  });
});
</script>
```

Когда документ веб-страницы будет готов, выполнить следующие действия.
При щелчке на любой кнопке сделать следующее:
Все элементы h1 медленно исчезают со страницы.
Все элементы h2 медленно появляются на странице.
Все элементы img скользят вверх и исчезают.
Конец функции click.
Конец функции ready.

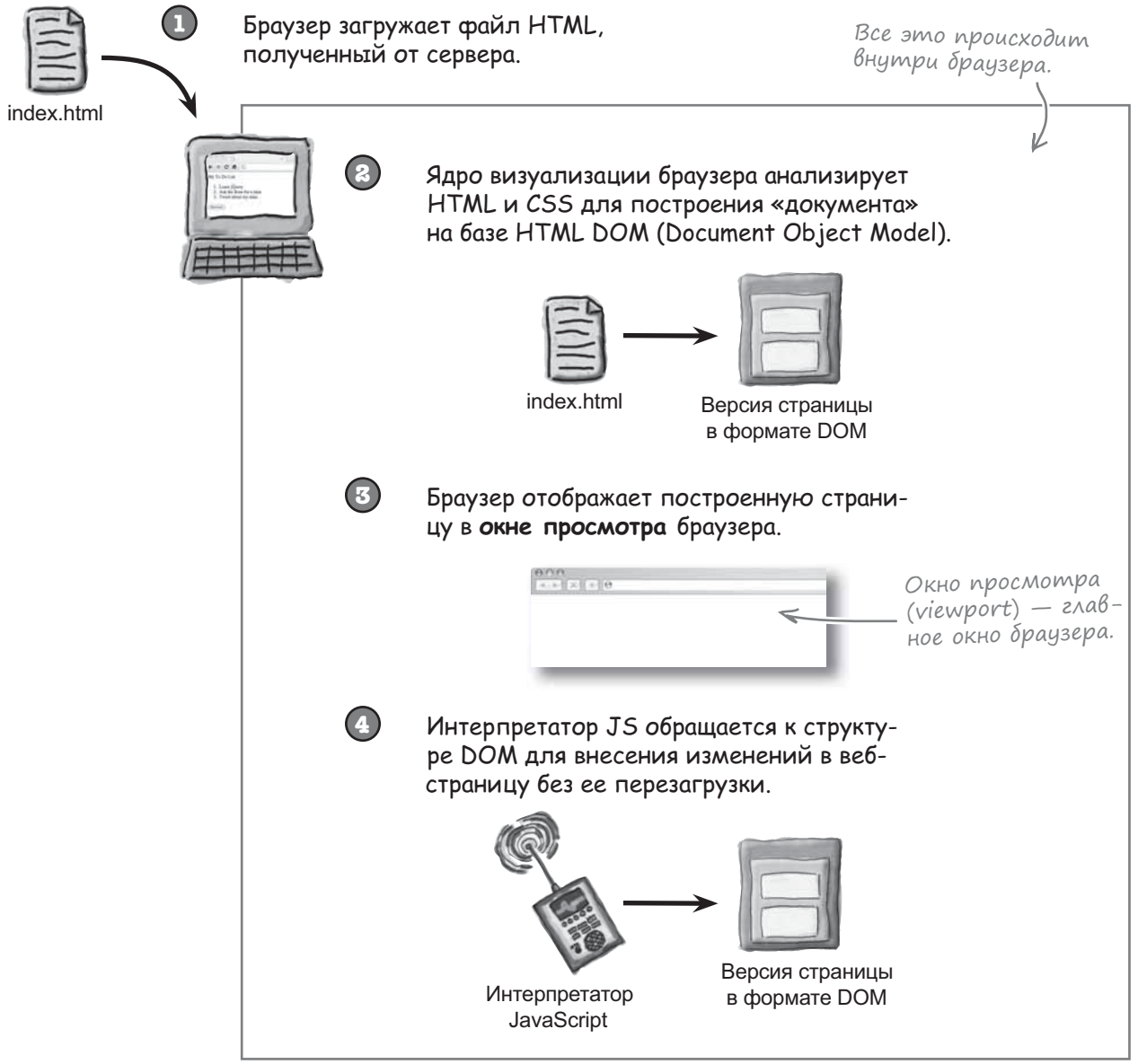
Но если я не обновляю страницу, как браузер узнает, что элемент нужно скрыть или переместить вверх?

Хороший вопрос. Чудеса какие-то, верно?

Давайте взглянем на веб-страницу с точки зрения браузера. Итак, как же jQuery может изменять страницу *из* браузера?

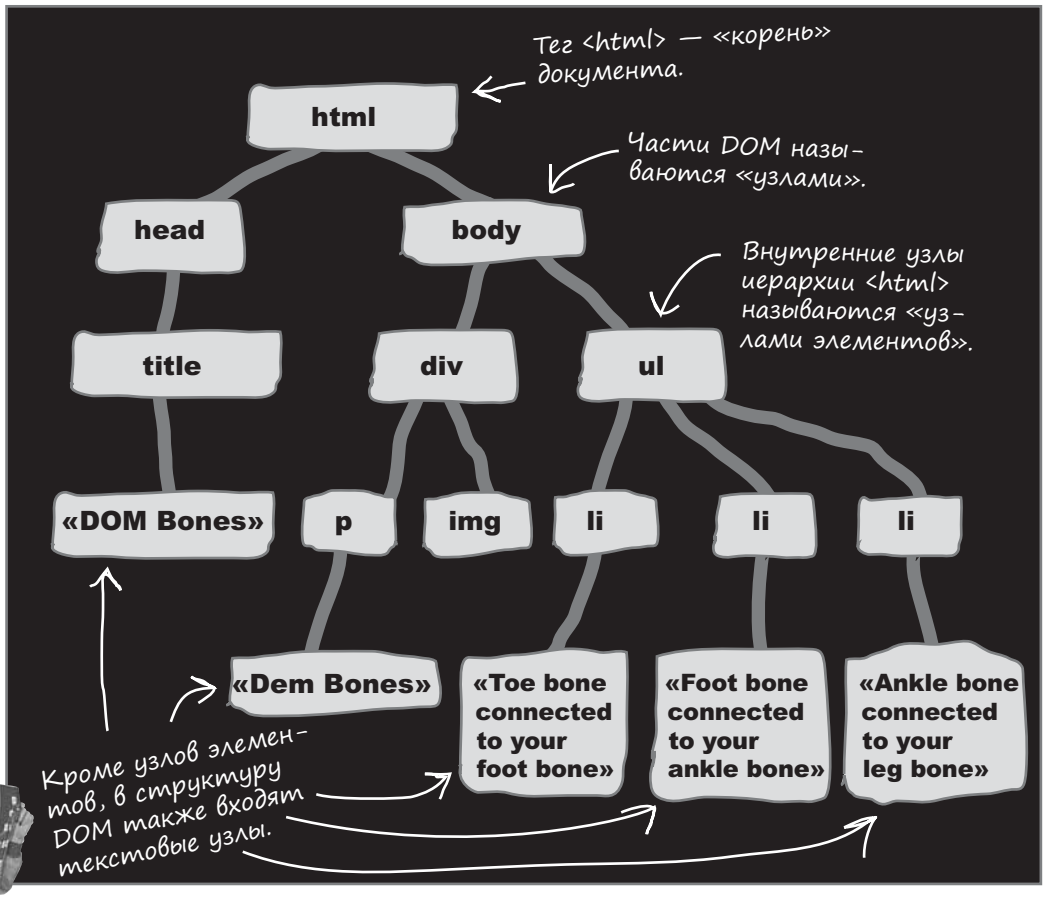
Что происходит в браузере

Пора разобраться, что же в действительности происходит в браузере при отображении веб-страницы. Ваш браузер использует модель HTML DOM (Document Object Model) для преобразования простой разметки HTML и кода CSS в полноценную страницу с текстом, графикой, видеороликами и прочим замечательным контентом, который мы с таким удовольствием просматриваем.



Скрытая структура веб-страницы

Долгие годы структура DOM помогала HTML, CSS и JavaScript эффективно работать в сочетании друг с другом. DOM образует стандартный «скелет» страницы, который используется всеми современными браузерами для повышения эффективности просмотра. Структуру DOM часто сравнивают с деревом: у нее тоже есть *корень* и *ветви*, завершающиеся *узлами*. Или, если хотите, DOM напоминает рентгеновский снимок построенной страницы.



На рентгеновском снимке врач видит скрытую структуру человеческого тела. Аналогичным образом DOM представляет скрытую структуру веб-страницы. Но в отличие от рентгенограмм JavaScript и jQuery могут использовать DOM для *изменения структуры* страницы.

jQuery упрощает работу с DOM

На первый взгляд может показаться, что операции с DOM слишком сложны, но, к счастью, использование jQuery сильно упрощает их. Не забудьте: операции jQuery тоже выполняются из кода JavaScript, но в куда более удобном виде. Предположим, вы хотите изменить код HTML, содержащийся в ***единственном* элементе абзаца на странице.**

В обычном коде JavaScript

Операция выполняется с основным документом.

Получить все элементы с именем тега «p».

```
document.getElementsByTagName("p")
[0].innerHTML = "Change the page.";
```

Выбрать нулевой элемент.

Записать в код HTML этого элемента...

...это значение.

В jQuery

Получить элемент абзаца.

Заменить код HTML этого элемента значением в круглых скобках.

```
$("p").html("Change the page.");
```

В jQuery используется «механизм селекторов»; таким образом, для получения элементов можно использовать селекторы, как это делается в CSS.

Или допустим, вы хотите изменить код HTML в ***пяти*** элементах абзацев нашей страницы:

В цикле перебираем элементы, которые требуется изменить.

```
for (i = 0; i <= 4; i++)
{
   document.getElementsByTagName("p")
[i].innerHTML="Change the page";
}
```

Получить текущий элемент.

Так как в jQuery используются селекторы CSS, эта операция выполняется так же, как и предыдущая.

```
$("p").html("Change the page.");
```

Одно из главных преимуществ jQuery заключается в том, что библиотека позволяет работать с DOM, почти ничего не зная об этой структуре. Всю «черную работу» выполняет код библиотеки. В этой книге вы научитесь использовать JavaScript в сочетании с jQuery. В главе 6 отношения между jQuery и JavaScript будут рассмотрены более подробно, а заодно вы повысите свою квалификацию в области JavaScript. А до этого момента все операции с DOM будут выполняться средствами jQuery.

А сейчас пришло время опробовать jQuery и DOM в действии...

Готово к употреблению

Введите следующий код в текстовом редакторе. Сохраните его, откройте в браузере и проверьте, как работает каждая кнопка. (И раз уж на то пошло, просмотрите код и попробуйте разобраться, как он работает...)

```
<!DOCTYPE html>
<html><head> <title>jQuery goes to DOM-ville</title>
<style>
       #change_me {
       position: absolute;
       top: 100px;
       left: 400px;
       font: 24px arial;}

       #move_up #move_down #color #disappear {
       padding: 5px;}
</style>
<script src="scripts/jquery-1.6.2.min.js"></script>
</head>
<body>
       <button id="move_up">Move Up</button>
       <button id="move_down">Move Down</button>
       <button id="color">Change Color</button>
       <button id="disappear">Disappear/Re-appear</button>

       <div id="change_me">Make Me Do Stuff!</div>
       <script>
              $(document).ready(function() {
                     $("#move_up").click( function() {
                            $("#change_me").animate({top:30},200);
                     });//end move_up
                     $("#move_down").click( function() {
                            $("#change_me").animate({top:500},2000);
                     });//end move_down
                     $("#color").click( function() {
                            $("#change_me").css("color", "purple");
                     });//end color
                     $("#disappear").click( function() {
                            $("#change_me").toggle("slow");
                     });//end disappear
              });//end doc ready
       </script>
</body>
</html>
```

index.html

Как *это* работает?

Манипуляции jQuery со страницей выглядят довольно впечатляюще, верно? Важно помнить, что при нажатии кнопок исходный код HTML и CSS остается неизменным. **Как же jQuery творит все эти чудеса? Взгляните на схему**.

Интерпретатор JavaScript не изменяет исходные файлы HTML и CSS. Он вносит изменения в представление страницы в формате DOM, хранящееся в памяти браузера.

Знак доллара ($) показывает, сколько денег вы сможете зарабатывать со своими новыми навыками jQuery... Шутка — однако знак $ играет очень важную роль в мире jQuery.

Функция jQuery (и ее сокращенная запись)

Знак доллара со скобками — это сокращенная запись *функции* jQuery. Мы используем ее вместо записи «`jQuery()`», когда хотим обозначить функцию jQuery (ее также называют «оберткой»).

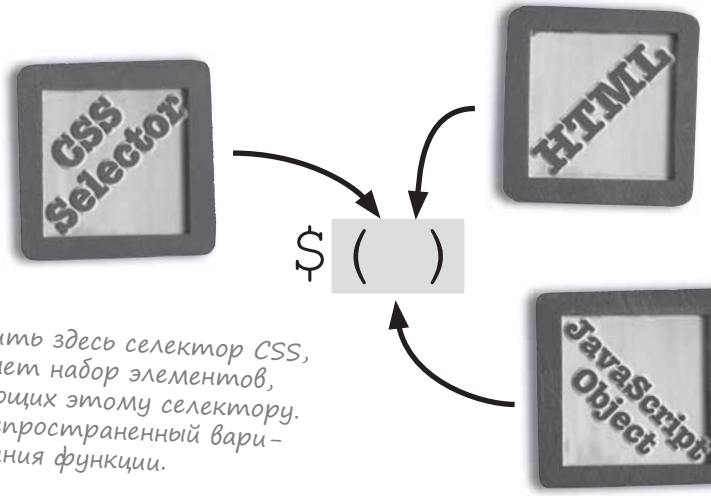
```
jQuery( )
```

Функция jQuery возвращает элементы, определяемые условиями в круглых скобках.

```
$( )
```

Сокращенная запись для jQuery. Вместо шести символов, образующих имя «jQuery», достаточно ввести всего один символ.

Оба имени, короткое и длинное, обозначают одно и то же: длинный блок программного кода. В этой книге будет использоваться сокращенная запись. В круглых скобках функции jQuery могут передаваться аргументы трех видов.



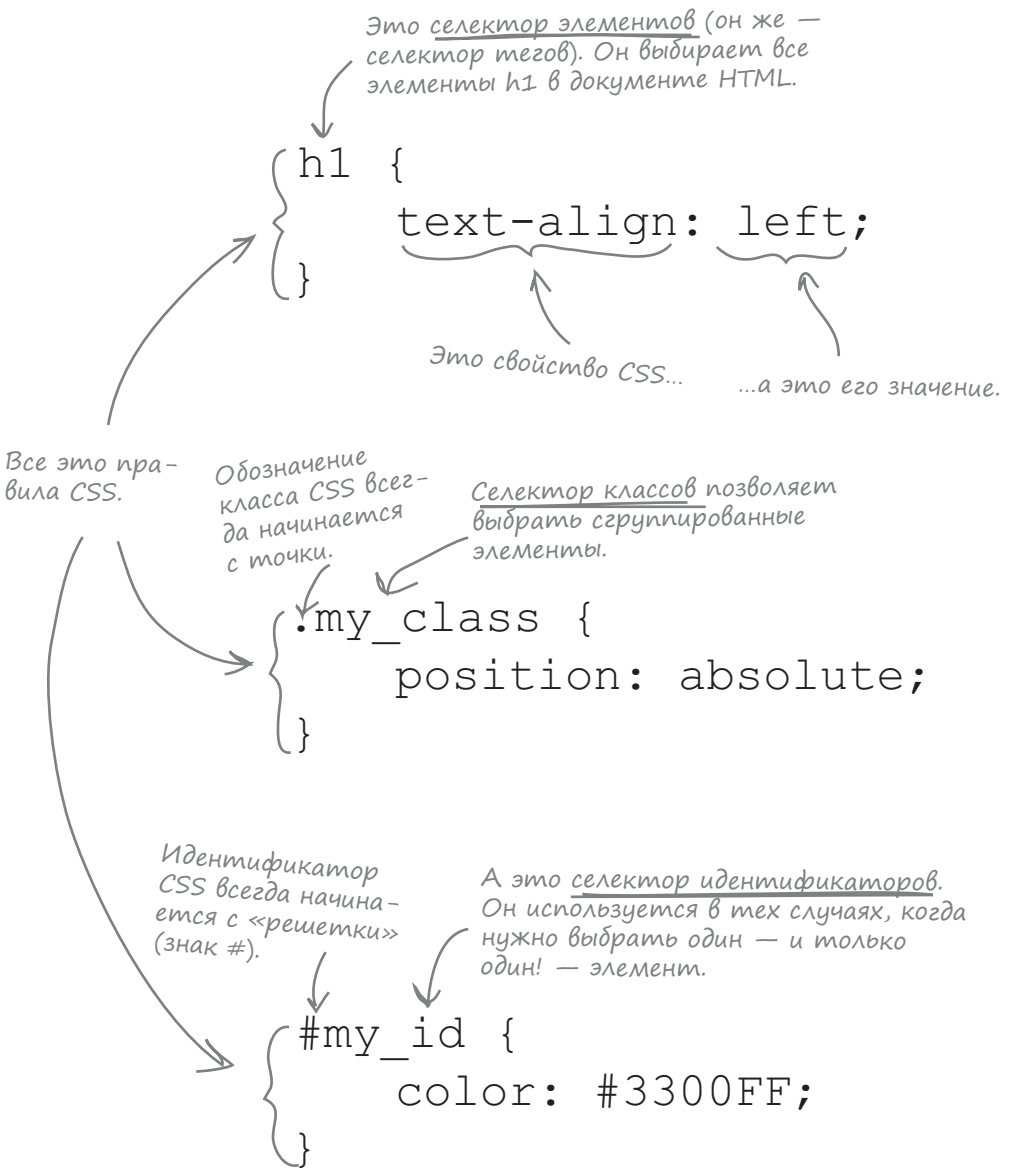
Если в скобках заключена строка HTML, то вы сможете добавлять элементы DOM в страницу «на ходу».

Если разместить здесь селектор CSS, то jQuery вернет набор элементов, соответствующих этому селектору. Это самый распространенный вариант использования функции.

На этот вариант пока не обращайте внимания. Он будет рассмотрен в следующих главах.

jQuery выбирает элементы по тем же правилам, что и CSS

Вы уже знаете о jQuery больше, чем может показаться на первый взгляд. Для выбора элементов в jQuery используются **селекторы** — те же, которые вы использовали в CSS. Если вы подзабыли, как работают селекторы CSS, краткая сводка поможет вам освежить память.



Селекторы: стили и сценарии

В jQuery селекторы, используемые в CSS для стилевого оформления страницы, используются для ***выполнения операций*** с элементами страницы.

Селектор CSS

Селектор элементов

```
h1 {
     text-align: left;
}
```

Селектор классов

```
.my_class{
     position: absolute;
 }
```

Селектор идентификаторов

```
#my_id {
     color: #3300FF;
};
```

Селектор jQuery

Селектор элементов jQuery

Метод

```
$("h1").hide();
```

Скрывает все элементы h1 на странице.

Селектор классов jQuery

Метод

```
$(".my_class").slideUp();
```

Сдвигает вверх все элементы, входящие в класс CSS my_class.

Селектор идентификаторов jQuery

Метод

```
$("#my_id").fadeOut();
```

Команда «растворяет» элемент с идентификатором CSS my_id до тех пор, пока он не станет невидимым.

Селекторы CSS выбирают элементы для применения стилей; селекторы jQuery выбирают элементы для добавления нового поведения.

В главе 2 (и в остальных главах) мы продолжим объединять селекторы с методами.

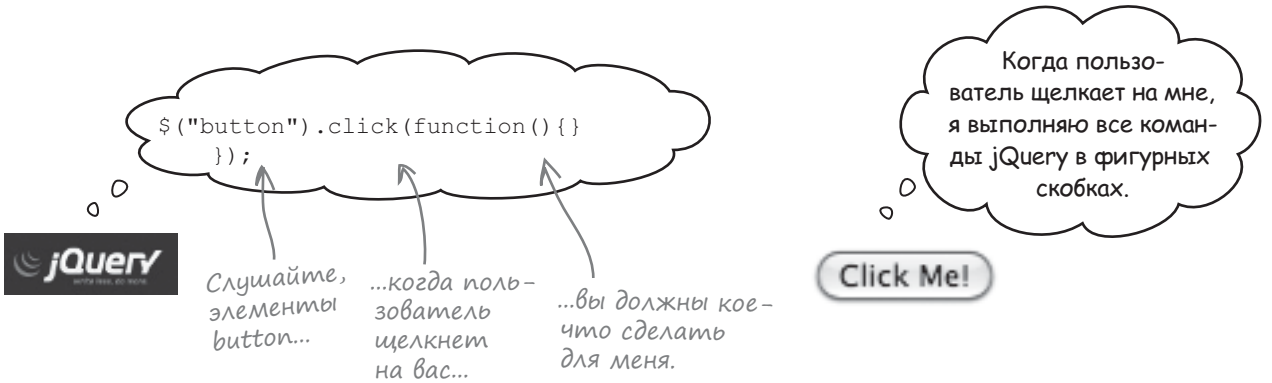
Использование селекторов jQuery

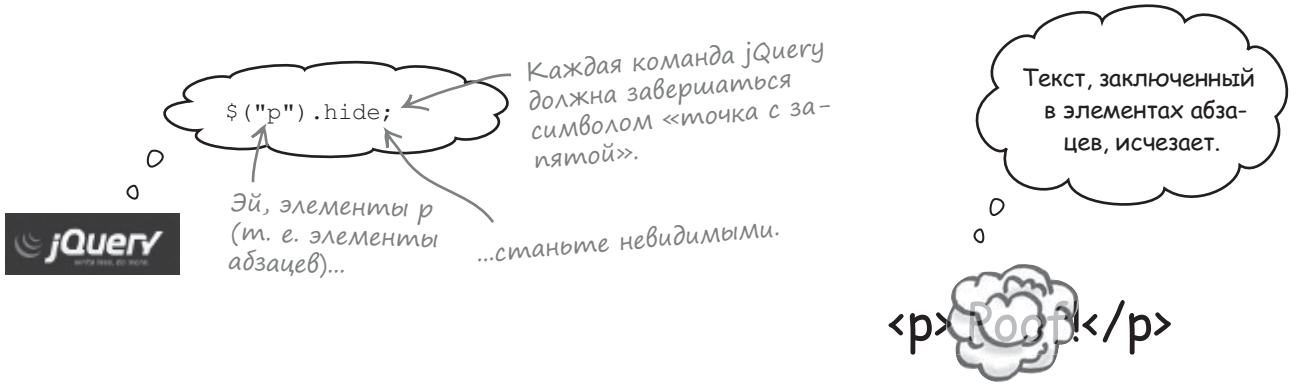
Как подсказывает само название, библиотека jQuery предназначена для создания ***запросов*** (querying). Вы что-то описываете при помощи селектора, а интерпретатор JavaScript запрашивает это «что-то» у структуры DOM. Если запрашиваемый элемент содержит вложенные элементы, то jQuery вернет и их. Давайте рассмотрим селектор jQuery, чтобы вы лучше поняли, как он работает.

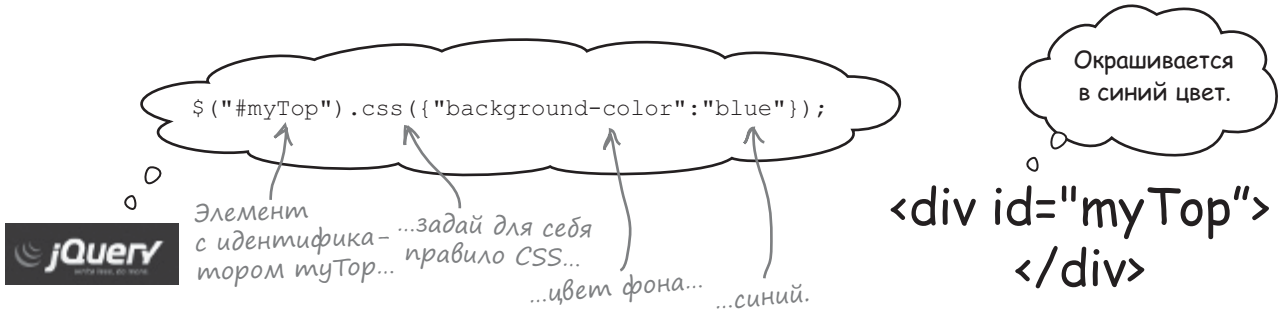
О том, что делать с полученными элементами, будет рассказано в главе 4. А пока нас интересует только механизм выбора этих элементов.

jQuery в переводе

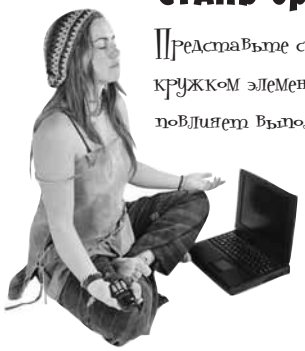
Чтобы показать вам, как легко пользоваться jQuery, мы разберем несколько выражений на языке jQuery, которые могли бы вам пригодиться в ваших путешествиях по стране DOM.







СТАНЬ браузером



Представьте себя на месте браузера. Обведите кружком элементы HTML (справа), на которые повлияет выполнение команды jQuery (слева).

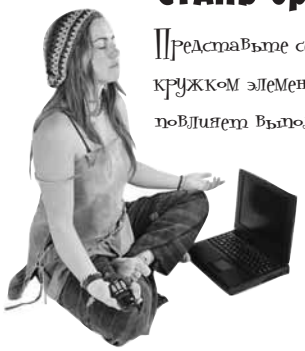
Команда jQuery	Элементы HTML

`$("p").hide();`	`<p>Проснувшись однажды утром после беспокойного сна. . .</p>`
	
	`<p>Грегор Замза обнаружил, что он у себя в постели превратился в страшное насекомое.</p>`
	
	`<p>Лежа на панцирнотвердой спине, он видел, стоило ему приподнять голову. . . </p>`

`$("span.Italian").toggle();`	`<span class="Italian">Nel Mezzo del cammin di nostra vita</span>`
	
	`<span class="English">In the middle of this road called "our life"</span>`
	
	`<span class="Italian">mi ritrovai per una selva oscura</span>`

`$("p#mytext").show();`	`<p id="mytext">Проснувшись однажды утром после беспокойного сна. . .`
	`</p>`
	
	`<p id="mytext">Грегор Замза обнаружил, что он у себя в постели превратился в страшное насекомое.</p>`
	
	`<p>Лежа на панцирнотвердой спине, он видел, стоило ему приподнять голову. . . </p>`

СТАНЬ браузером — решение



Представьте себя на месте браузера. Обведите кружком элементы HTML (справа), на которые повлияет выполнение команды jQuery (слева).

Команда jQuery

Элементы HTML

```
$("p").hide();
```

```
<p>Проснувшись однажды утром после беспокойного
сна. . .</p>

<p>Грегор Замза обнаружил, что он у себя в
постели превратился в страшное насекомое.</p>

<p>Лежа на панцирнотвердой спине, он видел,
стоило ему приподнять голову. . . </p>
```

```
$("span.Italian").toggle();
```

```
<span class="Italian">Nel Mezzo del cammin di
nostra vita</span>

<span class="English">In the middle of this road
called "our life"</span>

<span class="Italian">mi ritrovai per una selva
oscura</span>
```

```
$("p#mytext").show();
```

```
<p id="mytext">Проснувшись однажды утром после
беспокойного сна. . .
</p>

<p id="mytext">Грегор Замза обнаружил, что
он у себя в постели превратился в страшное
насекомое.</p>

<p>Лежа на панцирнотвердой спине, он видел,
стоило ему приподнять голову. . . </p>
```

Часто Задаваемые Вопросы

В: Зачем было создавать jQuery, если библиотека только использует JavaScript? Разве одного JavaScript недостаточно?

О: JavaScript хорошо подходит для многих задач (и особенно для манипуляций с DOM), но решение получается довольно сложным. Низкоуровневые операции с DOM — дело в лучшем случае непростое. Именно здесь на помощь приходит jQuery: библиотека скрывает многие сложности, связанные с операциями с DOM, и благодаря ей самые впечатляющие эффекты создаются на удивление просто. (Библиотеку jQuery создал Джон Резиг; дополнительная информация об авторе доступна по по адресу *http://ejohn.org/about*).

В: Зачем нужны все эти знаки доллара?

О: Это просто сокращенное обозначение, чтобы вам не приходилось снова и снова вводить «jQuery»! Впрочем, если вы работаете с другими языками на стороне клиента, полная запись `jQuery()` поможет избежать конфликтов имен.

В: Ранее вы уже упоминали о «клиентских сценариях». Что это такое?

О: Веб-разработчики часто называют браузер *клиентом*, потому что он использует данные, полученные с (веб-)сервера. Клиентский язык сценариев управляет работой браузера, тогда как серверные языки отдают инструкции серверу. Эта тема более подробно рассматривается в главах 8 и 9.

В: Как появилась модель DOM?

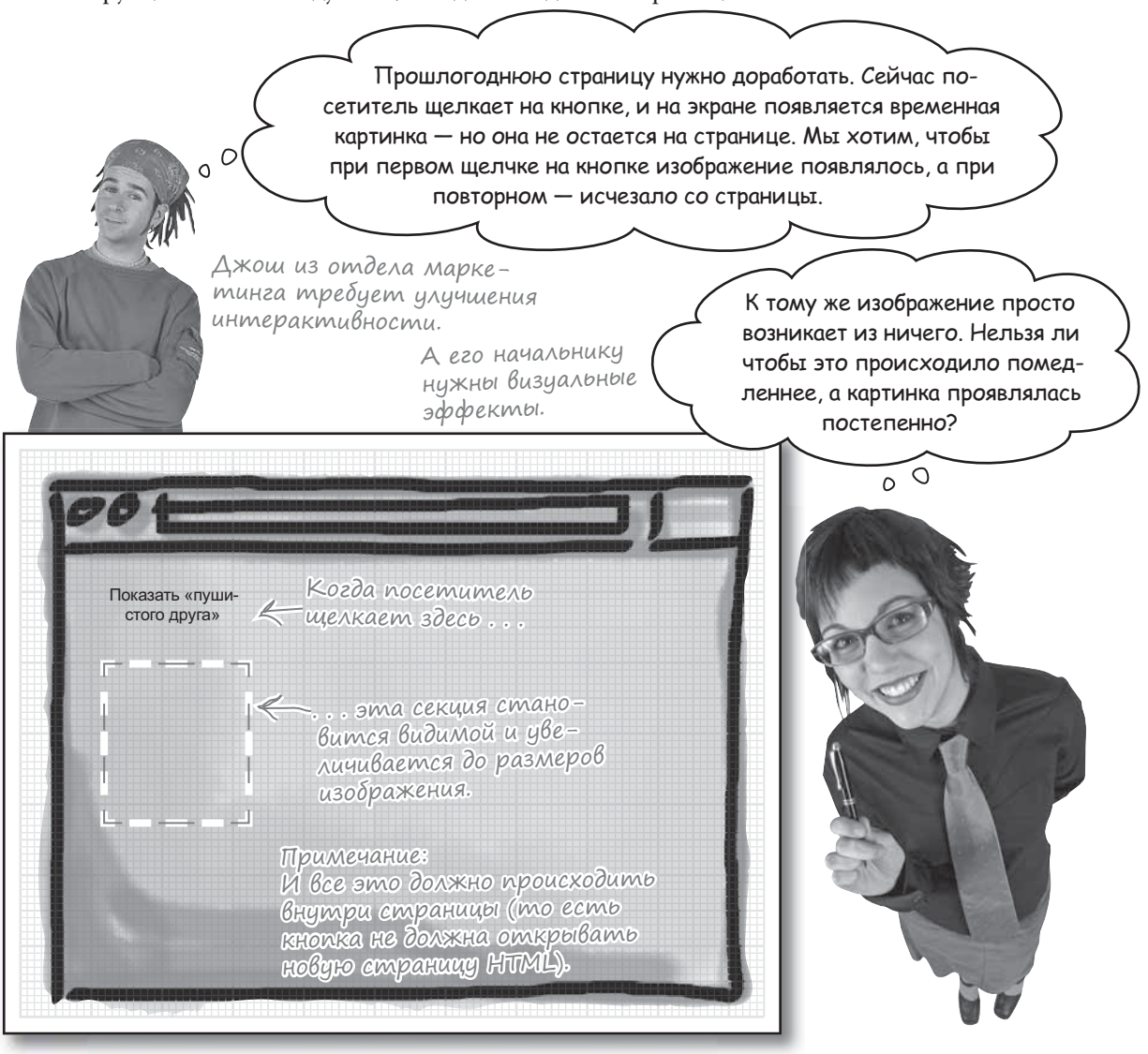
О: Хороший вопрос. Вебдизайнеры и разработчики устали от несовместимости браузеров и решили, что им необходим стандартный механизм добавления поведения и взаимодействия со страницами в любом браузере. Комитет W3C (World Wide Web Consortium) разработал стандарт DOM при участии многочисленных заинтересованных групп. За дополнительной информацией обращайтесь по адресу *http://w3.org/dom*.

В: Загрузка jQuery доступна в двух вариантах: рабочая версия и версия для разработчиков. Чем они отличаются друг от друга?

О: Рабочая версия более компактна, из нее удалены избыточные символы и пробелы. Она оптимизирована для достижения максимального быстродействия в ходе реальной эксплуатации, но в ней труднее разобраться. В версии для разработчиков расставлены удобные пробелы, она лучше читается. Эта версия предназначена для программистов, которые хотят покопаться в коде jQuery с целью его изменения и даже расширения (раз уж это проект с открытым исходным кодом!).

Ваш первый проект с jQuery

Вы только что поступили на должность веб-разработчика в Фонд спасения домашних питомцев. Группа маркетинга хочет запустить ежегодную кампанию по привлечению средств, в которой должна быть задействована переработанная версия прошлогодней веб-страницы «Поможем нашим пушистым друзьям». Вам выдали снимок экрана из прошлогодней кампании с подробными инструкциями по поводу того, что должна делать страница.



Никто не хочет подвести отдел маркетинга в первый же день — с этими ребятами лучше не ссориться! Для начала посмотрим, как была реализована старая версия страницы.

Возьми в руку карандаш

Прежде чем разбираться с тем, как включить в страницу функциональность jQuery, стоит посмотреть, как выглядел код HTML и CSS предыдущей версии. Ниже приведено содержимое файла из прошлогодней кампании. Найдите элементы, которые, по вашему мнению, понадобятся вам в реализации. Напишите рядом с каждым элементом, что необходимо сделать для выполнения требований отдела маркетинга. Первое описание мы заполнили за вас.

```
<!DOCTYPE html><html> <head>
<title>Furry Friends Campaign: jQuery
Proof-of-Concept</title>
<link rel="stylesheet" type="text/css"
href="styles/my_style.css">
</head>
<body>
<div id="showfriend">
<a href="#">Our Furry Friends Need
Your Help
<img src="images/furry_friend.jpg">
</a>
</div>
```

index.html

У якорного тега имеются состояния «hover» и «active», заданные в CSS. Пользователь наводит указатель мыши на ссылку — изображение появляется.

```
a:link img, a:visited img {
display:none;
}

a:hover img, a:active img {
display:block;
}
a{
text-decoration:none;
color: #000;
}
```

my_style.css

Возьми в руку карандаш
Решение

Прежде чем разбираться с тем, как включить в страницу функциональность jQuery, стоит посмотреть, как выглядел код HTML и CSS предыдущей версии. Ниже приведено содержимое файла из прошлогодней кампании. Найдите элементы, которые, по вашему мнению, понадобятся вам в реализации. Напишите рядом с каждым элементом, что необходимо сделать для выполнения требований отдела маркетинга. Не беспокойтесь, если ваши ответы где-то отличаются от наших.

```
<!DOCTYPE html><html> <head>
<title>Furry Friends Campaign: jQuery
Proof-of-Concept</title>
<link rel="stylesheet" type="text/css"
href="styles/my_style.css">
</head>
<body>
<div id="showfriend">
<a href="#">Our Furry Friends Need
Your Help
<img src="images/furry_friend.jpg">
</a>
</div>
```

index.html

У якорного тега имеются состояния «hover» и «active», заданные в CSS. Пользователь наводит указатель мыши на ссылку — изображение появляется.

Изображение встроено в якорный тег. Оно не должно отображаться до тех пор, пока пользователь не щелкнет на ссылке якорного тега.

```
a:link img, a:visited img {
display:none;
}

a:hover img, a:active img {
display:block;
}
a{
text-decoration:none;
color: #000;
}
```

my_style.css

Этот селектор CSS задает свойству display вложенного изображения значение «none», чтобы оно оставалось скрытым при загрузке страницы.

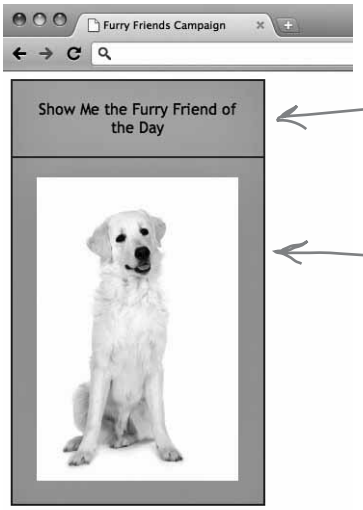
Когда пользователь наводит указатель мыши или щелкает на якорном теге, свойство display элемента img принимает значение «block». Изображение мгновенно появляется на экране.

Можно, но сначала нужно подготовиться.

Прежде чем использовать jQuery для реализации всех эффектов, которые желает видеть отдел маркетинга, мы должны убедиться в том, что у jQuery есть все необходимое для работы. Как вы уже знаете, одна из главных задач jQuery — манипуляции с элементами HTML, поэтому нам понадобится хорошая *структура*. Для выбора элементов jQuery использует те же селекторы, что и CSS; следовательно, нам также понадобятся хорошо определенные *стили*.

Вернемся к требованиям

Размышляя о структуре, всегда полезно вернуться на шаг назад и понять, что же вы, собственно, пытаетесь построить. Отдел маркетинга хочет, чтобы изображение опускалось и проявлялось при щелчке в области страницы «Show Me the Furry Friend of the Day». Какие изменения в HTML и CSS для этого нужно внести?



Преобразуем в область div с отслеживанием щелчков.

Изображение тоже оформляется в виде области div, которая изначально находится в скрытом состоянии. Мы присвоим этой области идентификатор picframe.

Подготовка файлов HTML и CSS

Задание!

Прежде чем браться за написание команд jQuery, следует подумать, какие изменения необходимо внести в файлы HTML и CSS. Откройте файлы jQuery главы 1 (если вы еще не загрузили их, обращайтесь к разделу «Как работать с книгой» во Введении). Найдите среди файлов главы 1 папку *Begin*. Включите в файлы код, выделенный ниже жирным шрифтом.

```
<!DOCTYPE html>
<html><head>
    <title>Furry Friends Campaign</title>
    <link rel="stylesheet" type="text/css" href="styles/my_style.css">
  </head>
  <body>
    <div id="clickMe">Show Me the Furry Friend of the Day</div>
    <div id="picframe">
     <img src="images/furry_friend.jpg">
    </div>
    <script src="scripts/jquery-1.6.2.min.js"></script>
     <script>
       $(document).ready(function(){
          $("#clickMe").click(function() {

          });
       });
     </script>
  </body>
</html>
```

Этот фрагмент создает область div, отслеживающую щелчки мышью. Применим к области стилевое оформление в файле CSS, чтобы она по внешнему виду не отличалась от области div с идентификатором picframe.

Область div с идентификатором будет «скользить», открывая скрытое изображение.

Изображение furry_friend.jpg вложено в элемент picframe.

index.html

```
#clickMe {
    background: #D8B36E;
    padding: 20px;
    text-align: center;
    width: 205px;
    display: block;
    border: 2px solid #000;
}
#picframe {
    background: #D8B36E;
    padding: 20px;
    width: 205px;
    display: none;
    border: 2px solid #000;
}
```

Этот фрагмент определяет оформление области clickMe, чтобы она не отличалась по виду от области div с идентификатором picframe.

В селекторе picframe задается значение «display: none», чтобы картинка не отображалась при загрузке страницы.

my_style.css

Код под увеличительным стеклом

Итак, файлы HTML и CSS готовы. Пора повнимательнее присмотреться к коду, заключенному в теги `<script>`.

При первой же возможности я начну выполнять код в фигурных скобках!

Эй, DOM...

...когда загрузка завершится и все будет готово к работе...

...сделай кое-что для меня.

```
$(document).ready(function(){
```

Селектор идентификаторов для области clickMe.

```
  $("#clickMe").click(function()
  {
```

Точка отделяет селектор от метода.

Кнопка с идентификатором clickMe связывается с событием click. После этого кнопка начинает отслеживать щелчки мышью.

Код, который должен выполняться при нажатии кнопки, заключается в фигурные скобки («блок кода»).

```
  });
```

Точка с запятой завершает конструкцию jQuery click.

```
});
```

А эта точка с запятой завершает функцию jQuery ready.

В описании встречается много новых терминов.

Вскоре все эти события, методы и функции будут рассмотрены более подробно.

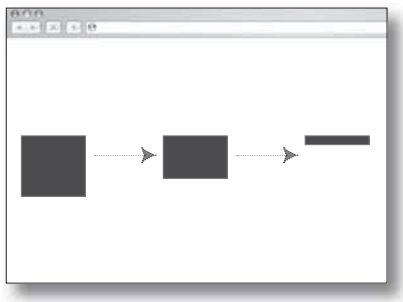
Верно. Разметка HTML и код CSS готовы; пора переходить к jQuery.

Вместо мгновенного появления содержимое `picframe` должно разворачиваться и и постепенно проявляться на странице. К счастью, создатели jQuery предусмотрели эффекты для управления обоими визуальными действиями. Эффектам jQuery в книге посвящена целая глава (глава 5), так что не старайтесь запомнить все сейчас. Начнем с эффекта скольжения.

Поехали...

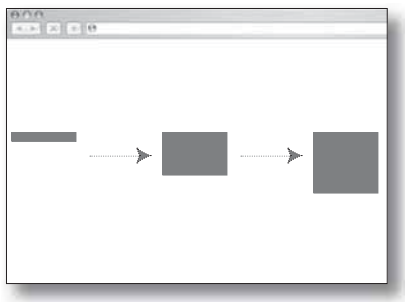
Начнем с эффекта скольжения, которого от нас требует начальник группы маркетинга. Существуют три варианта реализации скольжения:

```
$("div").slideUp();
```



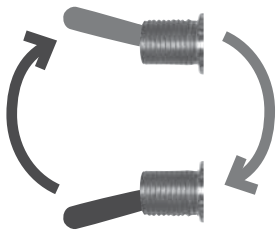
Метод slideUp изменяет свойство height элемента, пока оно не станет равным 0, после чего скрывает элемент.

```
$("div").slideDown();
```



Метод slideDown изменяет свойство height элемента от 0 до значения, заданного в стиле CSS.

```
$("div").slideToggle();
```



Действие slideToggle означает: «Если элемент свернут, развернуть его, а если развернут — свернуть».

Эффекты изменения прозрачности

Мы также хотим, чтобы изображение плавно переходило от полной прозрачности к полной видимости. И для этой категории эффектов в jQuery тоже существуют специальные методы. Имена этих методов очень похожи на имена методов скольжения: `FadeIn`, `FadeOut`, `FadeTo` и `FadeToggle`. В нашем примере будет использоваться метод `FadeIn`, управляющий прозрачностью элементов HTML.

То, что должно проявляться на странице: в данном случае элемент img.

Вы можете задать скорость проявления, указывая нужное значение в круглых скобках. Величина обычно задается в миллисекундах (мс).

```
$("img").fadeIn();
```

Проявляющийся элемент постепенно переходит из состояния полной прозрачности в состояние полной непрозрачности.

МОЗГОВОЙ ШТУРМ

Как вы думаете, сколько команд jQuery потребуется для достижения нужного эффекта?

Запишите эти команды на листке бумаги. Если не уверены, хотя бы попробуйте сформулировать их суть словами; постепенно мы научим вас мыслить понятиями jQuery.

И это все?

Невероятно, но для реализации этих эффектов достаточно написать всего **две строки** кода jQuery. Теперь вы понимаете, почему у jQuery так много поклонников? Включите фрагмент, выделенный жирным шрифтом, в файл *index.html* — вот и все!

```
<!DOCTYPE html>
<html>
  <head>
    <title>Furry Friends Campaign</title>
    <link rel="stylesheet" type="text/css" href="styles/my_style.
css">
  </head>
  <body>
    <div id="clickMe">Show me the Furry Friend of the Day</div>
    <div id="picframe">
    <img src="images/furry_friend.jpg">
    </div>
    <script src="scripts/jquery-1.6.2.min.js"></script>
    <script>
      $(document).ready(function(){
        $("#clickMe").click(function() {

          $("img").fadeIn(1000);
          $("#picframe").slideToggle("slow");

        });
      });
    </script>
  </body>
</html>
```

В jQuery важно выстроить эффекты в такой последовательности, чтобы они не мешали друг другу. Позднее мы еще вернемся к этой проблеме.

Сначала к изображению применяется эффект проявления.

Значения в круглых скобках обеспечивают дополнительную настройку эффектов. За подробностями обращайтесь к главе 5.

index.html

ТЕСТ-ДРАЙВ

Откройте страницу в своем любимом браузере и убедитесь в том, что все работает.

Щелкните здесь.

Изображение постепенно проявляется и разворачивается на странице.

Будьте осторожны!

Проверьте в разных браузерах.

Да, jQuery одинаково работает во всех браузерах — но это не означает, что во всех браузерах будут одинаково работать стили, определенные в файле CSS, или динамические стили, примененные к элементам страницы!

Пушистые Друзья спасены

Вы успешно справились с поставленной задачей — для этого было достаточно слегка изменить HTML и CSS и добавить пару строчек jQuery. Благодарные зверюшки вас не забудут...

Ваш инструментарий jQuery

Глава 1 осталась позади, а ваш творческий инструментарий расширился: в него добавилась основная функция jQuery, селекторы, события click и эффекты изменения прозрачности.

Функция jQuery

Используется для выбора элементов страницы HTML, с которыми выполняется операция.

Сокращенная запись $ избавляет вас от необходимости снова и снова набирать «jQuery».

Функции jQuery могут передаваться селекторы, код HTML и даже объекты JavaScript.

Селектор

В jQuery, как и в CSS, для выбора элементов используются селекторы.

Селекторы jQuery позволяют идентифицировать практически любые элементы HTML.

Эффекты изменения прозрачности

К выбранному элементу можно применять различные эффекты изменения прозрачности с использованием методов FadeIn, FadeOut, FadeTo и FadeToggle.

Эффекты изменения прозрачности могут применяться к любым элементам: тексту, графике и т. д.

Чтобы управлять скоростью изменения, укажите время (в миллисекундах) в круглых скобках в конце команды.

Хватай и действуй

jQuery помогает выбирать элементы веб-страниц и выполнять с ними всевозможные операции. В этой главе более подробно рассмотрены селекторы и методы jQuery. Селекторы jQuery выбирают элементы страницы, а методы выполняют операции с этими элементами. Библиотека jQuery, словно сборник магических заклинаний, позволяет изменять окружающую реальность. Вы можете заставить изображение исчезнуть или появиться из ниоткуда или же выбрать фрагмент текста и анимировать изменение размера его шрифта... Но довольно разговоров — хватайте элементы веб-страниц и действуйте!

Подруга просит тебя помочь оформить сайт

Вы получили сообщение от своей подруги — профессионального фотографа-портретиста. Она хочет провести рекламную акцию «Прыгаем от радости» по получению скидок, и ей нужна ваша помощь в оформлении сайта.

От: **Эмили**
Тема: **Рекламная акция «Прыгаем от радости»!**

Привет,

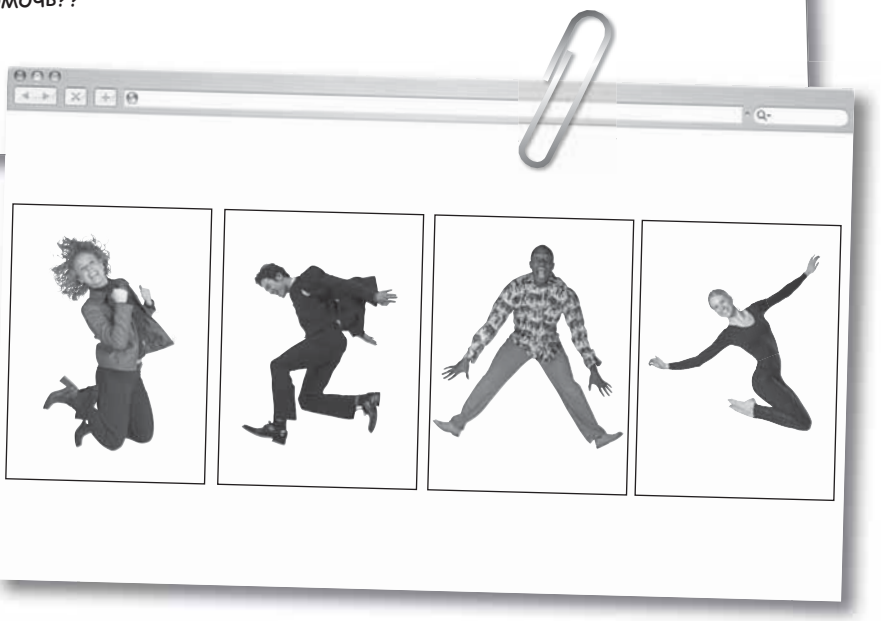
Читала у тебя в Твиттере, что ты занялся интерактивным веб-программированием. Надеюсь, ты поможешь мне с реализацией некоторых интерактивных возможностей для рекламной акции «Прыгаем от радости» на моем сайте. Я хочу, чтобы посетители могли получить скидку перед оформлением своего заказа. Так пользователи будут взаимодействовать с сайтом и проводить на нем больше времени (и хочется надеяться, будут больше покупать!).

Страница должна состоять из четырех областей, по одному из четырех изображений в каждой. Когда пользователь щелкает в одной из областей, под изображением в этой области появляется сообщение «Размер вашей скидки:» и случайное значение (от 5 до 10 процентов). Если пользователь щелкает снова, то старое сообщение исчезает и появляется новое.

Прилагаю набросок того, как это все должно выглядеть.

Сможешь помочь??

--
Эмили



Что требуется от проекта?

Эмили — хороший фотограф, но ее запрос выглядит довольно туманно. Давайте повнимательнее присмотримся к сообщению и попробуем понять, чего же она, собственно, хочет. Прежде чем браться за написание кода jQuery, необходимо предельно четко выяснить, что же требуется от проекта (и от пользователя).

Возьми в руку карандаш

Выделите из сообщения список конкретных задач, которые должно решать веб-приложение. По этому списку мы сможем убедиться в том, что наше веб-приложение отвечает потребностям заказчика.

Список задач:

1.

2.

3.

4.

5.

Преобразование пользовательских *запросов* в непосредственные *требования проекта* — важный навык, который совершенствуется со временем и практикой.

Не забыть!

Возьми в руку карандаш
Решение

Выделите из сообщения список конкретных задач, которые должно решать веб-приложение. По этому списку мы сможем убедиться в том, что наше веб-приложение отвечает потребностям заказчика. Приводим наше решение.

Список задач:

1. *Страница должна состоять из четырех областей. Каждая область содержит одно изображение.*

2. *Области должны реагировать на щелчки мышью.*

3. *Нам понадобится сообщение, состоящее из текста («Размер вашей скидки:») и случайного значения (от 5 до 10 процентов).*

4. *Когда пользователь щелкает в одной из областей, под изображением в этой области должно появляться сообщение.*

5. *Если пользователь щелкает снова, то старое сообщение исчезает и появляется новое.*

Стоп! Не надо торопиться!

Начинать работу над каждым проектом jQuery с проработки требований — безусловно полезная привычка. Но прежде чем приступать к написанию кода jQuery, необходимо немного поработать над структурой и стилями. Мы сделали это в главе 1, а теперь перед переходом к jQuery придется потрудиться еще основательнее.

Начинаем с div

На странице должны находиться четыре области, реагирующие на щелчки мышью; начнем с создания этих областей. Самый полезный и универсальный элемент HTML для наших целей — тег . Он очень хорошо подходит для определения структуры, так как является блочным элементом. Кроме того, внешний вид и поведение элементов div удобно определяется посредством применения стилей.

Упражнение

Откройте свой любимый текстовый редактор и создайте необходимые файлы HTML и CSS. В приведенной ниже заготовке кода отсутствуют некоторые ключевые элементы. Включите в страницу пункты следующего списка, вычеркивая их по мере выполнения.

- ☐ Тег для включения библиотеки jQuery, версия 1.6.2.
- ☐ Тег с идентификатором header.
- ☐ Тег с идентификатором main.
- ☐ Разместите в каждом из четырех элементов div, вложенных в main, отдельное изображение (графические файлы можно загрузить по адресу *www.thinkjquery.com/chapter02/images.zip*).

```
<html>
  <head>
    <title>Jump for Joy</title>
    <link href="styles/my_style.css" rel="stylesheet">
  </head>
  <body>
.....................................................
      <h2>Jump for Joy Sale</h2>
  </div>
.....................................................
      <div><img src="images/jump1.jpg"/></div>
      <div> ..................................</div>
      <div> ..................................</div>
      <div> ..................................</div>
    </div>
.....................................................
<script > </script>  </body>
</html>
```

index.html

```
div{
   float:left;
   height:245px;
   text-align:left;
   border: solid #000 3px;
}
#header{
   width:100%;
   border: 0px;
   height:50px;
}
#main{
   background-color: grey;
   height: 500px;
}
```

my_style.css

Откройте свой любимый текстовый редактор и создайте файлы HTML и CSS, необходимые для этого упражнения. В приведенной ниже заготовке кода отсутствуют некоторые ключевые элементы. После включения пунктов списка ваша страница должна выглядеть так, как показано в следующем решении.

- ☑ Тег для включения библиотеки jQuery, версия 1.6.2.
- ☑ Тег `<div>` с идентификатором `header`.
- ☑ Тег `<div>` с идентификатором `main`.
- ☑ Разместите в каждом из четырех элементов `div`, вложенных в `main`, отдельное изображение.

Ваши файлы HTML и CSS должны выглядеть так.

```
<html>
  <head>
    <title>Jump for Joy</title>
    <link href="styles/my_style.css" rel="stylesheet">
  </head>
  <body>
  <div id="header">
      <h2>Jump for Joy Sale</h2>
  </div>
  <div id="main">
      <div><img src="images/jump1.jpg"/></div>
      <div> <img src="images/jump2.jpg"> </div>
      <div> <img src="images/jump3.jpg"> </div>
      <div> <img src="images/jump4.jpg"> </div>
    </div>
<script src="scripts/jquery-1.6.2.min.js"></script>
<script > </script>  </body>
</html>
```

Элемент div с идентификатором header

Элемент div с идентификатором main

Включение библиотеки jQuery.

index.html

```
div{
        float:left;
        height:245px;
        text-align:left;
        border: solid #000 3px;
}
#header{
        width:100%;
        border: 0px;
        height:50px;
}
#main{
        background-color: grey;
        height: 500px;
}
```

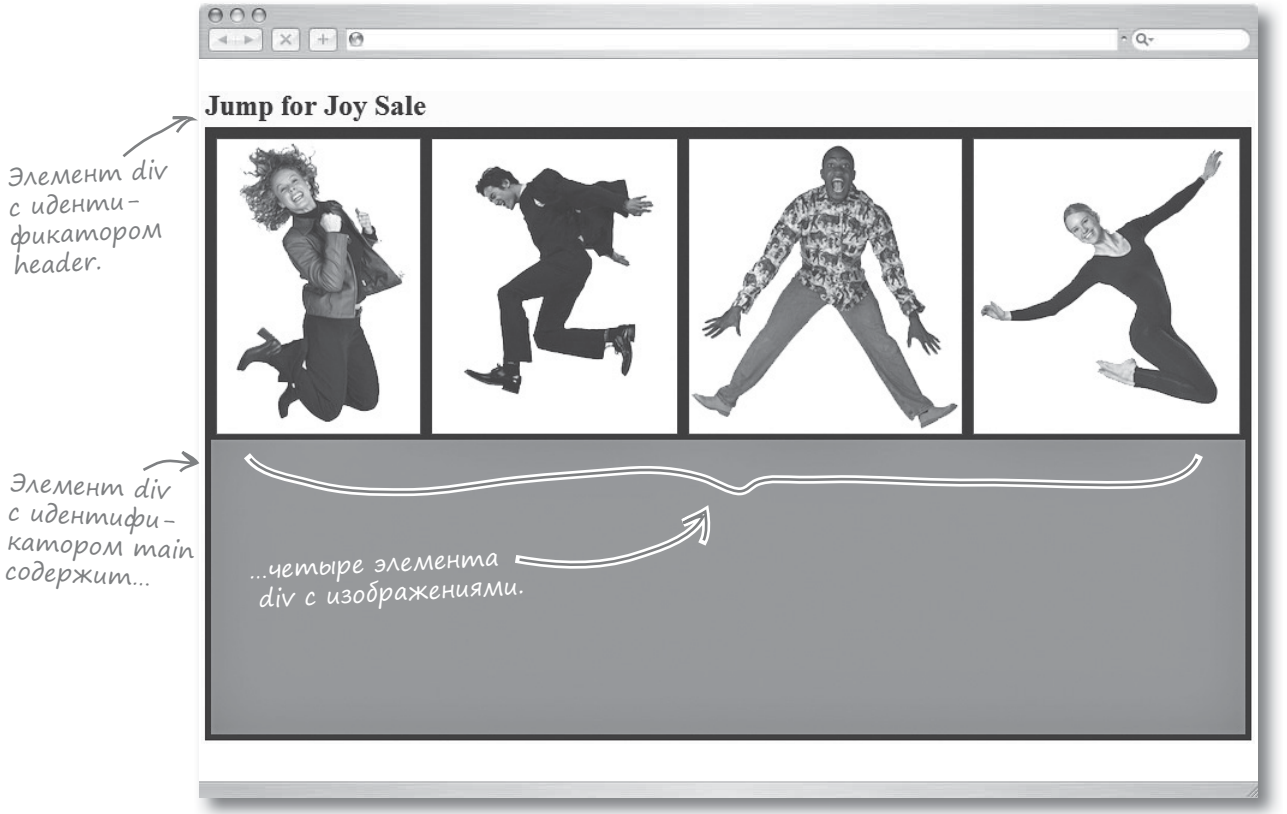
CSS

my_style.css

Элементы div для изображений.

ТЕСТ-ДРАЙВ

Откройте страницу в своем любимом браузере и убедитесь в том, что все работает правильно. С этим наглядным примером вам будет проще понять, как должна работать итоговая версия страницы.



МОЗГОВОЙ ШТУРМ

Страница содержит четыре области с изображениями. Как заставить эти области реагировать на щелчки мышью?

Событие click под увеличительным стеклом

Как мы уже знаете, организовать отслеживание события click элементом в jQuery совсем не сложно.

При щелчке на элементе страницы происходит *событие*, которое может привести к выполнению *функций*. Позднее события и функции будут рассмотрены более подробно, а пока давайте посмотрим, как событие **click** работает с тегами абзацев (или тегами `<div>`).

Элемент, с которым связывается событие click.

Здесь мы сообщаем интерпретатору JS, что элементы абзацев должны что-то делать, когда пользователь щелкает на них мышью.

Функция объединяет операции, которые должны выполняться при обнаружении события.

```
$("p").click( function() {
    alert("You rang?");
});
```

Так как функция заключена в круглые скобки, относящиеся к событию click, она будет выполняться при выполнении пользователем щелчка мышью.

Открывающая фигурная скобка начинает «блок» программного кода. Блок можно сравнить с абзацем текста: абзац состоит из строк текста, блок состоит из строк кода.

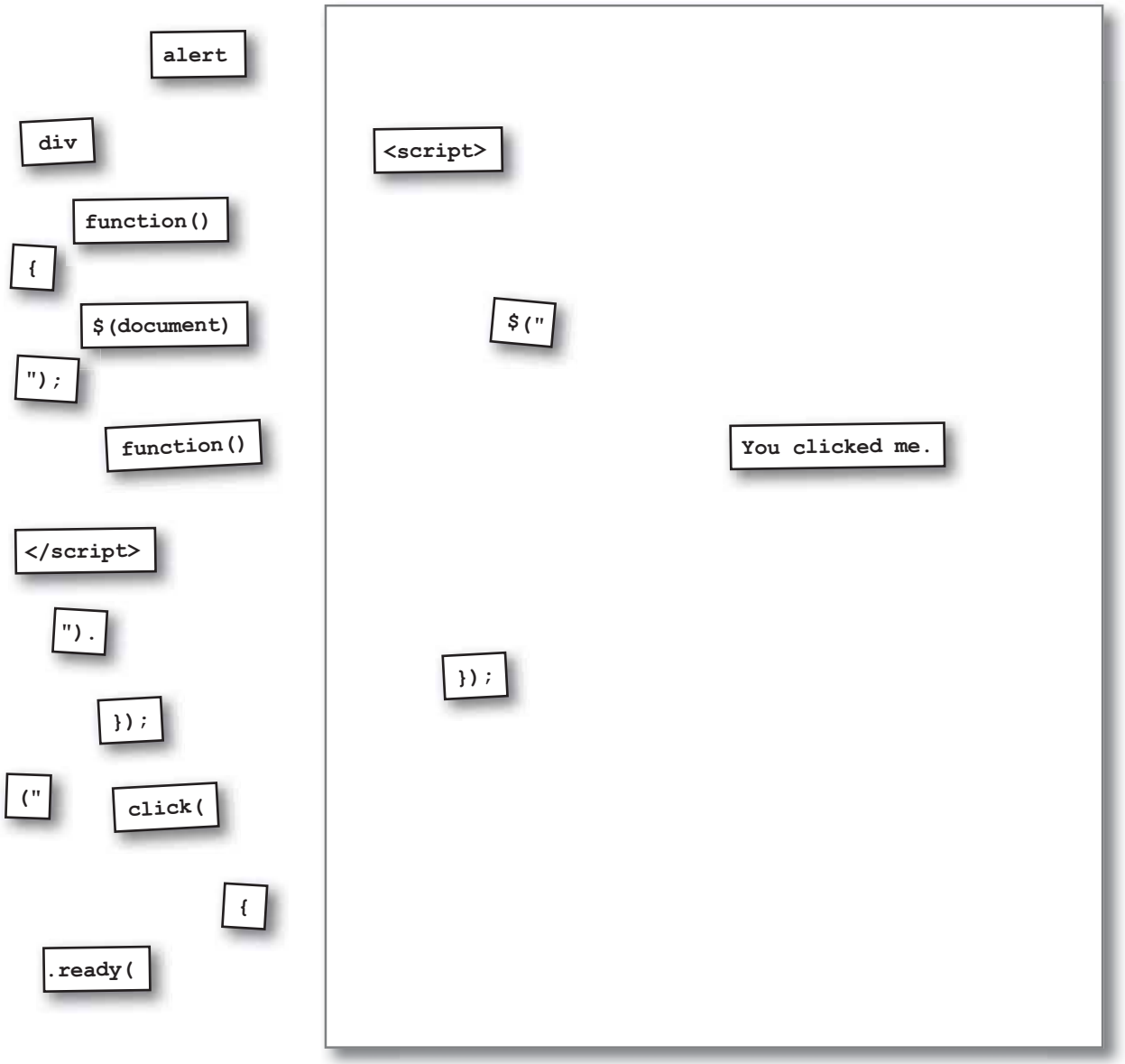
Команда alert поможет нам убедиться в том, что функция была вызвана правильно.

Текст, заключенный в кавычки, появится в новом окне.

Закрывающая фигурная скобка завершает «блок» кода.

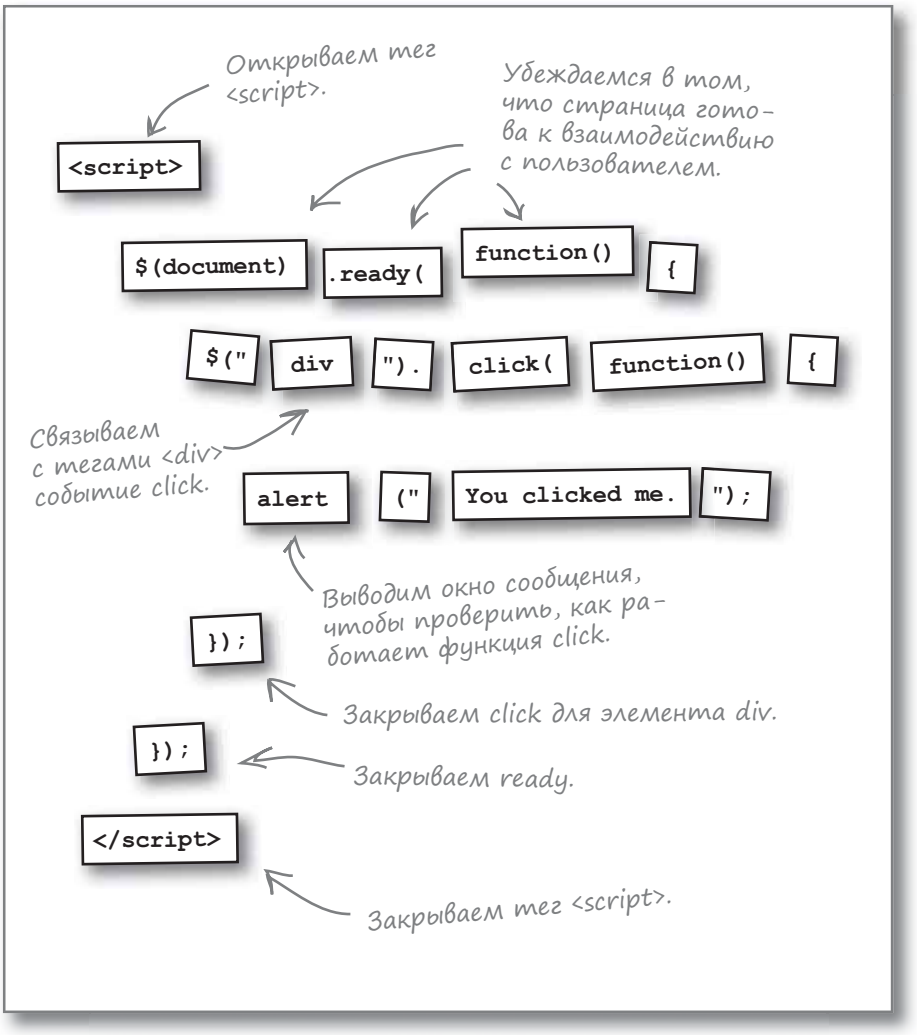
Развлечения с магнитами

Сложите из магнитов фрагмент кода, после выполнения которого элементы div станут реагировать на щелчки мышью. При щелчке на `div` функция JavaScript alert должна выводить текст «You clicked me». Мы уже расставили несколько магнитов по местам.



Развлечения с магнитами

Сложите из магнитов фрагмент кода, после выполнения которого элементы div станут реагировать на щелчки мышью. При щелчке на `div` функция JavaScript alert должна выводить текст «You clicked me».



Включение метода click в страницу

Используя код из решения с магнитами на предыдущей странице, включите этот сценарий в свой файл HTML. Не забудьте, что он должен быть заключен в тег <script>!


```
<html>
  <head>
    <title>Jump for Joy</title>
    <link href="styles/my_style.css" rel="stylesheet">
  </head>
  <body>
    <div id="header">
      <h2>Jump for Joy Sale</h2>
    </div>
    <div id="main">
      <div><img src="images/jump1.jpg"/></div>
      <div><img src="images/jump2.jpg"/></div>
      <div><img src="images/jump3.jpg"/></div>
      <div><img src="images/jump4.jpg"/></div>
    </div>
    <script src="scripts/jquery-1.6.2.min.js"></script>
    <script >
      $(document).ready(function() {
          $("div").click(function() {
            alert("You clicked me.");
          });//end click function
      });//end doc ready
    </script>
  </body>
</html>
```

Функция alert открывает в браузере окно с сообщением. Мы будем использовать ее для проверки результатов включения в код новых переменных, функций и т. д.

Добавьте эти строки между тегами <script>, чтобы области div реагировали на щелчки мышью.

Некоторые программисты добавляют комментарии, которые помогают определить, к какой команде относятся фигурные и круглые скобки. Впрочем, использование комментариев — вопрос стиля программирования, и выбор исключительно за вами.

index.html

ТЕСТ-ДРАЙВ

Откройте страницу в своем любимом браузере и убедитесь в том, что все правильно работает. Если щелкнуть на изображении, на странице появляется окно сообщения.

Окно сообщения, добавленное нами в программу. Как видите, функция click работает.

Да, но сообщение появляется всегда, где бы я ни щелкала. Почему так происходит?

Хммм, действительно.

Кажется, мы немного увлеклись со щелчками. Давайте снова вернемся к событию click.

Интерпретатор JS делает в точности то, что мы требуем. Он выбирает все области div...

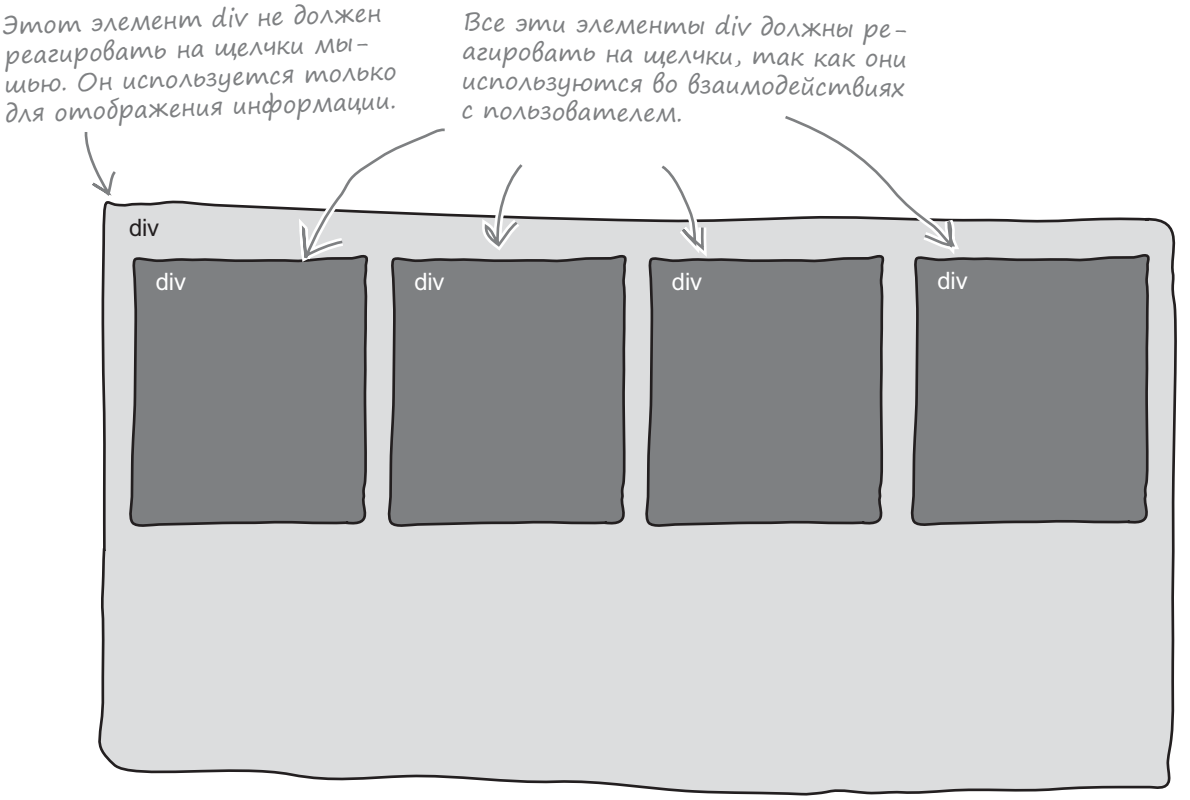
...и связывает каждую область с методом click.

```
$("div").click( );
```

Чтобы получить сообщение, необязательно даже щелкать на изображениях. В структуре нашей страницы используются элементы div, вложенные в другие элементы div; при щелчке на них браузер думает, что вы щелкнули на обоих элементах, и вы можете получить сразу два сообщения. Похоже, необходимо более конкретно объяснить jQuery, чего мы хотим…

Выражайтесь точнее

Проблема в том, что мы слишком расплывчато сформулировали правило выбора элементов. Так как же отобрать четыре вложенных области `div` без вмещающего элемента `div`? Вспомните, о чем говорилось в главе 1: селекторы jQuery используют классы и идентификаторы CSS. Вы всегда можете уточнить, с какими элементами должна выполняться операция, назначая этим элементам классы и идентификаторы.



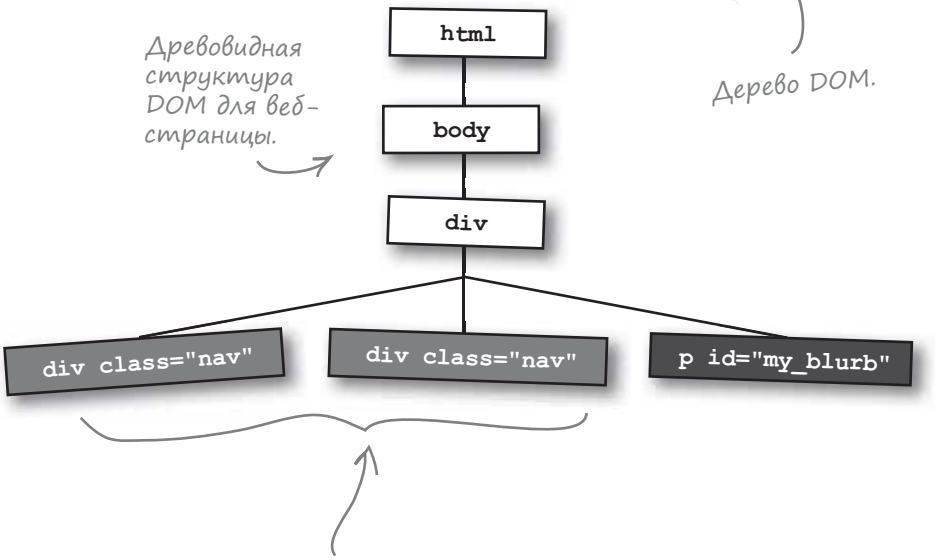
МОЗГОВОЙ ШТУРМ

Что лучше использовать для определения элементов div в нашем примере — только классы CSS, только идентификаторы или их сочетание? Какой способ лучше подойдет и почему?

Назначение классов

В CSS классы используются для группировки элементов и назначения им общих стилевых атрибутов. Страница может содержать один или несколько элементов, относящихся к одному классу. В jQuery можно использовать селектор классов, чтобы действие методов jQuery распространялось на определенную группу элементов. И в CSS, и в jQuery для обозначения классов используется символ «.», поэтому присвоить класс элементу проще простого.

Дерево DOM.




```
.nav {
      display: block;
      border: solid #00f 1px;
      width: 100%;
}
```

CSS

Код CSS

```
$(".nav").click( function(){
      alert("You clicked
me!");
});
```

jQuery

Код jQuery

Идентификаторы элементов

Селектор идентификаторов предназначен для выбора одного конкретного элемента страницы. В jQuery, как и в CSS, селекторы идентификаторов обозначаются символом #. Они особенно хорошо подходят для максимально точного определения элемента страницы, а также для определения элементов, заведомо существующих только в одном экземпляре (например, заголовка или завершителя страницы).

Селекторы идентификаторов определяют один конкретный элемент.

```
#my_blurb {
      display: block;
      border: 0px;
      height: 50%;
}
```

Код CSS

```
$("#my_blurb").slideToggle("slow");
```

Код jQuery

Кто и что делает?

Укажите, для каких целей могут использоваться классы и идентификаторы, сделав пометку в соответствующем столбце. Не забывайте, что класс и идентификатор иногда могут делать одно и то же!

	Класс	Идентификатор
Однозначно определяет уникальный элемент страницы	☐	☐
Определяет один или несколько элементов страницы	☐	☐
Может использоваться одним методом JavaScript для кросс-браузерной идентификации элемента	☐	☐
Может использоваться в CSS для применения стилей к элементам	☐	☐
Может многократно назначаться одному элементу	☐	☐

Кто и что делает? Решение

Укажите, для каких целей могут использоваться классы и идентификаторы, сделав пометку в соответствующем столбце. Не забывайте, что класс и идентификатор иногда могут делать одно и то же!

	Класс	Идентификатор
Однозначно определяет уникальный элемент страницы	☐	☑
Определяет один или несколько элементов страницы	☑	☐
Может использоваться одним методом JavaScript для кросс-браузерной идентификации элемента	☐	☑
Может использоваться в CSS для применения стилей к элементам	☑	☑
Может многократно назначаться одному элементу	☑	☐

Часто Задаваемые Вопросы

В: Что такое *блочный* элемент?

О: Блочные элементы отображаются внутри своих родительских элементов как прямоугольные объекты, не выходящие за их границы. Для них определяются свойства блочных полей, ширины и высоты, которые могут задаваться независимо от окружающих элементов.

В: Почему тег `<script>` размещается в конце страницы перед тегом `</body>`? Мне казалось, что он должен находиться в тегах `<head>` `</head>`?

О: Да, когда-то это считалось правильным (а кое-кто продолжает так считать). Однако у такого размещения сценариев есть один недостаток: оно блокирует параллельную загрузку в браузере. В общем случае изображения с разных серверов могут загружаться одновременно, но после того как ваш браузер встретит тег `<script>`, дальнейшая параллельная загрузка невозможна. Размещение сценариев в конце страницы сокращает время загрузки страницы.

В: Для чего нужны окна сообщений JavaScript?

О: Окна сообщений не блещут красотой, но приносят немалую пользу. По сути функция *alert* отображает простое окно с текстом, указанным в круглых скобках при ее вызове. Если вы хотите вывести строку текста, заключите ее в кавычки. Чтобы вывести значение переменной, укажите имя переменной без кавычек. Значения переменных также можно объединять со строками при помощи знака +. Наверняка вы уже встречались с окнами сообщений, даже если не обращали на них особого внимания — например, когда забывали заполнить обязательное поле на форме. В нашем случае окно сообщения используется в целях тестирования и отладки. Конечно, существуют и более совершенные средства вывода отладочной информации; они будут представлены в следующих главах книги.

Беседа у камина

Беседа у камина: селекторы CSS и jQuery обсуждают свои различия.

Селектор CSS:

Привет, Селектор jQuery. Хорошо, что ты здесь — так окружающие скорее поймут, что своим существованием ты обязан исключительно мне.

Да уж, стиля мне не занимать — но кто сказал, что я ничего не делаю? Я могу моментально изменить внешний вид многих объектов на странице.

И что же можешь сделать ты, чего не могу я?

Началось, технический жаргон... Что это значит «возвращаешь» элементы?

Но я тоже могу влиять на все выбранные элементы — скажем, назначить им всем розовый цвет фона. И не забывай, что вся твоя функциональность обеспечивается моими механизмами.

Да, пожалуй, это впечатляет.

Селектор jQuery:

А, да, спасибо... Конечно, моя сила во многом обусловлена твоими способностями к выбору элементов. Однако я ориентируюсь на поведение, а не на стилевое оформление. Твое дело — простое украшение страницы, а благодаря мне выполняется серьезная работа.

Не спорю, польза от тебя есть. У тебя своя работа — изменение внешнего вида элементов, а у меня своя — совершенно другая.

Я нахожу элементы и возвращаю их, чтобы метод мог выполнить какую-либо операцию с возвращаемым набором.

Допустим, кто-то использует меня для выбора всех элементов абзацев на странице. Я отбираю все абзацы и передаю их методу jQuery для дальнейшей обработки.

Верно, твой механизм селекторов лежит в основе части моих возможностей, но другую часть предоставляет JavaScript. Не забудь о слове «запрос» (Query) в моем имени. Я запрашиваю у браузера элемент и передаю его методу jQuery, а метод заставляет этот элемент летать по странице или даже раствориться в воздухе.

Да, и ты кое в чем прав — без тебя я бы этого сделать не смог.

Три уровня веб-страницы

Классы и идентификаторы закладывают основу для трех уровней веб-страницы, рассмотренных нами в главе 1: структуры, стилевого оформления и сценарного кода. Селекторы *связывают* эти уровни, позволяя им взаимодействовать друг с другом. HTML предоставляет «строительный материал» (элементы, атрибуты и т. д.), или **структуру** веб-страницы. CSS предоставляет **стилевое оформление**, то есть описание представления и размещения этих элементов. JavaScript и jQuery предоставляют **сценарный код**, управляющий поведением (функциональностью) этих элементов.

Допустим, элементу img, для которого должен выполняться метод `slideUp`, назначен класс `slideshow`:

Элементу img назначается класс slideshow.

```
<img class="slideshow" />
```

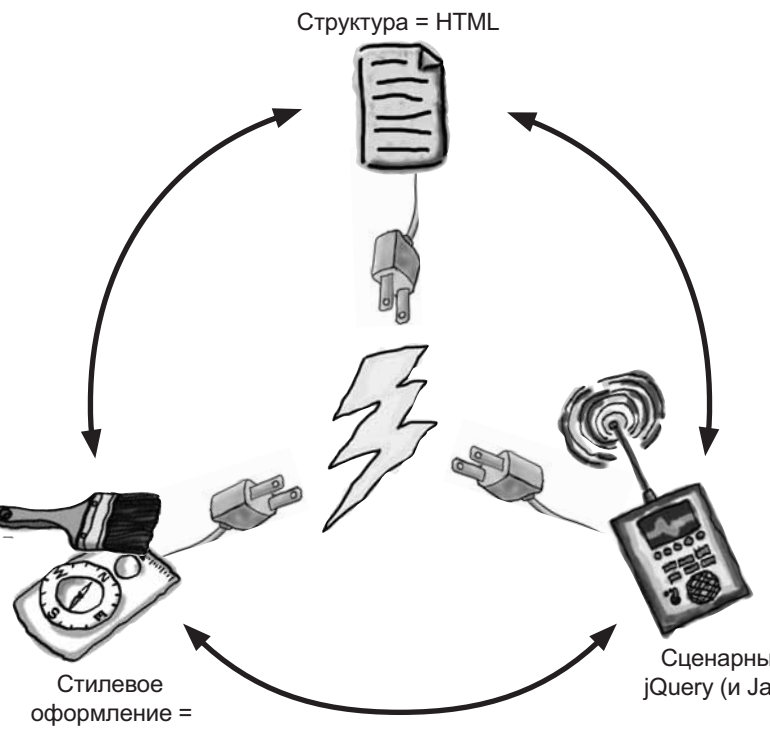


Таблица стилей задает параметры представления и позиционирования класса slideshow с использованием селектора.

```
.slideshow{
   float: left;
   height: 100;
}
```

```
$(".slideshow").slideUp()
```

Сценарная библиотека jQuery выбирает элемент и задает его поведение.

Возьми в руку карандаш

Измените структуру, стилевое оформление и сценарный код страницы, чтобы на щелчки реагировали только четыре области div. В файле CSS создайте класс CSS (с именем `guess_box`) и примените его в тегах `html` и `script`. Кажется, один из элементов `div` также потерял свой атрибут id. Удастся ли вам определить отсутствующее значение и вернуть его на место?

```
div{
   float:left;
   height:245px;
   text-align:left;
   border: solid #000 3px;
}
#header{
   width:100%;
   border: 0px;
   height:50px;
}
#main{
   background-color: grey;
   height: 500px;
}

.....
height 245px;
.....
.....
```

my_style.css

```
<html>
  <head>
    <title>Jump for Joy</title>
    <link href="styles/my_style.css" rel="stylesheet">
  </head>
  <body>
    <div id="header">
      <h2>Jump for Joy Sale</h2>
    </div>
    <div .....>
      <div ..... ><img src="images/jump1.jpg"/></div>
      <div ..... <img src="images/jump2.jpg"/></div>
      <div ..... ><img src="images/jump3.jpg"/></div>
      <div ..... ><img src="images/jump4.jpg"/></div>
    </div>
    <script src="scripts/jquery-1.6.2.min.js"></script>
    <script>
      $(document).ready(function() {
        $(".....").click(function() {
          alert("You clicked me.");
        });
      });
    </script>
  </body>
</html>
```

index.html

Возьми в руку карандаш
Решение

Всем элементам div, в которых будет скрываться код скидки, назначается класс `guess_box`. Также измените селектор, чтобы в нем использовался этот класс, и включите его в файл CSS. Атрибут id пропал у главного элемента div.

```
<html>
  <head>
    <title>Jump for Joy</title>
    <link href="styles/my_style.css" rel="stylesheet">
  </head>
  <body>
    <div id="header">
      <h2>Jump for Joy Sale</h2>
    </div>
    <div id="main" >
      <div class="guess_box" ><img src="images/jump1.jpg"/></div>
      <div class="guess_box" ><img src="images/jump2.jpg"/></div>
      <div class="guess_box" ><img src="images/jump3.jpg"/></div>
      <div class="guess_box" ><img src="images/jump4.jpg"/></div>
    </div>
    <script src="scripts/jquery-1.6.2.min.js"></script>
    <script>
      $(document).ready(function() {
        $(".guess_box").click(function() {
          alert("You clicked me.");
        });
      });
    </script>
  </body>
</html>
```

index.html

Метод click связывается только с классом guess_box, а не со всеми элементами div.

```
div{
   float:left;
   height:245px;
   text-align:left;
   border: solid #000 3px;
}
#header{
   width:100%;
   border: 0px;
   height:50px;
}
#main{
   background-color: grey;
   height: 500px;
}
.guess_box{
   height 245px;
}
```

my_style.css

Здесь определяется класс для четырех областей div. Значение height соответствует высоте используемых изображений, так что вся графика сохраняет нормальное выравнивание.

Возвращаемся к списку

Вернемся к списку задач и посмотрим, что же мы выполнили из пожеланий Эмили.

- [x] Страница должна состоять из четырех областей. Каждая область содержит одно изображение.
- [x] Области должны реагировать на щелчки мышью.
- [ ] Нам понадобится сообщение, состоящее из текста («Размер вашей скидки:») и случайного значения (от 5 до 10 процентов).
- [ ] Когда пользователь щелкает в одной из областей, под изображением в этой области должно появляться сообщение.
- [ ] Если пользователь щелкает снова, то старое сообщение исчезает и появляется новое.

В общем-то ничего.

Однако вывод сообщений для пользователя связан с некоторыми нюансами. Не забывайте, что для разных посетителей сайта могут выводиться разные сообщения.

МОЗГОВОЙ ШТУРМ

Необходимо создать сообщение и где-то сохранить его для вывода. Как вы думаете, как это лучше сделать?

Выделение памяти для хранения данных

Следующее требование в нашем списке — вывод текста, который остается неизменным при каждом выполнении сценария. Но помимо этого, необходимо хранить число, которое генерируется в зависимости от случайной величины. С другой стороны, на протяжении работы сценария это число должно каким-то образом сохраняться. Для хранения такой информации при работе с jQuery используются **переменные** JavaScript.

Ключевое слово var объявляет переменную.

За ключевым словом var указывается имя переменной.

Так значение переменной задается в программном коде.

```
var pts = 250;
```

При объявлении переменной интерпретатор JavaScript выделяет блок памяти браузера, в котором будут храниться ваши данные.

Переменной присваивается имя, по которому вы будете позднее обращаться к ней в своем сценарии.

Значение, сохраняемое в переменной, указывается после знака равенства.

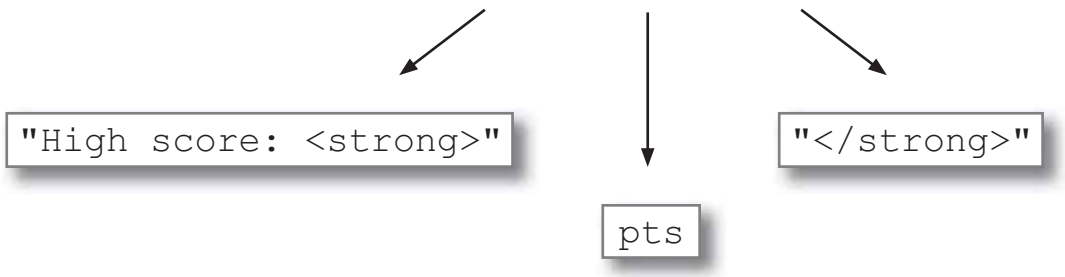
Теперь, чтобы получить доступ к сохраненным данным, достаточно указать имя переменной.

Если вы захотите больше узнать о переменных JavaScript и математических функциях, найдите книгу М. Моррисона «Изучаем JavaScript» (Питер, 2012)!

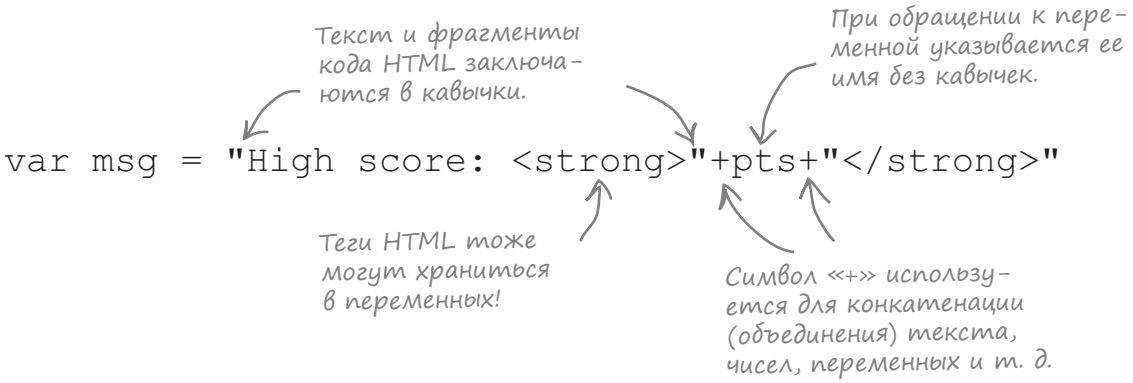
Конкатенация и слияние данных

В разных сценариях jQuery нам придется сохранять разные типы данных: числа, текст, логические признаки «истина/ложь». Достаточно часто (особенно когда потребуется выводить разные сообщения для пользователей) разметка HTML будет смешиваться с другими данными. Как же переменные объединяются с другими значениями? Посредством ***конкатенации***. Допустим, в вашей видеоигре переменная с именем pts используется для хранения рекорда, и вы хотите вывести ее значение.

Нужно объединить три фрагмента:



Получается следующее:



Упражнение

Следующий фрагмент кода JavaScript создает переменную с именем `discount`, в которой хранится случайное число от 5 до 10. Напишите код создания переменной `discount_msg`, содержащей текст сообщения и случайное число. Позаботьтесь о том, чтобы сообщение содержалось в элементе абзаца.

```
var discount = Math.floor((Math.random()*5) + 5);
```

.....

Упражнение
Решение

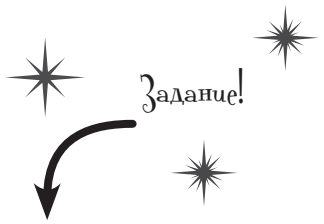
Следующий фрагмент кода JavaScript создает переменную с именем `discount`, в которой хранится случайное число от 5 до 10. Напишите код создания переменной `discount_msg`, содержащей текст сообщения и случайное число. Позаботьтесь о том, чтобы сообщение содержалось в элементе абзаца.

Математические операции и функция random рассматриваются в главе 3.

```
var discount = Math.floor((Math.random()*5) + 5);
var discount_msg = "<p>Your Discount is "+ discount +"%</p>";
```

Возвращаемся к программному коду...

Итак, теперь у нас есть переменная, в которой хранится объединенное сообщение о скидке. Остается лишь обновить фрагмент, заключенный в теги <script>.




```
<script>
  $(document).ready(function() {

    $(".guess_box").click( function() {

      var discount = Math.floor((Math.random()*5) + 5);
      var discount_msg = "<p>Your Discount is "+ discount +"%</p>";
      alert(discount);

    });
  });
</script>
```

Создаем новые переменные JavaScript.

Мы передаем переменную discount функции alert, чтобы убедиться в том, что она содержит нужный текст.

index.html

Вставка сообщения

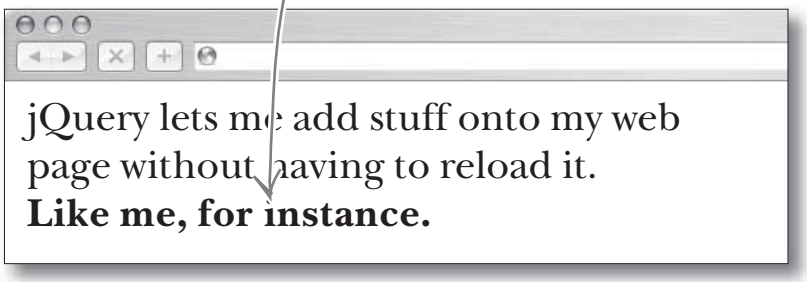
Сообщение готово, но как вывести его на странице под изображением, на котором был сделан щелчок? По сути, речь идет о *вставке* нового содержимого в страницу. В jQuery предусмотрено несколько способов вставки содержимого в существующие элементы. Самые полезные способы будут подробно рассмотрены в главе 4, а пока мы воспользуемся действием `append`.

```
<p>jQuery lets me add stuff onto my web page
without having to reload it.</p>
```

Эта команда jQuery приказывает интерпретатору JS присоединить содержимое, заданное в круглых скобках, ко всем элементам абзацев.

```
$("p").append(" <strong>Like me, for instance.</strong>");
```

Если выполнить эту команду в секции сценарного кода, то на странице появится текст, выделенный жирным шрифтом.



Итоговый код HTML в модели DOM.

```
<p>jQuery lets me add stuff onto my web page
without having to reload it.</p> <strong>Like me,
for instance.</strong>
```

Упражнение

Используя то, что вы уже знаете о селекторах, а также только что описанный метод `append`, напишите код присоединения переменной `discount` к элементу `guess_box`.

.....

Упражнение
Решение

Как видите, включить новое сообщение в веб-страницу совсем несложно!

```
$(".guess_box").append(discount_msg);
```

append — метод jQuery. Методы используются в jQuery для выполнения различных операций.

Часто Задаваемые Вопросы

В: Существуют ли ограничения для имен классов?

О: Имя класса должно начинаться с символа подчеркивания (_), дефиса (-) или буквы латинского алфавита (a–z), за которыми может следовать любое количество дефисов, символов подчеркивания, букв или цифр. Правда, есть еще правило: если первым символом является дефис, то вторым символом должна быть буква или символ подчеркивания, а имя должно содержать не менее двух символов.

В: Существуют ли ограничения для имен переменных?

О: Да! Имена переменных не могут начинаться с цифр. Кроме того, они не могут содержать знаки математических операторов (+ * - ^ / ! \), пробелы или знаки препинания. При этом допускается использование символов подчеркивания. Имена переменных не могут совпадать с ключевыми словами JavaScript (например, `window`, `open`, `array`, `string`, `location`), и в них учитывается регистр символов.

В: Сколько классов можно назначить элементу?

О: Согласно стандартам, четко определенного максимума не существует, но в условиях реального использования ограничение составляет около 2000 классов на элемент.

В: Можно ли выбрать все элементы страницы?

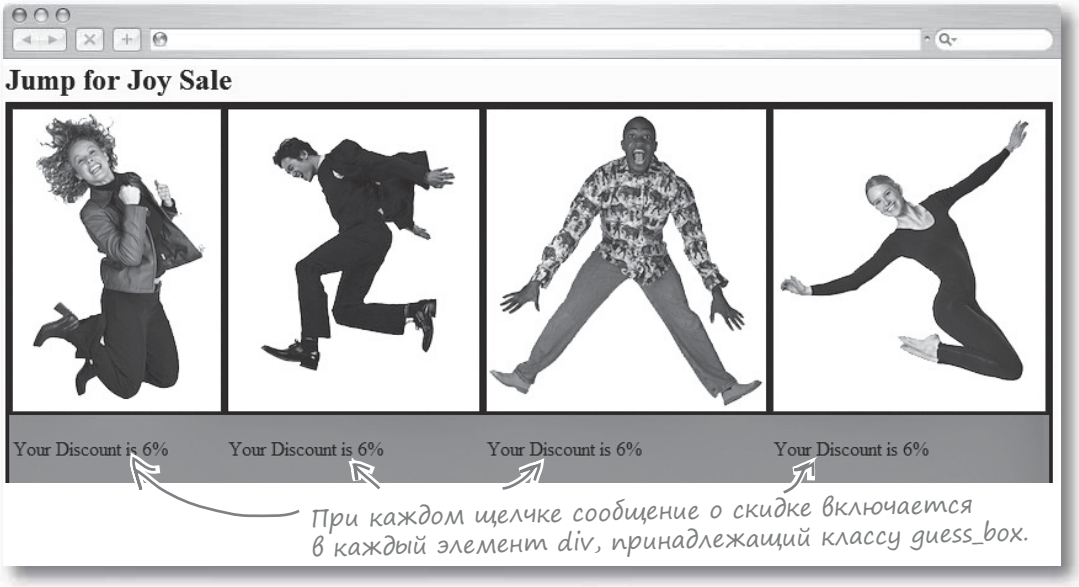
О: Да! Для получения всех элементов достаточно передать jQuery селектор "*".

В: Если я присваиваю элементам класс или идентификатор, не определяя для него стилевого оформления, повлияет ли это на их внешний вид в браузере?

О: Нет, в браузерах не существует оформления по умолчанию для классов или идентификаторов. Некоторые браузеры используют несколько иную процедуру обработки таких элементов, но наличие класса или идентификатора, к которому не применено оформление CSS, не отразится на внешнем виде элементов.

ТЕСТ-ДРАЙВ

Откройте страницу в своем любимом браузере и убедитесь в том, что все работает. Обратите особое внимание на вызов *alert* – убедитесь в том, что значение переменной `discount` успешно сохраняется.



Все отлично работает, но...

Переменная `discount` генерирует случайное число, а сообщение присоединяется к странице так, как и ожидалось, но мы сталкиваемся с неожиданным побочным эффектом: информация о скидке выводится во всех областях div. Это не совсем то, чего мы добивались. Что же пошло не так?

```
<script>
  $(document).ready(function() {
    $(".guess_box").click( function() {
      var discount = Math.floor((Math.random()*5) + 5);
      var discount_msg = "<p>Your Discount is "+discount+"%</p>";
      alert(discount_msg);
      $(".guess_box").append(discount_msg);
    });
  });
</script>
```

Здесь элемент связывается с методом click, чтобы все элементы класса guess_box реагировали на щелчки мышью.

Просто для проверки значения переменной.

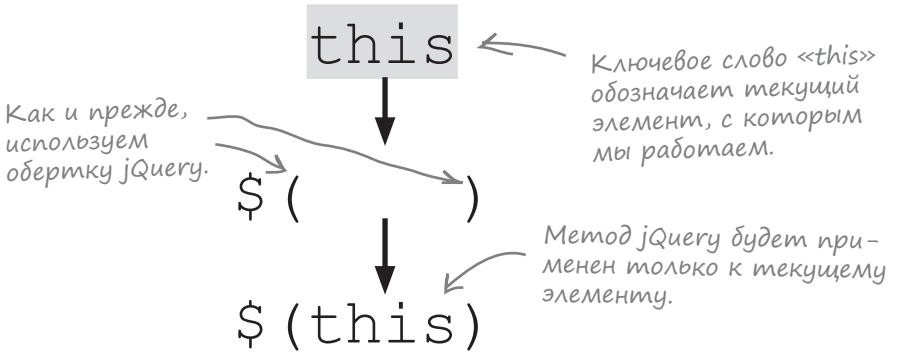
Селектор работает на уровне класса, так что операция распространяется на все элементы div, принадлежащие классу.

Сообщение о скидке должно выводиться только в той отдельной области div, на которой щелкает пользователь. Как же выбрать **только** ту область, в которой был сделан щелчок, и присоединить содержимое `discount` только к этой области?

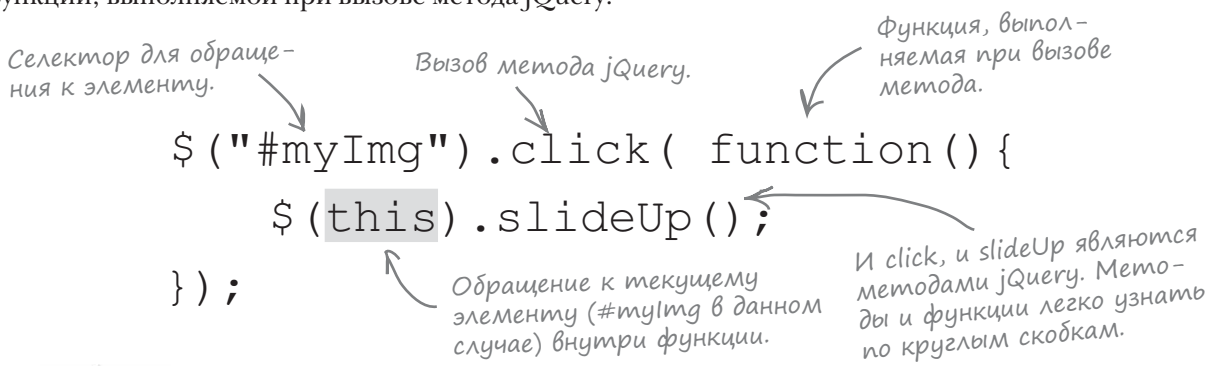

А как было бы замечательно, если бы в jQuery можно было легко определить элемент div, на котором щелкнул пользователь? Но это, конечно, всего лишь мечты...

Дайте мне $(this)

В этой главе рассматривались селекторы jQuery и их использование для выбора элементов, с которыми работают методы jQuery. Довольно часто мы хотим предельно конкретно указать, какой элемент требуется выбрать. В подобных ситуациях часто используется простейший селектор `$(this)`, обозначающий *текущий* элемент.



Помните, что смысл конструкции `$(this)` зависит от контекста. Иначе говоря, смысл `$(this)` изменяется в зависимости от того, где и как вы используете эту конструкцию. В частности, ее уместно использовать внутри функции, выполняемой при вызове метода jQuery:



Для любознательных

this и $(this)

В JavaScript ключевое слово «this» обозначает элемент DOM, с которым мы работаем в своем коде. Добавление `$( )` (в результате чего получается `$(this)`) позволяет нам взаимодействовать с элементом DOM с использованием методов jQuery.

Использование $(this)

Давайте посмотрим, как `$(this)` поможет справиться с возникшей проблемой. Измените программный код так, чтобы в нем использовалась конструкция `$(this)`; ниже соответствующий фрагмент выделен жирным шрифтом.

```
<script type="text/javascript">
  $(document).ready(function() {

    $(".guess_box").click( function() {

      var discount = Math.floor((Math.random()*5) + 5);
      var discount_msg = "<p>Your Discount is "+ discount +"%</p>";
      alert(discount_msg);
      $(this).append(discount_msg);

    });
  });//end doc ready
</script>
```

index.html

ТЕСТ-ДРАЙВ

Откройте страницу в своем любимом браузере и убедитесь в том, что все работает. Обратите особое внимание на вызов *alert* – убедитесь в том, что значение переменной `discount` успешно сохраняется. Щелкните несколько раз в разных местах; проверьте правильность вывода случайного числа, присоединенного к текстовому сообщению.

$(this) прекрасно работает! Но теперь при повторных щелчках на странице появляются лишние сообщения. Можно ли от них избавиться?

Хороший вопрос!

Мы подошли к последнему пункту нашего списка задач.

- [x] Страница должна состоять из четырех областей. Каждая область содержит одно изображение.
- [x] Области должны реагировать на щелчки мышью.
- [x] Нам понадобится сообщение, состоящее из текста («Размер вашей скидки:») и случайного значения (от 5 до 10 процентов).
- [x] Когда пользователь щелкает в одной из областей, под изображением в этой области должно появляться сообщение.
- [ ] Если пользователь щелкает снова, то старое сообщение исчезает и появляется новое.

МОЗГОВОЙ ШТУРМ

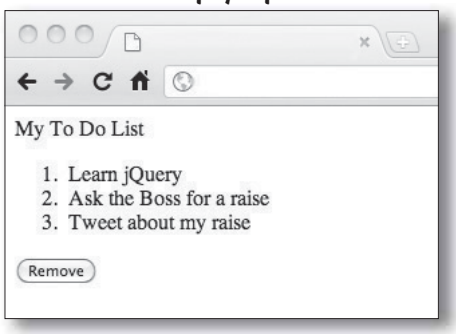
Как вы думаете, как удалить со страницы последнее сообщение?

Скатертью дорога! Метод remove

Как удалить последнее сообщение и создать новое? Используйте метод `remove`. Этот метод предназначен для удаления со страницы отдельного элемента или группы элементов. Взгляните на очень простую страницу со списком и кнопкой.

1. Слева показано, как страница выглядит в браузере, а справа приведена разметка HTML для ее создания.

В браузере



В разметке HTML

```
<div>My To Do List</div>
<ol>
    <li>Learn jQuery</li>
    <li>Ask the Boss for a raise</li>
    <li>Tweet about my raise</li>
</ol>
<button id="btnRemove">
```

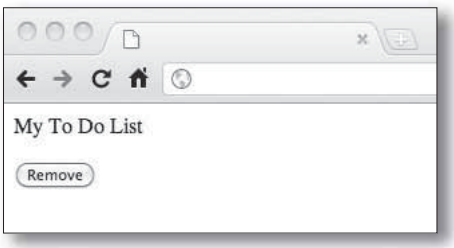
2. А это код кнопки, удаляющей все пункты из списка:

```
$("#btnRemove").click(function(){
    $("li").remove();
});
```

remove — еще один метод jQuery. Методы jQuery определяют действия, выполняемые с веб-страницей.

3. Посмотрите на страницу в браузере и на HTML после выполнения кода jQuery: все пункты списка исчезли, их нет даже в HTML!

В браузере



В разметке HTML

```
<div>My To Do List</div>
<ol>
</ol>
<button id="btnRemove">
```

МОЗГОВОЙ ШТУРМ

Как может выглядеть селектор, который удалит со страницы только сообщение о скидке?

Селекторы потомков

Селекторы потомков — еще одна разновидность селекторов, которая может использоваться в jQuery. Оказывается, они идеально подходят для нашей ситуации. Селекторы потомков позволяют определять *отношения между элементами*: вы можете выбирать родительские, дочерние или сестринские (одноранговые) элементы.

Слева указывается селектор родителя.

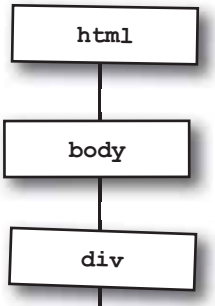
Справа указывается селектор потомка.

```
$("div p")
```

Имена родителя и потомка разделяются пробелом.

А так выбирается «потомок потомка».

```
$("div div p")
```



Селектор возвращает все элементы div, родителем которых является элемент div.

```
$("div div")
```

Объединение селекторов классов и идентификаторов с селекторами потомков позволяет очень точно определить выбираемые элементы, что особенно важно при работе со сложными веб-страницами.

```
$("div p#my_blurb")
```

Выбор всех элементов img, которые являются потомками потомков элементов div.

```
$("div div img")
```

(«Внучатые» элементы?)

```
$("div p")
```

Выбор всех элементов p, родителями которых являются элементы div.

Возьми в руку карандаш

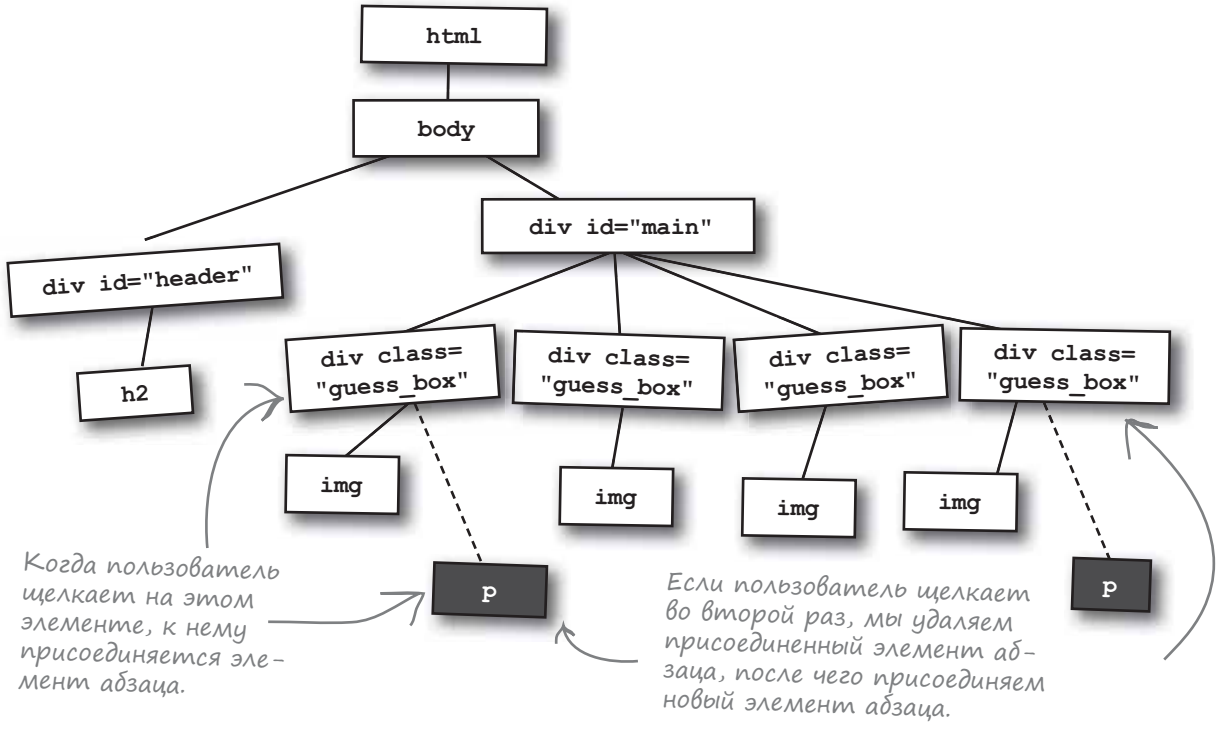
Напишите команду, которая будет удалять последнее сообщение в модели DOM из нашего примера с использованием селектора потомков.

.....

Возьми в руку карандаш
Решение

Начните с селектора классов `.guess_box`, за ним укажите селектор потомков `p` для обращения к добавленному элементу абзаца. Затем вызовите метод `remove`, чтобы удалить из страницы все элементы `p`, родитель которых принадлежит к классу `guess_box`.

```
$(".guess_box p").remove();
```



Итак, теперь я могу присоединять и удалять фрагменты страницы по своему усмотрению. Важно ли, где и когда это происходит?

Да, порядок добавления и удаления элементов важен.

Ведь вы не сможете удалить элемент до того, как он будет добавлен, — и было бы бессмысленно удалять элемент сразу же после его добавления, верно?

Возьми в руку карандаш

Решите, где должна находиться команда `remove`. Запишите ее в одной из строк 1, 2 или 3 и объясните, почему разместили ее именно в этой строке. Подумайте, *когда* должен удаляться абзац, и методом исключения выберите правильное место для выполнения этой операции.

```
<script>
  $(document).ready(function() {
    1. .....
    $(".guess_box").click( function() {
    2. .....
      var discount = Math.floor((Math.random()*5) + 5);
      var discount_msg = "<p>Your Discount is "+ discount +"%</p>";
      alert(discount_msg);
      $(this).append(discount_msg);
    3. .....
    });
  });
</script>
```

index.html

Почему я считаю, что команда должна находиться именно здесь:

.....

.....

.....

.....

.....

Возьми в руку карандаш
Решение

Решите, где должна находиться команда `remove`. Запишите ее в одной из строк 1, 2 или 3 и объясните, почему разместили ее именно в этой строке. Подумайте, *когда* должен удаляться абзац, и методом исключения выберите правильное место для выполнения этой операции.

```
<script>
  $(document).ready(function() {
    1. .....
    $(".guess_box").click( function() {
    2.     $(".guess_box p").remove(); .....
      var discount = Math.floor((Math.random()*5) + 5);
      var discount_msg = "<p>Your Discount is "+ discount +"%</p>";
      alert(discount_msg);
      $(this).append(discount_msg);
    3. .....
    });
  });
</script>
```

index.html

Почему я считаю, что команда должна находиться именно здесь:

Команда remove не может находиться в строке 1, потому что эта строка находится за пределами функции click класса guess_box. Она также не может находиться в строке 3, потому что в этом случае она будет удалять только что присоединенный фрагмент. Последнее сообщение о скидке должно удаляться перед генерированием нового, поэтому мы размещаем remove в первой строке блока кода (в фигурных скобках) функции click класса guess_box.

Будьте осторожны!

Очень важно, чтобы методы jQuery вызывались в правильном порядке и в правильный момент.

Причем это особенно важно тогда, когда вы выводите важную информацию для своих посетителей, а потом удаляете ее снова. Мы еще вернемся к теме последовательности и своевременности применения эффектов в главе 5.

Часто Задаваемые Вопросы

В: Иногда после вызова метода `remove` элементы, удаленные из разметки, остаются на странице. Почему это происходит?

О: Часто при использовании команды получения исходного кода страницы браузер обращается к серверу с новым вызовом. Инспекторы DOM (например, Chrome Developer Tools или Firebug для Firefox) покажут вам содержимое DOM для текущего состояния страницы.

В: Что делают методы `Math.floor` и `Math.random`?

О: Метод `floor` округляет число вниз до ближайшего целого и возвращает результат. Метод `random` возвращает случайное число от 0 до 1. При умножении его на другое число мы получаем случайное число в интервале от 0 до числа-множителя.

В: Откуда взялось ключевое слово `this`?

О: Во многих объектно-ориентированных языках ключевым словом `this` (или `self`) в методах экземпляров обозначается объект, для которого был вызван метод, выполняемый в настоящий момент.

В: Значит, я могу выводить сообщение о скидке тогда, когда пользователь щелкает на одном из изображений, и удалять ее при щелчке на другом изображении. Но ведь для реального приложения этого было бы недостаточно, верно?

О: Вы абсолютно правы. Это всего лишь первая часть головоломки. Этот код должен передать информацию о предоставленной скидке в тот момент, когда пользователь переходит к оформлению заказа. Чтобы скидка была учтена при расчете, ее необходимо передать серверу для обработки. Такая функциональность более подробно рассматривается в главах 8–10.

ТЕСТ-ДРАЙВ

Включите только что написанную строку кода в файл *index.html*. Затем откройте страницу в своем любимом браузере и убедитесь в том, что все работает. Несколько раз щелкните на изображении и проверьте, что сообщение о скидке успешно выводится, а старое сообщение удаляется перед присоединением нового кода.

Ваша очередь прыгать от радости

Поздравляем! Вы справились со всеми требованиями, и рекламная акция теперь заработает.

- [x] Страница должна состоять из четырех областей. Каждая область содержит одно изображение.
- [x] Области должны реагировать на щелчки мышью.
- [x] Нам понадобится сообщение, состоящее из текста («Размер вашей скидки:») и случайного значения (от 5 до 10 процентов).
- [x] Когда пользователь щелкает в одной из областей, под изображением в этой области должно появляться сообщение.
- [x] Если пользователь щелкает снова, то старое сообщение исчезает и появляется новое.

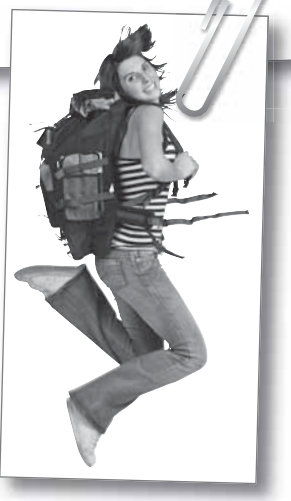
От: **Эмили**
Тема: **Re: Рекламная акция «Прыгаем от радости!»**

Спасибо за то, что ты сделал для меня! Мой сайт стал гораздо лучше.

Надеюсь, посетители будут в таком же восторге от моих фотографий, как я — от новой страницы!

--
Эмили

> PS. Прилагаю свой автопортрет — так я выглядела, когда увидела новую веб-страницу... Надеюсь, не нужно объяснять, что я делаю?



ГЛАВА 2

Ваш инструментарий jQuery

Глава 2 осталась позади, а ваш творческий инструментарий дополнился основными принципами работы с селекторами и некоторыми методами jQuery.

$(this)

Выбирает «текущий» элемент.

Смысл $(this) изменяется в зависимости от того, где находится эта конструкция.

Селекторы

$(this) — выбирает текущий элемент.

$("div") — выбирает все элементы div на странице.

$("div p") — выбирает все элементы p, вложенные непосредственно в элементы div.

$(".my_class") — выбирает все элементы класса my_class.

$("div.my_class") — выбирает только те элементы, которым назначен класс my_class. (Разные типы элементов могут относиться к одному классу.)

$("#my_id") — выбирает элемент с идентификатором my_id.

Методы jQuery

Метод jQuery представляет собой фрагмент кода, определенный в библиотеке jQuery и предназначенный для многократного использования. Методы используются для выполнения различных операций в jQuery и JavaScript. Считайте, что метод — своего рода команда, выполняющая некоторые операции с веб-страницей.

.append — вставляет заданное содержимое в DOM. Данные присоединяются к элементу, для которого вызывается метод.

.remove — удаляет элемент из DOM.

Страница в центре событий

jQuery позволяет легко включить в любую веб-страницу поддержку действий и интерактивности. В этой главе вы узнаете, как заставить вашу страницу реагировать на действия, выполняемые пользователем. Возможность выполнения кода в ответ на действия пользователя поднимает сайт на совершенно новый уровень. Также в этой главе рассказано, как создавать функции, чтобы однократно написанный код можно было использовать много раз.

Ни минуты покоя

Эмили довольна вашей работой для рекламной акции «Прыгаем от радости», но после встречи с бухгалтером она хочет внести на сайте некоторые изменения.

От: **Эмили**
Тема: **RE: Прыгаем от радости**

Привет,

Ты отлично справился с поддержкой нашей рекламной акции! Я встречалась с нашим бухгалтером, и мы обсудили некоторые показатели успеха нашей рекламной акции.

Бухгалтер предложил внести в приложение некоторые изменения, которые должны повысить уровень продаж.

Посетители по-прежнему получают четыре изображения для выбора скидки. Однако на этот раз величина скидки остается постоянной. Бухгалтер рекомендует установить величину в 20% от объема заказа, такая высокая скидка будет для них более привлекательной.

Посетитель получает только одну возможность найти код скидки, который при каждом посещении скрывается под случайно выбранным изображением. Если посетитель находит код скидки, щелкнув на изображении, то код появляется на экране. В противном случае приложение показывает, под каким изображением была спрятана скидка.

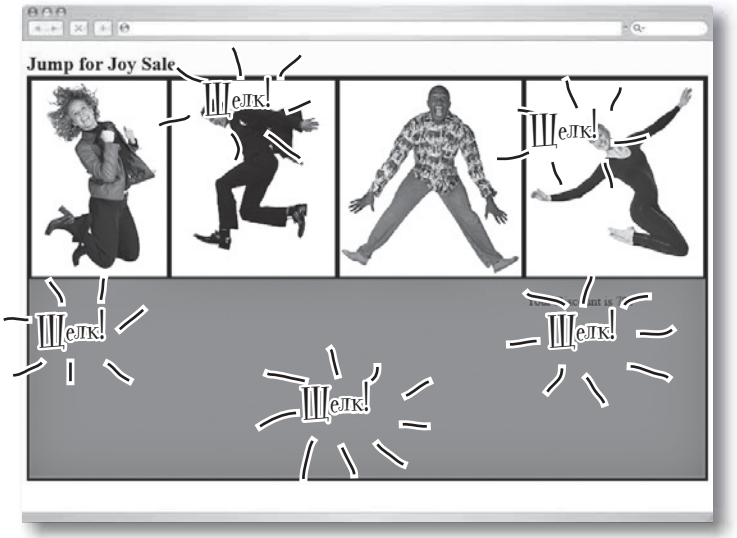
Удастся ли тебе справиться с этой задачей так же хорошо, как и с первой?

--
Эмили Сондерс
jumpforjoyphotos.hg

Эмили приложила фотографии своего бухгалтера — тот явно не собирается прыгать от радости. Может, после внесения изменений на сайте дело пойдет на лад?

В словах бухгалтера есть резон...

Если скидка будет скрываться только под одним изображением, Эмили не придется возиться с лишними кодами, а посетители все равно будут взаимодействовать с сайтом. Похоже, все эти новые возможности нужны *только* для одного: заставить посетителей побольше щелкать...



Возьми в руку карандаш

Пришло время создавать новый список требований. Вы уже знаете, что делать: просмотрите сообщение Эмили и выберите все новые возможности приложения, которые она хочет видеть на своем сайте. Опишите простыми словами, как должно работать новое решение.

Требования:

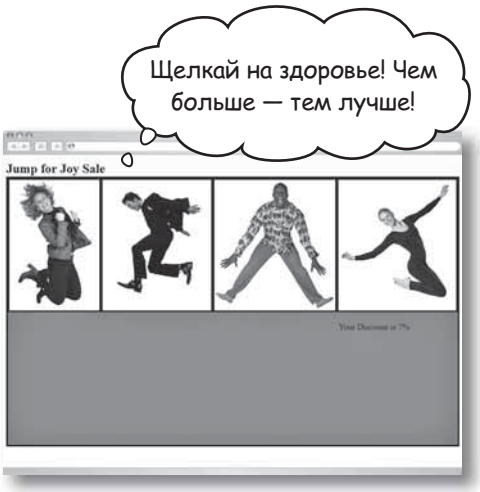
Возьми в руку карандаш
Решение

Пришло время создавать новый список требований. Вы уже знаете, что делать: просмотрите сообщение Эмили и выберите все новые возможности приложения, которые она хочет видеть на своем сайте. Опишите простыми словами, как должно работать новое решение.

Требования:

- *Скидка должна скрываться только в одном из четырех изображений, и при каждой загрузке страницы изображения должны располагаться в случайном порядке.*
- *После загрузки страницы посетителю предоставляется только одна возможность найти скидку. Следовательно, если первая попытка оказалась неудачной, то дальнейшие щелчки необходимо заблокировать.*
- *После того как посетитель выбрал изображение и щелкнул на нем, следует сообщить о результате. Если изображение было выбрано правильно, нужно вывести величину скидки.*
- *Вместо случайной скидки будет использоваться постоянная скидка 20 %. Таким образом, вместо процента в сообщении следует вывести код скидки.*

Как наше решение работает сейчас



Как оно должно работать

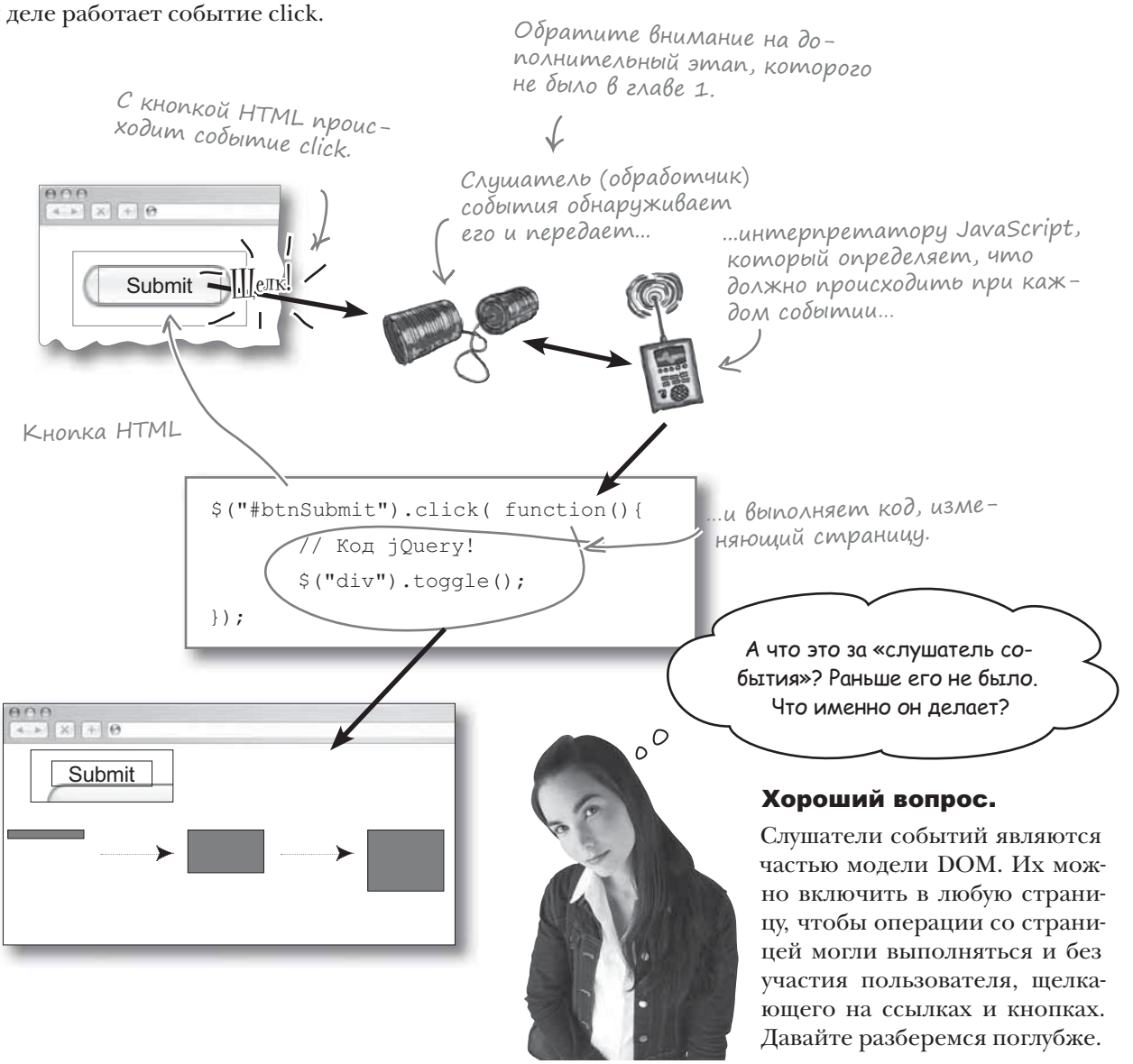
МОЗГОВОЙ ШТУРМ

Вы знаете, как включить в страницу функцию click. Но как убедиться в том, что посетитель использует ее только один раз?

В предыдущих главах вы узнали, как организовать выполнение кода при вызове click. Нельзя ли использовать эту информацию в реализации нового решения?

Реакция на события

Работа приложения «Прыгаем от радости» основана на щелчках мышью. В jQuery и JavaScript щелчок называется *событием* (существует много других событий, но нас пока интересуют только щелчки). Механизм событий позволяет вам выполнять код, когда со страницей что-то происходит (например, пользователь щелкает на кнопке). Выполняемый код оформлен в виде *функции* – а как говорилось ранее, функции делают код jQuery более эффективным и пригодным для повторного использования. Вскоре функции будут рассмотрены более подробно, а пока давайте посмотрим, как же на самом деле работает событие click.



Хороший вопрос.

Слушатели событий являются частью модели DOM. Их можно включить в любую страницу, чтобы операции со страницей могли выполняться и без участия пользователя, щелкающего на ссылках и кнопках. Давайте разберемся поглубже.

За кулисами слушателя событий

При помощи слушателей событий браузер получает информацию о том, что пользователь сделал на странице, и указывает интерпретатору JavaScript, должен ли тот что-то предпринять по этому поводу.

jQuery предоставляет очень простые механизмы связывания слушателей событий с любым *элементом страницы*, так что пользователям не приходится долго кликать по ссылкам и кнопкам!

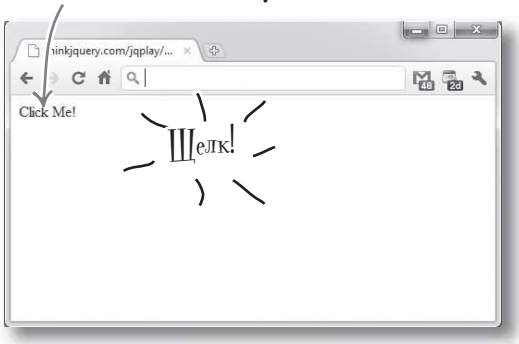
Добавляется событие click.

Элемент с идентификатором showMessage.

```
$("#showMessage").click(function() {
        alert('You Clicked Me!');
});
```

Выполняется этот код.

1 **Пользователь щелкнул на элементе.**



4 **Пользователь видит, как на странице что-то происходит.**

Что вы видите в браузере

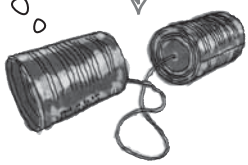
За сценой

3 **Интерпретатор JavaScript выполняет функцию, заданную в слушателе события.**

2 **Слушатель события, связанный с элементом, «слышит» событие click и сообщает интерпретатору JavaScript, какой код связан с этим событием.**

Я должен вывести окно сообщения!

Я слышу щелчок.



Слушатель события

Интерпретатор JavaScript

Связывание события

Добавление события к элементу называется *связыванием* события с элементом. При этом слушатель события узнает, что он должен сообщить интерпретатору JavaScript, какую функцию тот должен вызвать.

Есть два способа связывания событий с элементами.

Способ 1

Этот способ используется для связывания событий с элементами при загрузке страницы. Такая форма записи часто называется «сокращенной».

```
$("#myElement").click( function() {
      alert($(this).text());
});
```

Способ 2

Этот способ работает так же, как и предыдущий, но он также может использоваться для добавления событий к элементам, включенным в страницу после ее загрузки (например, в результате создания новых элементов DOM).

Оба способа добавляют слушателя события click к элементу с идентификатором myElement.

```
$("#myElement").bind('click', function() {
      alert($(this).text());
});
```

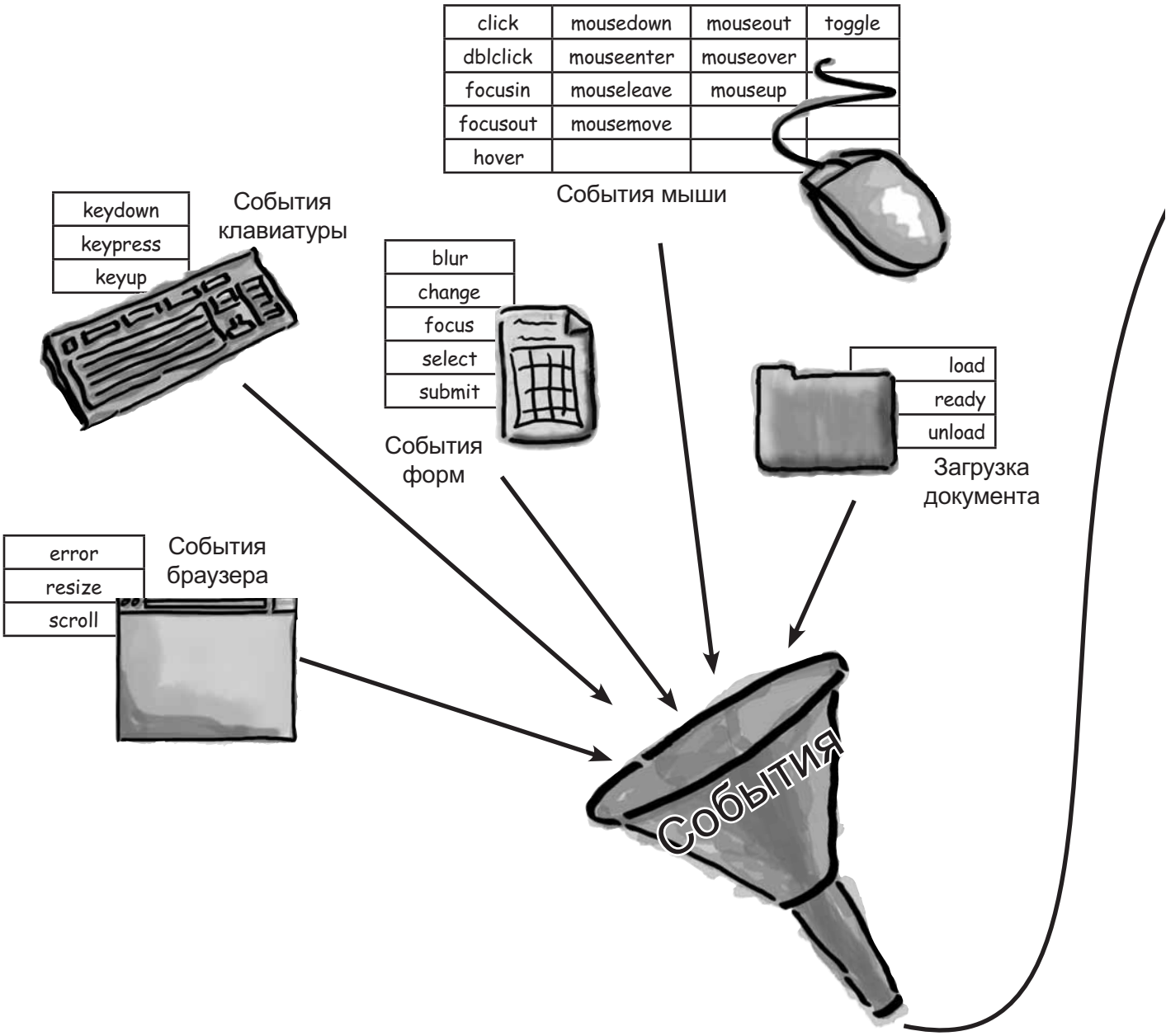
Будьте осторожны!

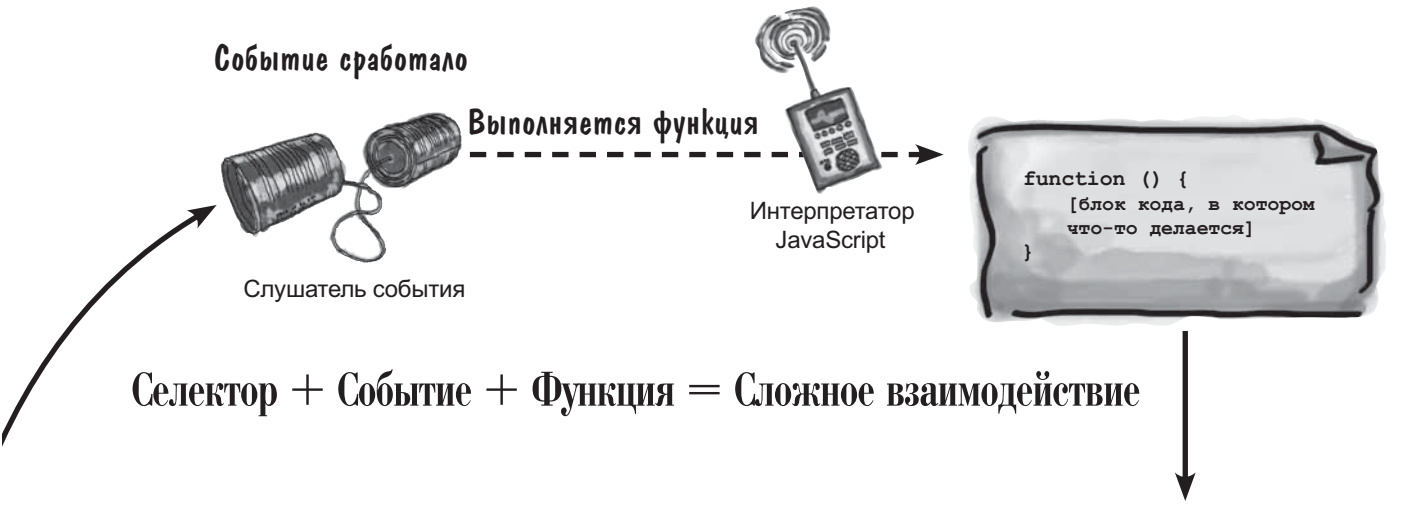
Способ 1 — всего лишь сокращенная форма способа 2, но он может использоваться только в том случае, если элементы DOM уже существуют.

В jQuery много таких сокращенных вариантов записи, которые упрощают программный код. Они существуют исключительно для удобства — однако у них есть свои ограничения. Способ 2 следует использовать для добавления событий к новым элементам DOM, созданным в вашем коде (например, когда на странице создается новое изображение, реагирующее на щелчки, или в список включается новая строка, с которой должен взаимодействовать пользователь).

Срабатывание событий

События могут срабатывать при выполнении разнообразных операций со страницей. Более того, операции, выполняемые с самим браузером, тоже могут стать источниками событий!





Селектор + Событие + Функция = Сложное взаимодействие

Часто Задаваемые Вопросы

В: А что это за функции в событиях?

О: Они называются *обработчиками*. Функция-обработчик выполняется при каждом срабатывании события. Функции будут более подробно рассмотрены позднее в этой главе.

В: Где можно узнать о разных видах событий?

О: На сайте jquery.com, в разделе «Documentation → Events».

В: Сколько существует разных категорий событий?

О: В jQuery события делятся на пять категорий: события браузера, события загрузки документа, события форм, события клавиатуры и события мыши.

В: А сколько вообще существует разных событий?

О: Около 30 разных типов во всех категориях.

В: Что может привести к срабатыванию события на странице?

О: Большинство типов событий срабатывает от операций с устройствами ввода (клавиатура, мышь). Тем не менее браузер, документ страницы, код jQuery и даже форма HTML на странице тоже могут быть источниками событий.

ОТКРОВЕННО О СОБЫТИЯХ

Интервью недели: Чем так замечательны события?

HeadFirst: Добрый вечер, Событие, мы очень рады встрече.

Событие: Да, и я тоже.

HeadFirst: Так кто же вы? Что собой представляет настоящее Событие?

Событие: Я — объект, помогающий людям взаимодействовать с веб-страницей. В наши дни я уже твердо стою на ногах, но до появления jQuery мое существование было... как бы это выразиться... немного рассеянным.

HeadFirst: Как интересно. Мы еще вернемся к этой теме, но почему вы говорите о рассеянности? Откуда вы вообще появились?

Событие: Это долгая история. В середине 1990-х годов фирма Netscape выпустила Navigator 2.0. Какие были времена... Моя модель тогда была очень простой. DOM, JavaScript и я — все мы только-только появились. Существовал стандарт W3C, в котором говорилось, как браузеры должны реализовать нас!

HeadFirst: Но это было давно. С тех пор вы сильно изменились.

Событие: Да, все мы изменились. Попутно мы даже ввязались в «войну браузеров» между Microsoft Internet Explorer и Netscape. Обе компании пытались обойти друг друга при помощи всяких классных штучек, которые не входили в стандарт, но поддерживались только в браузере этой компании. В конечном итоге победа досталась Microsoft.

HeadFirst: Похоже, непростое было время.

Событие: Да, но сейчас дела налаживаются. В 1997 году Netscape и Microsoft выпустили версии 4.0 своих браузеров. В них было много новых событий, а наши возможности по работе со страницей заметно расширились. События тогда пользовались уважением.

HeadFirst: И что произошло?

Событие: Ситуация вышла из-под контроля. Браузер Netscape был преобразован в проект с открытым кодом и позднее превратился в Mozilla Firefox. Но какое-то время браузер Netscape продолжал существовать, причем в нем и в Internet Explorer использовались разные модели событий. Многие решения работали только в одном браузере, а это сильно раздражало пользователей, заходивших на сайт в другом браузере. В конечном итоге Netscape не стало, но на сцене появилось несколько новых браузеров.

HeadFirst: Тогда почему вы говорите, что дела налаживаются?

Событие: Полной совместимости между браузерами по-прежнему нет. Набор событий, поддерживаемых Internet Explorer, отличается от наборов событий Firefox, Google Chrome, Apple Safari и Opera. Однако ситуация меняется к лучшему с выходом каждой новой версии. Браузеры постепенно начинают лучше соответствовать стандартам. Но самое замечательное, что jQuery решает эти проблемы за веб-разработчика.

HeadFirst: Правда? Вот здорово! А как? И что это за объект, которым, как вы говорите, вы являетесь?

Событие: jQuery выбирает способ обработки событий в зависимости от того, какой браузер используется посетителем ваших

сайтов. А что касается объекта, то здесь нет ничего сложного. В сущности, объект — это всего лишь переменные и функции, объединенные в одну структуру.

HeadFirst: И где можно побольше узнать об этих переменных и функциях?

Событие: Вы найдете информацию обо мне в официальной документации jQuery: *http://api.jquery.com/category/events/event-object/*.

HeadFirst: Спасибо! Непременно почитаю. А что нужно сделать, чтобы использовать вас на странице?

Событие: Для начала меня нужно с чем-то связать — ведь слушатель событий должен знать, что на меня следует обращать внимание. Затем какое-то условие должно приводить к моему срабатыванию, и тогда при возникновении события будет выполняться нужный код.

HeadFirst: Хорошо, но как узнать, какой именно код должен выполняться?

Событие: Этот код определяется при моем связывании с элементом. При этом можно задать практически любой набор операций — поэтому события и считаются такими полезными. Кроме того, меня можно отсоединить от элемента. Когда это происходит, слушатель перестает обнаруживать события для этого элемента, и тот код, который должен выполняться при моем срабатывании, выполняться уже не будет.

HeadFirst: Беседа получилась очень интересной, но наше время на исходе. Где я могу подробнее узнать о вас и о типах событий, которые могут происходить со страницами?

Событие: Уже упоминавшаяся ссылка объясняет, как работает объект. Дополнительную информацию обо мне и о разных типах событий можно найти в разделе документации на сайте jQuery. Спасибо за приглашение!

HeadFirst: И вам спасибо, что пришли. До встреч в программном коде!

Функции и переменные более подробно рассматриваются в этой главе.

Кроме того, в следующих главах будут описаны другие аспекты программирования на JavaScript.

Удаление событий

Наряду со связыванием событий с элементами также часто возникает необходимость удаления событий из элементов — например, когда вы не хотите, чтобы пользователи могли повторно щелкнуть на форме или же какая-либо операция со страницей должна выполняться только один раз. Именно такая ситуация встретилась нам в новых требованиях рекламной акции «Прыгаем от радости».

После связывания события с элементом мы можем удалить это событие из элементов, так чтобы оно больше не вызывалось.

Для удаления одного события:

Команда `unbind` приказывает браузеру прекратить прослушивать заданное событие конкретного элемента.

К элементу с идентификатором myElement добавляется слушатель события click.

Код, который выполняется при щелчке на myElement:

```
$("#myElement").bind ('click', function() {
    alert($(this).text());
});

$("#myElement").unbind('click');
```

Событие click отсоединяется от myElement.

Для удаления всех событий:

К элементу с идентификатором myElement добавляется слушатель события focus.

```
$("#myElement").bind ('focus', function() {
    alert("I've got focus");
});

$("#myElement").click(function(){
    alert('You clicked me.');
});

$("#myElement").unbind();
```

К элементу с идентификатором myElement добавляется слушатель события click.

Браузер перестанет прослушивать события элемента myElement.

Итак, слушатель события работает в браузере, присоединяется к элементам, ждет возникновения событий и приказывает интерпретатору JavaScript что-то сделать при обнаружении события, верно?

Да! Именно так.

Посмотрим, как события помогут нам в реализации первого требования.

- После загрузки страницы посетителю предоставляется только одна возможность найти скидку. Следовательно, если первая попытка оказалась неудачной, то дальнейшие щелчки необходимо заблокировать.

Не разрешайте пользователю повторный поиск скидки!

Возьми в руку карандаш

Используя то, что вы уже знаете о `$(this)`, и то, что вы узнали о событиях, измените код из предыдущей главы — включите в него команду удаления события click из областей `div`.

```
$(".guess_box").click( function() {
  $(".guess_box p").remove();
  var my_num = Math.floor((Math.random()*5) + 5);
  var discount = "<p>Your Discount is "+my_num+"%</p>";
  $(this).append(discount);

.....

.....

.....
});
```

index.html

Возьми в руку карандаш
Решение

Используя то, что вы уже знаете о `$(this)`, и то, что вы узнали о событиях, измените код из предыдущей главы — включите в него команду удаления события click из областей `div`.

```
$(".guess_box").click( function() {
  $(".guess_box p").remove();
  var my_num = Math.floor((Math.random()*5) + 5);
  var discount = "<p>Your Discount is "+my_num+"%</p>";
  $(this).append(discount);

  $(this).unbind("click");

});
```

Приказываем браузеру остановить прослушивание событий из текущего элемента.

index.html

ТЕСТ-ДРАЙВ

Измените файл *index.html* и сохраните его. Пощелкайте на странице и убедитесь в том, что она работает именно так, как нужно.

Jump for Joy Sale

Your Discount is 9%

А вы пробовали пощелкать на странице во время пробного запуска? И что произошло?

Наверное, то же, что и у меня. А это совсем не то, о чем говорится в требованиях...

Верно, слушатель события click удаляется не из всех областей.

Событие click удаляется только из той области div, на которой был сделан щелчок. На других областях по-прежнему можно щелкать. Как бы удалить слушателя события click из всех остальных элементов...

МОЗГОВОЙ ШТУРМ

Как бы вы удалили событие click **из всех областей** после того, как посетитель щелкнет на **одной области**? Нужно ли для этого перебирать все элементы один за другим?

Перебор элементов

Нередко бывает нужно выполнить некоторую операцию *с каждым элементом, входящим в группу.*

К счастью, jQuery предоставляет возможность *циклического перебора* группы элементов, определяемой заданным селектором. Проще говоря, вы последовательно, один за одним, перебираете входящие в группу элементы и выполняете некоторую операцию с текущим элементом.

Перебор всех элементов, соответствующих заданному селектору.

Выполняемая функция-обработчик

Селектор jQuery.

```
$(".nav_item").each(function(){

    $(this).hide();

});
```

Код выполняется для каждого элемента, соответствующего селектору.

Итератор .each берет группу элементов и последовательно выполняет операцию с каждым элементом группы.

Перебор будет более подробно рассмотрен позднее в книге.

Часто Задаваемые Вопросы

В: Могу ли я инициировать события в своем коде?

О: Да! Этот прием используется относительно часто — например, при отправке форм для проверки данных или закрытии модальных временных окон.

В: И как это делается?

О: События инициируются методом `.trigger` в сочетании с селектором — например, `$("button:first").trigger('click');` или `$("form").trigger('submit');`.

В: Могу ли я работать с событиями в веб-странице без использования jQuery?

О: Да, можете. jQuery всего лишь упрощает связывание событий с элементами, потому что библиотека обеспечивает межбраузерную совместимость и использует для связывания событий с элементами простые, легко запоминающиеся функции.

В: Как работает `.each`?

О: `.each` использует селектор, от которого исходит вызов, и создает массив элементов, соответствующих данному селектору. Затем в цикле происходит последовательный перебор элементов массива. Не беспокойтесь, вскоре массивы и циклы будут рассмотрены более подробно!

В: Итак, я могу создавать элементы средствами jQuery после загрузки страницы. Могут ли эти элементы получать события?

О: Да, могут. После того как элемент будет создан, вы можете воспользоваться методом `.bind` для связывания его со слушателем событий. Кроме того, если вам заранее известно, что ваш элемент будет вести себя как другие, уже созданные элементы, вы можете воспользоваться методом `.live`. Это приведет к тому, что обработчик события будет связываться со всеми элементами, соответствующими текущему селектору, сейчас и в будущем. Это решение подходит даже для элементов, которые еще не были включены в DOM.

Возьми в руку карандаш

Напишите код, использующий перебор для удаления слушателя события **click** из всех областей div, реагирующих на щелчки в странице «Прыгаем от радости». Внимательно прочитайте свой код и проверьте — может, какие-то из его фрагментов стали лишними?

```
$(".guess_box").click( function() {
   $(".guess_box p").remove();
   var my_num = Math.floor((Math.random()*5) + 5);
   var discount = "<p>Your Discount is "+my_num+"%</p>";
   $(this).append(discount);


   $(this).unbind('click');

});
```

index.html

Возьми в руку карандаш
Решение

Вызов метода `.each` для класса `.guess_box` перебирает все элементы этого класса, чтобы вы могли последовательно отсоединить метод click от каждого элемента. Команда с вызовом `.remove` стала лишней; так как посетители всего равно смогут щелкнуть всего один раз, удалять старое сообщение не нужно.

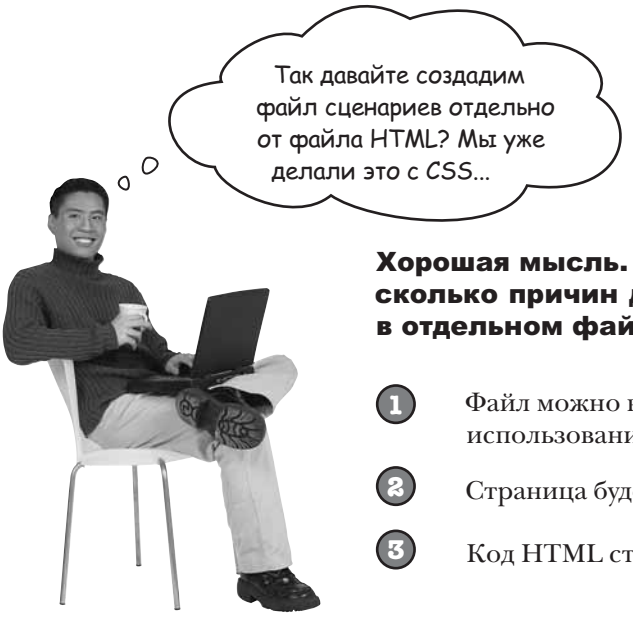
Так как пользователям разрешается щелкать только один раз, удалять старое сообщение уже необязательно!

```
$(".guess_box").click( function(){
  $(".guess_box p").remove();
    var my_num = Math.floor((Math.random()*5) + 5);
    var discount = "<p>Your Discount is "+my_num+"%</p>"
    $(this).append(discount);

    $(".guess_box").each( function(){
      $(this).unbind('click');
    });
});
```

Перебираем элементы .guess_box и удаляем из них событие click.

index.html

Стоп!
Вы не находите, что в файле HTML стало слишком много сценарного кода? Прежде чем двигаться дальше, хорошо бы придать ему стройности...



Хорошая мысль. Вообще-то существует несколько причин для хранения кода jQuery в отдельном файле.

1. Файл можно включать в разные страницы (повторное использование кода).
2. Страница будет загружаться быстрее.
3. Код HTML станет более стройным и удобочитаемым.

Упражнение

Вы уже знаете, как включить в HTML файлы CSS и библиотеку jQuery. Включение отдельного файла, содержащего ваш сценарный код JavaScript/jQuery, происходит практически так же!

В корневом веб-каталоге уже должна существовать папка *scripts* (где вы разместили библиотеку jQuery).

1. **В своем любимом текстовом редакторе создайте файл с именем *my_scripts.js* и сохраните его в папке *scripts*.**
2. **Перенесите весь код JavaScript и jQuery из файла *index.html* в новый файл. Переносить теги `<script>` и `</script>` необязательно.**
3. **Создайте ссылку на файл в странице HTML. Для этого перед закрывающим тегом `</body>` размещается следующая строка:**

```
<script src="scripts/my_scripts.js"></script>
```

Упражнение
Решение

Вы уже знаете, как включить в HTML файлы CSS и библиотеку jQuery. Включение отдельного файла, содержащего ваш сценарный код JavaScript/jQuery, происходит практически так же!

1. **В своем любимом текстовом редакторе создайте файл с именем *my_scripts.js* и сохраните его в папке *scripts*.**

2. **Перенесите весь код JavaScript и jQuery из файла *index.html* в новый файл. Переносить теги `<script>` и `</script>` необязательно.**

```
$(document).ready(function() {
  $(".guess_box").click( function() {
    var my_num = Math.floor((Math.random()*5) + 5);
    var discount = "<p>Your Discount is "+my_num+"%</p>";
    $(this).append(discount);

    $(".guess_box").each( function(){
      $(this).unbind('click');
    });
  });
});
```

my_scripts.js

Теперь этот файл можно включать во все файлы HTML, входящие в проект, — и все они смогут пользоваться вашим кодом jQuery. Вам уже не придется повторять этот код в каждом файле.

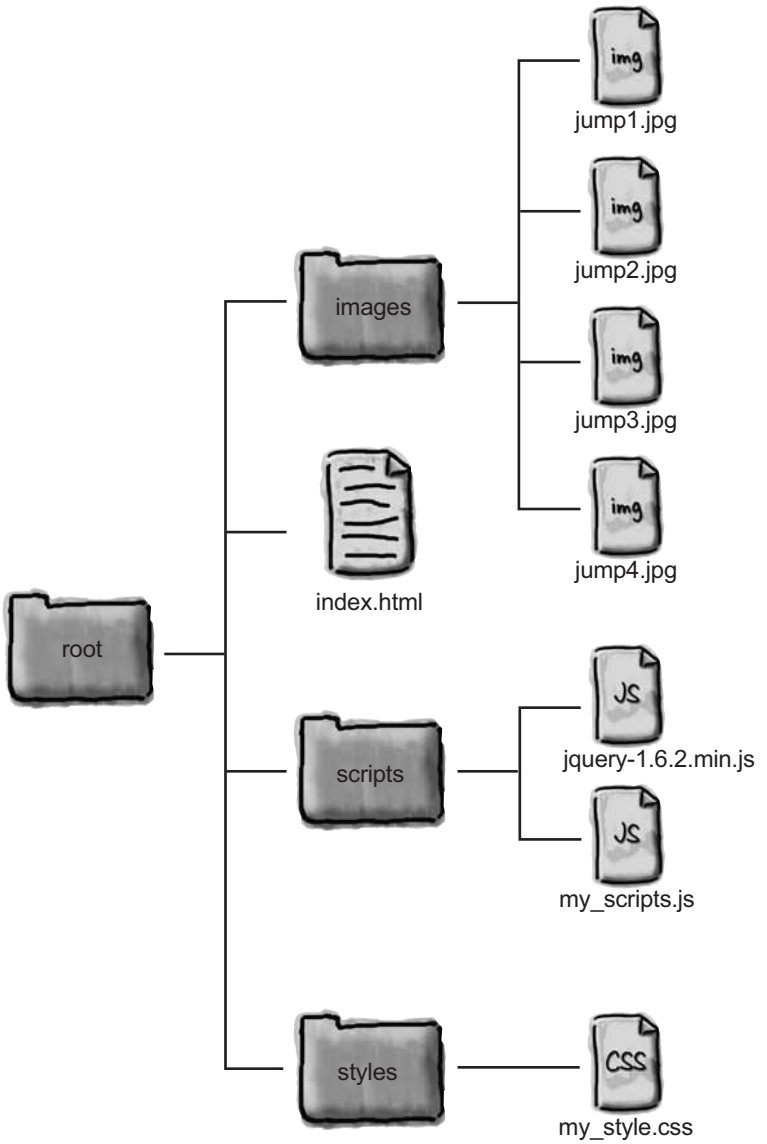
3 **Создайте ссылку на файл в странице HTML. Для этого перед закрывающим тегом** `</body>` **размещается следующая строка:**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Jump for Joy</title>
    <link href="styles/styles.css" rel="stylesheet">
  </head>
<body>
  <div id="header">
    <h2>Jump for Joy Sale</h2>
  </div>
  <div id="main ">
    <div class="guess_box"><img src="images/jump1.jpg"/></div>
    <div class="guess_box"><img src="images/jump2.jpg"/></div>
    <div class="guess_box"><img src="images/jump3.jpg"/></div>
    <div class="guess_box"><img src="images/jump4.jpg"/></div>
  </div>
  <script src="scripts/jquery.1.6.2.min.js"></script>
  <script src="scripts/my_scripts.js"></script>
</body>
<html>
```

index.html

Структура проекта

Мы только что внесли важные изменения в структуру файлов проекта. Давайте посмотрим, что же получилось в итоге.



Часто Задаваемые Вопросы

В: Почему наш файл имеет расширение *.js*?

О: Потому что библиотека jQuery написана на JavaScript и весь написанный для нее код должен включаться по правилам кода JavaScript.

В: Как отделение сценарного кода ускоряет загрузку страницы?

О: Если ваш файл *.js* включен в несколько файлов HTML, то браузер запросит его только один раз. Полученный файл сохраняется в кэше браузера, чтобы ему не приходилось заново обращаться к серверу для каждой страницы HTML, содержащей ссылку на ваш файл.

В: Почему теги `<script>` и `</script>` необязательны в файле *my_scripts.js*?

О: Потому что это теги HTML. Так как наш код уже включается в страницу как файл JavaScript, браузер уже знает тип его содержимого.

Развлечения с магнитами

Разложите магниты в соответствии со структурой файлов проекта. Убедитесь в том, что вы знаете, как правильно организовать разделение кода HTML, CSS и jQuery. В будущем вы должны безошибочно справляться с этой задачей.

```
<!DOCTYPE html>
<html>

.....
   <title>Jump for Joy</title>
    <link href="styles/styles.css" rel="stylesheet">
  </head>

.....
  <div id="header">
    <h2>Jump for Joy Sale</h2>

.....
  <div  id="main">
    <div .....><img src="images/jump1.jpg"/></div>
    <div class="guess_box"><img src="images/jump2.jpg"/></div>
    <div class="guess_box"><img src="images/jump3.jpg"/></div>
    <div .....><img src="images/jump4.jpg"/></div>
  </div>
  <script src="scripts/ ....."></script>
  <.....src="scripts/my_scripts.js"></script>
</body>
<html>
```

index.html

```
$(document).ready(function() {
  $(.....).click( function() {
    var.....= Math.floor((Math.random()*5) + 5);
    var discount = "<p>Your Discount is "+my_num+"%</p>";
    $(.....).append(discount);

    $(".guess_box").each( function(){
      $(this).unbind('click');
    });

.....
});
```

my_scripts.js

my_num

this

});

<head>

".guess_box"

jquery-1.6.2.min.js

<body>

class="guess_box"

script

class="guess_box"

Развлечения с магнитами — решение

Разложите магниты в соответствии со структурой файлов проекта. Убедитесь в том, что вы знаете, как правильно организовать разделение кода HTML, CSS и jQuery. В будущем вы должны безошибочно справляться с этой задачей.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Jump for Joy</title>
    <link href="styles/styles.css" rel="stylesheet">
  </head>
  <body>
    <div id="header">
      <h2>Jump for Joy Sale</h2>
    </div>
    <div  id="main">
      <div class="guess_box" ><img src="images/jump1.jpg"/></div>
      <div class="guess_box"><img src="images/jump2.jpg"/></div>
      <div class="guess_box"><img src="images/jump3.jpg"/></div>
      <div class="guess_box" ><img src="images/jump4.jpg"/></div>
    </div>
    <script src="scripts/ jquery-1.6.2.min.js "></script>
    <  script src="scripts/my_scripts.js"></script>
</body>
<html>
```

index.html

```
$(document).ready(function() {
  $( ".guess_box" ).click( function() {
    var my_num = Math.floor((Math.random()*5) + 5);
    var discount = "<p>Your Discount is "+my_num+"%</p>";
    $( this ).append(discount);

    $(".guess_box").each( function(){
      $(this).unbind('click');
    });
  });
});
```

my_scripts.js

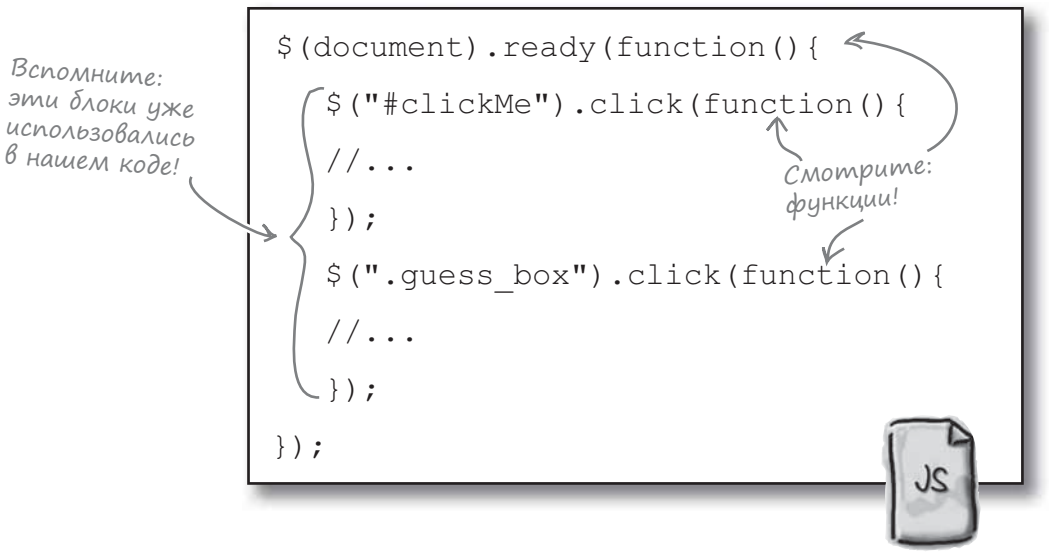

А как было бы замечательно, если бы единожды написанный код jQuery можно было использовать снова и снова? Но это, конечно, всего лишь мечты...

Использование функций

От добавления и удаления событий на страницах мы переходим к другой полезной возможности, которая повысит эффективность использования jQuery на наших сайтах: ***функциям***.

Функцией называется программный блок, отделенный от основного кода, который можно в любой момент выполнить в сценарии.

Представьте себе, мы постоянно использовали функции в этой книге. Помните эти конструкции?

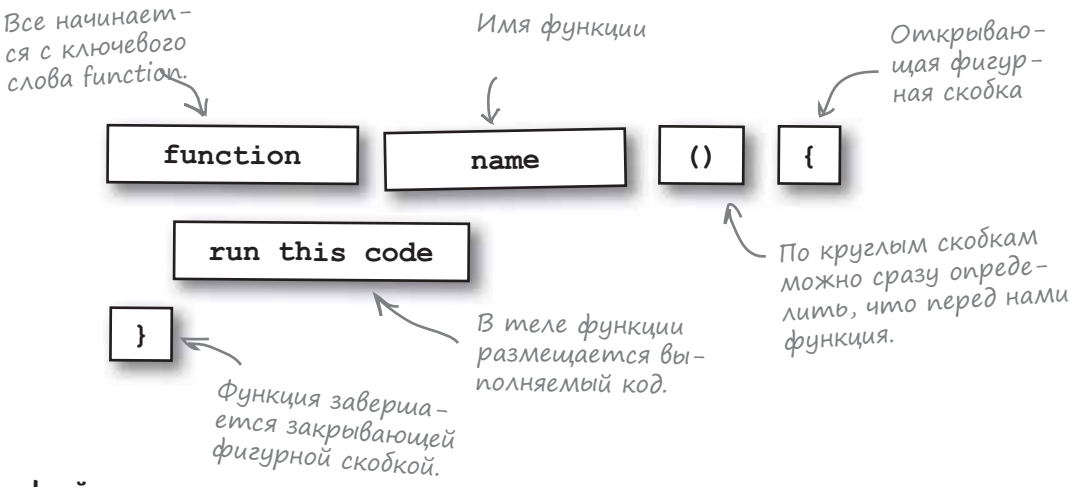


В jQuery существует много готовых функций, но вы также можете писать собственные функции для реализации возможностей, не поддерживаемых в jQuery. Единожды написанную функцию можно использовать снова и снова без повторения ее кода в сценарии. Чтобы выполнить код функции, достаточно вызвать ее по имени.

В сущности, пользовательская функция — это фрагмент кода jQuery, которому присвоено имя для удобства использования.

Как устроена функция

Определение функции связывает имя с выполняемым кодом. Синтаксис большинства базовых функций JavaScript выглядит так:



Имена функций

Присвоить имя функции можно двумя способами.

Объявление функции

Первый способ — *объявление функции* — определяет именованную функциональную переменную без присваивания. Объявления начинаются с ключевого слова `function`:

```
function myFunc1(){
      $("div").hide();
}
```

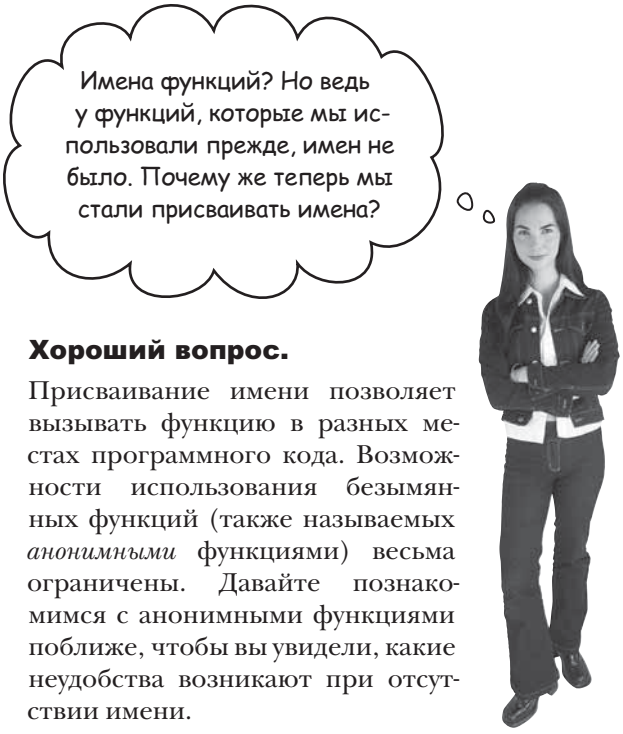
Имя функции

Функциональное выражение

В именованном *функциональном выражении* функция определяется как часть внешней конструкции (чаще всего команды присваивания):

```
var myFunc2 = function() {
      $("div").show();
}
```

Функция присваивается специальной переменной.



Хороший вопрос.

Присваивание имени позволяет вызывать функцию в разных местах программного кода. Возможности использования безымянных функций (также называемых *анонимными* функциями) весьма ограничены. Давайте познакомимся с анонимными функциями поближе, чтобы вы увидели, какие неудобства возникают при отсутствии имени.

Анонимная функция

Анонимные функции определяются без имени и выполняются непосредственно при определении в коде. Все переменные, объявленные в таких функциях, доступны *только во время выполнения функции.*

Эта функция не будет вызываться в других местах кода.

```
$(document).ready(function() {
    $(".guess_box").click( function() {

        var my_num = Math.floor((Math.random()*5) + 5);
        var discount = "<p>Your Discount is "+my_num+"%</p>";
        $(this).append(discount);

        $(".guess_box").each( function(){
            $(this).unbind('click');
        });
    });
});
```

переменные

Если вы захотите использовать этот код в другом месте, его придется продублировать.

my_scripts.js

Будьте осторожны!

Функцию, которой не присвоено имя, нельзя вызывать в других местах.

Часто Задаваемые Вопросы

В: Чем объявление функции отличается от именованного функционального выражения?

О: Прежде всего временем использования. Хотя обе формы делают одно и то же, функция, определяемая в форме *именованного функционального выражения*, может использоваться в коде только после выполнения команды, содержащей выражение. С другой стороны, функция, определяемая в форме *объявления функции*, может вызываться в любом месте страницы (даже в обработчике onload).

В: Существуют ли ограничения для имен, присваиваемых функциям?

О: Да. Имена функций не могут начинаться с цифр, в них не допускается использование математических операторов и знаков препинания, хотя допускается символ подчеркивания (_). Кроме того, имя функции не может содержать пробелы, и в именах функций и переменных учитывается регистр символов.

Именованные функции как обработчики событий

Ранее мы видели, как анонимные функции используются в качестве обработчиков событий. Пользовательские именованные функции тоже могут использоваться для обработки событий, для чего они вызываются непосредственно из кода. Вернемся к примерам именованных функций, приведенным двумя страницами ранее.

Объявление функции

```
function myFunc1(){
       $("div").hide();
}
```

Функциональное выражение

```
var myFunc2 = function() {
       $("div").show();
}
```

Функция вызывается в программном коде.

Здесь нужны круглые скобки.

```
myFunc1();
```

Области div скрываются в соответствии с объявлением функции.

Имена функций

Когда функция вызывается как обработчик события, круглые скобки не нужны.

```
$("#myElement").click(myFunc2);
```

Функция вызывается как обработчик события.

Это означает, что после щелчка на myElement области div будут видны.

Развлечения с магнитами

Разложите магниты так, чтобы код, проверяющий скидку, был выделен в именованную функцию `checkForCode`, которая используется в качестве обработчика события click для областей div.

```
$(document).ready(function() {
  $(".guess_box").click( checkForCode );

.....

    var my_num = Math.floor((Math.random()*5) + 5);
        discount = "<p>Your Discount is "+my_num+"%</p>";
  .....
    $(this).append(          );
                 .....

    $(".guess_box").each( function(){
      $(this).unbind('          ');
                     .....
    });

});
```

"+my_num+"

function

}

discount

()

(

{

click

checkForCode

append

)

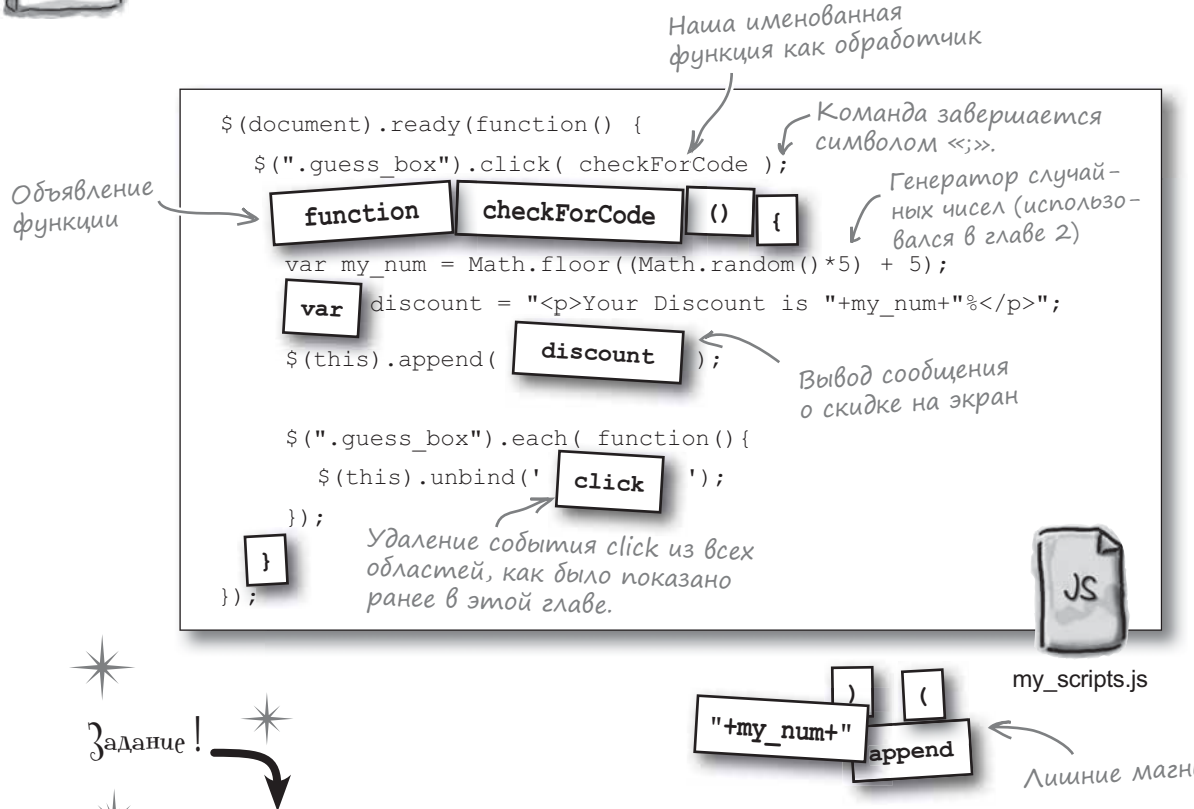
var

JS

my_scripts.js

Развлечения с магнитами. Решение

Разложите магниты так, чтобы код, проверяющий скидку, был выделен в именованную функцию `checkForCode`, которая используется в качестве обработчика события click для областей div.



Задание!

Отличная работа! Внесите эти изменения в файл с кодом. Поздравляем, вы создали свою первую функцию и использовали ее как функцию-обработчик события click.

МОЗГОВОЙ ШТУРМ

Как бы вы написали функцию, скрывающую сообщение о скидке в случайной области div, и другую функцию, генерирующую случайное число для самого сообщения о скидке?

Подсказка: генератор случайных чисел может использоваться как в существующей функции checkForCode, так и в функции выборов сообщения о скидке в случайной области div.

Получается, что функция должна работать по-разному в зависимости от того, на какой области div щелкнул пользователь... Как бы это сделать?

Иногда бывает нужно, чтобы результат работы функции изменялся в зависимости от дополнительной информации, передаваемой при вызове.

Наши функции могут получать *переменные* — как говорилось в главе 2, переменные используются для хранения информации, которая может изменяться со временем. Давайте вспомним, как работают переменные.

Ключевое слово var объявляет переменную.

За ключевым словом var указывается имя переменной.

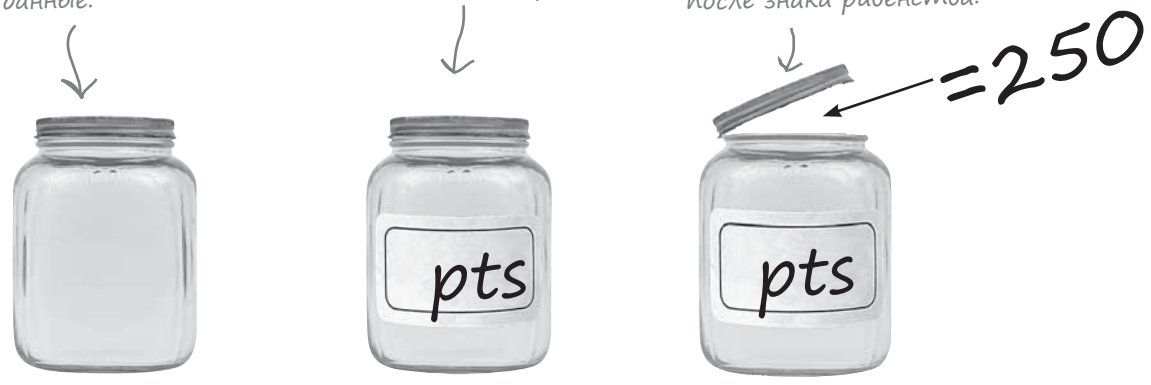
Так значение переменной задается в программном коде.

```
var pts = 250;
```

При объявлении переменной интерпретатор JavaScript выделяет блок памяти браузера, в котором будут храниться ваши данные.

Переменной присваивается имя, по которому вы будете позднее обращаться к ней в своем сценарии.

Значение, сохраняемое в переменной, указывается после знака равенства.

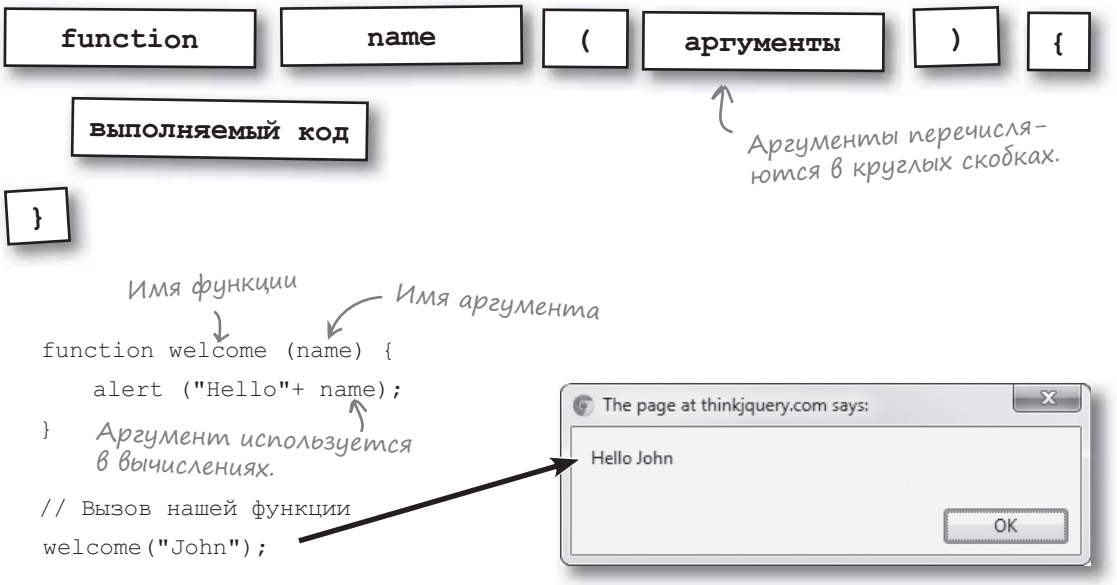


А ведь мы уже использовали переменные в своем коде, помните?

```
var my_num = Math.floor((Math.random()*5) + 5);

var discount = "Your Discount is "+my_num+"%";
```

Передача переменных функциям

Переменные, передаваемые функциям, называются *аргументами*. (Иногда также их называют *параметрами*.) Давайте разберемся, как же происходит передача аргументов функциям.



Нашей функции не нужно знать, какие данные хранятся в переменной; она просто выводит ее текущее содержимое. Таким образом, для изменения результата работы функции достаточно изменить передаваемую переменную (вместо изменения кода самой функции, что безусловно усложнит ее повторное использование!).

Объединение переменных и функций на первый взгляд выглядит невразумительно.

Однако вы можете рассматривать свою функцию как *рецепт* — допустим, рецепт коктейля. Набор простых, легко повторяемых действий для создания коктейля (порцию того, немного этого, перемешать и т. д.) является аналогом функции, а ингредиенты — аналогами передаваемых переменных. Кто-нибудь желает джин-тоник?

Функция также может возвращать значения

Функция возвращает результат своей работы при помощи команды, состоящей из ключевого слова `return` и возвращаемого значения. Результат передается в точку вызова функции для дальнейшего использования.

Возвращаемое значение может быть числом, строкой текста и даже элементом DOM (объектом).

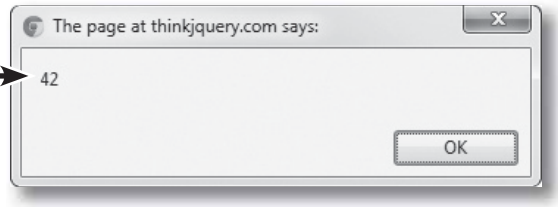
Имя функции

Имена аргументов

```
function multiply (num1, num2) {
   var result = num1*num2;
   return result;
}

// Call our function
var total = multiply (6, 7);

alert (total);
```

Операции с аргументами

Возвращаемое значение



Возьми в руку карандаш

Итак, теперь вы умеете создавать собственные функции. Включите в файл *my_scripts.js* новую функцию, которая получает один аргумент (с именем `num`) и возвращает случайное число в интервале от 0 до переданного числа. Присвойте функции имя `getRandom`.

Подсказка: Вспомните код, который мы использовали для генерации случайных чисел. Подумайте, как оформить его в виде функции.

Возьми в руку карандаш
Решение

Итак, теперь вы умеете создавать собственные функции. Включите в файл *my_scripts.js* новую функцию, которая получает один аргумент (с именем `num`) и возвращает случайное число в интервале от 0 до переданного числа. Присвойте функции имя `getRandom`.

Имя функции

Имя аргумента

```
function getRandom(num){
    var my_num = Math.floor(Math.random()*num);
    return my_num;
}
```

Функция возвращает случайное число.

my_scripts.js

Функции и аргументы — это, конечно, хорошо, но насколько мы приблизились к завершению кода страницы «Прыгаем от радости»?

Выглядит неплохо! Но это только начало — посмотрим, что ваша группа скажет о выборе «правильного» изображения...

Джим: Кроме новой функции `getRandom`, нам понадобится еще одна функция.

Фрэнк: Точно — для включения кода скидки в случайное выбранное изображение... Кстати, в ней можно использовать функцию `getRandom`.

Джо: Логично. После щелчка мы проверяем, правильно ли пользователь выбрал изображение.

Джим: Погодите-ка, что? Как проверить, правильно ли выбрано изображение?

Фрэнк: Условная конструкция!

Джим: Что?

Фрэнк: Условные конструкции позволяют проверить некоторое условие и выполнить код в зависимости от результата проверки.

Джо: Например, можно проверить, что переменная имеет определенное значение или что два значения равны?

Фрэнк: Точно! Мы даже можем проверить, находится ли элемент внутри другого элемента... Думаю, здесь нам это пригодится.

Условные конструкции и принятие решений

В jQuery используются *условные конструкции* JavaScript, позволяющие выполнять разный код в зависимости от решения, принятого в коде на основе уже имеющейся информации. Ниже приведен пример использования условных конструкций JavaScript. Другие примеры будут представлены в главе 6.

Начало условной команды if

Проверяемое условие

```
if( myBool == true ){
    // Здесь что-то делаем!
}else{
    // Иначе делаем что-то другое!
}
```

Оператор проверки равенства

Переменная JavaScript

Код, который должен выполняться, если условие истинно.

Код, который должен выполняться, если условие ложно.

Примечание: команда if может и не содержать секции else, но вреда от нее не будет.

Развлечения с магнитами

Расставьте магниты так, чтобы создать именованную функцию `hideCode`, которая при помощи условной конструкции прячет новый элемент `span` с идентификатором `has_discount` в случайно выбранном элементе `div` класса `.guess_box`.

```
var ..... = function (){
    var numRand = .....(4);
    $(.....).each(function(index, value) {
        if(numRand == index){
            $(this).append("<span id='.....'></.....>");
            return false;
        }
    .....
}
```

my_scripts.js

".guess_box"

span

()

hideCode

{

});

)

getRandom

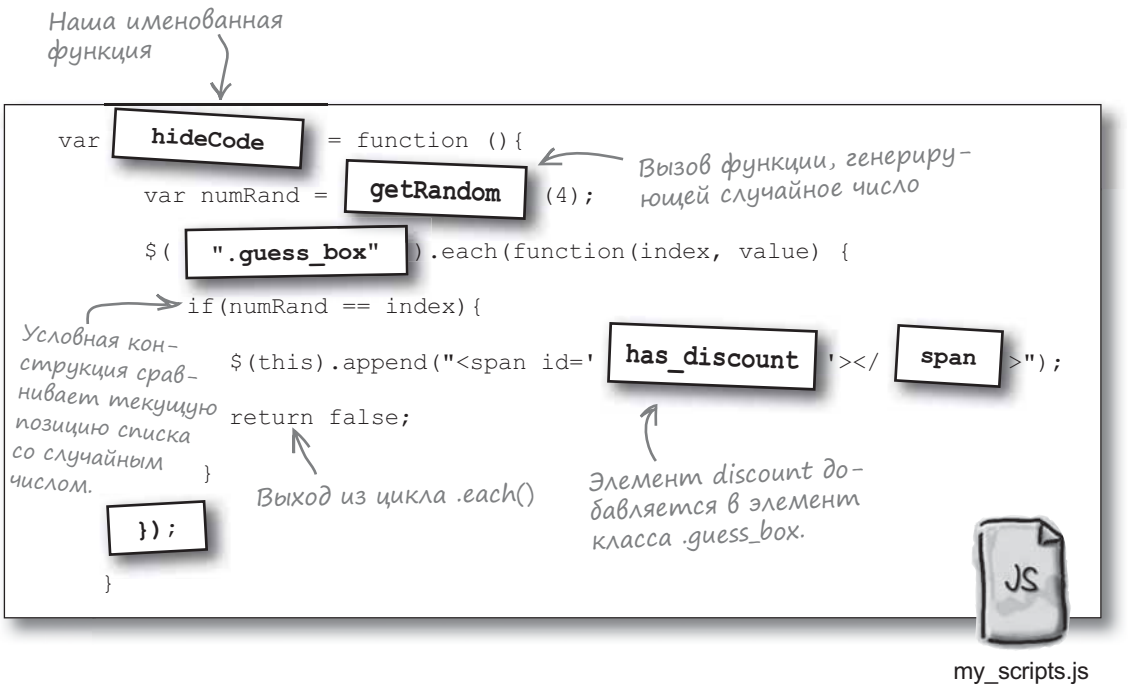
has_discount

(

}

Развлечения с магнитами. Решение

Расставьте магниты так, чтобы создать именованную функцию `hideCode`, которая при помощи условной конструкции прячет новый элемент `span` с идентификатором `has_discount` в случайно выбранном элементе `div` класса `.guess_box`. Далее приводится наше решение.

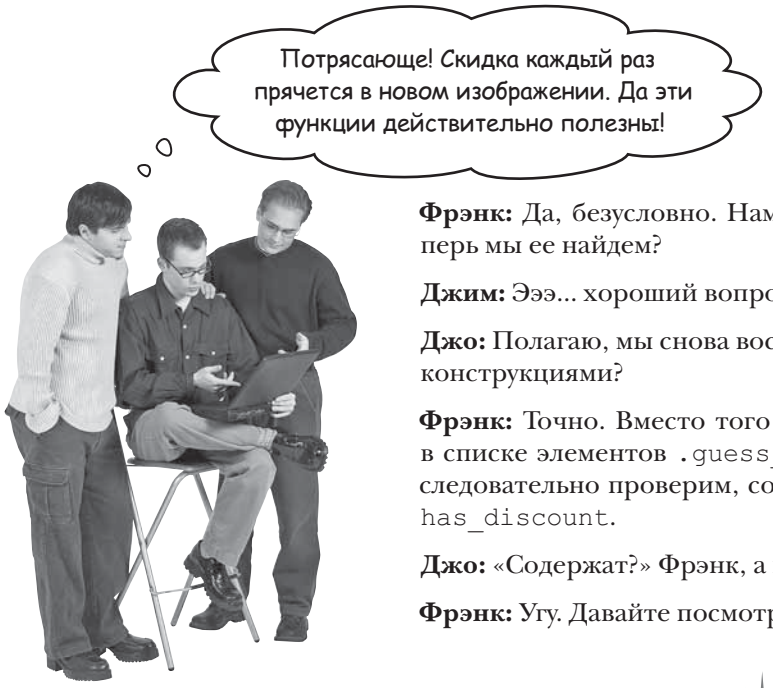


Будьте осторожны!

***Индекс* элемента списка обозначает его текущую позицию в списке.**

Индексы элементов всегда начинаются с 0.

Таким образом, индекс первого элемента равен 0, индекс второго — 1, и т. д. Мы еще вернемся к теме индексирования при рассмотрении массивов и циклов в главе 6.



Фрэнк: Да, безусловно. Нам удалось спрятать скидку, но как теперь мы ее найдем?

Джим: Эээ... хороший вопрос. Я не знаю.

Джо: Полагаю, мы снова воспользуемся волшебными условными конструкциями?

Фрэнк: Точно. Вместо того чтобы выбирать случайный индекс в списке элементов `.guess_box`, мы переберем элементы и последовательно проверим, содержат ли они спрятанный элемент `has_discount`.

Джо: «Содержат?» Фрэнк, а ведь в этом что-то есть.

Фрэнк: Угу. Давайте посмотрим, чем нам поможет jQuery.

Задание!

Включите в функцию `checkForCode` новый код, основанный на идеях Джима, Фрэнка и Джо.

Объявление переменной discount

Условная конструкция проверяет, нашел ли пользователь элемент div со скидкой.

Текущий элемент — т. е. элемент, вызвавший функцию.

Поиск элемента DOM с идентификатором has_discount.

Этот метод jQuery проверяет, содержит ли первый параметр то, что определяется вторым параметром.

```
function checkForCode(){
    var discount;
    if($.contains(this, document.getElementById("has_discount") ) )
    {
       var my_num = getRandom(5);
       discount = "<p>Your Discount is "+my_num+"%</p>";
    }else{
       discount = "<p>Sorry, no discount this time!</p>" ;
    }
    $(this).append(discount);

    $(".guess_box").each( function(){
       $(this).unbind('click');
    });
}
```

В зависимости от того, нашел пользователь скидку или нет, выводятся разные сообщения.

my_scripts.js

Задание!

Пришло время написать несколько функций: первая генерирует случайное число, вторая прячет признак скидки, а третья проверяет наличие спрятанного признака.

```
$(document).ready(function() {
    $(".guess_box").click( checkForCode );
    function getRandom(num){
        var my_num = Math.floor(Math.random()*num);
        return my_num;
    }
    var hideCode = function(){
        var numRand = getRandom(4);
        $(".guess_box").each(function(index, value) {
            if(numRand == index){
                $(this).append("<span id='has_discount'></span>");
                return false;
            }
        });
    }
    hideCode();
    function checkForCode(){
        var discount;
        if($.contains(this, document.getElementById("has_discount") ) )
        {
            var my_num = getRandom(5);
            discount =  "<p>Your Discount is "+my_num+"%</p>" ;
        }else{
            discount = "<p>Sorry, no discount this time!</p>" ;
        }
        $(this).append(discount);
        $(".guess_box").each( function(){
            $(this).unbind('click');
        });
    }
}); //End document.ready()
```

Функция вызывается при щелчке на элементе, принадлежащем классу .guess_box.

Функция, генерирующая случайное число

Именованная функция, скрывающая признак скидки

Вызов именованной функции

А эта функция проверяет и сообщает размер скидки.

my_scripts.js

Но это еще не все

Только вы решили, что работа над рекламной акцией «Прыгаем от радости» завершена, как у Эмили появились новые требования...

От: **Эмили**
Тема: **RE: Рекламная акция «Прыгаем от радости»**

Привет,

Спасибо за все, что ты для нас сделал.

Я подумала, а нельзя ли «подсветить» изображение перед щелчком? Пользователь будет точно знать, на каком изображении он щелкает, а это поможет избежать недоразумений.

Кроме того, нельзя ли вывести сообщение о скидке в удобной текстовой области под изображениями на экране? Допустим, код скидки будет состоять из текста и числа? Я подумала, что это будет удобно... Да, и нельзя ли, чтобы число было больше 10? Скажем, в диапазоне от 1 до 100?

Напиши, сможешь ли ты внести эти небольшие изменения.

--
Эмили Сондерс
jumpforjoyphotos.hg

Возьми в руку карандаш

Вы уже знаете, что делать. Выделите требования из сообщения Эмили.

Требования:

Возьми в руку карандаш Решение

Вы уже знаете, что делать. Выделите требования из сообщения Эмили.

Требования:

- *Визуально выделить область перед щелчком, чтобы пользователь точно знал, какой вариант он выбирает.*
- *Вывести код скидки в специальной области страницы. Код должен состоять из текста и числа от 1 до 100.*

Часто Задаваемые Вопросы

В: Обязательно ли указывать возвращаемое значение во всех функциях?

О: Формально необязательно. Все функции возвращают значение независимо от того, указали вы его или нет. Если возвращаемое значение не указано, то функция возвращает неопределенное значение `undefined`. Если ваш код не может обработать `undefined`, это приведет к ошибке. Следовательно, возвращаемое значение все же лучше указывать, даже в простейшем виде `return false;`.

В: Существуют ли ограничения для аргументов или параметров, передаваемых функции?

О: Нет, функции можно передать любой объект, элемент, переменную или значение. Также можно передать большее количество параметров, чем ожидает ваша функция. Лишние параметры игнорируются. Если передать слишком мало параметров, то недостающим параметрам автоматически присваивается `undefined`.

В: Что делает метод `$.contains`?

О: Это статический метод библиотеки jQuery, который получает два параметра. Он проверяет все дочерние элементы первого параметра, ищет среди них второй параметр и возвращает `true` или `false` в зависимости от результата. В нашем примере выражение `$.contains(document.body, document.getElementById "header"))` истинно; с другой стороны, выражение `$.contains(document.getElementById("header"), document.body)` будет ложно.

В: Что такое «статический метод» jQuery?

О: Это означает, что функция связана с библиотекой jQuery, а не с конкретным объектом. Для вызова таких методов не нужен селектор, достаточно указать имя jQuery или его сокращенную форму (`$`).

В: Что делают `index` и `value` в нашей функции-обработчике `.each`?

О: `index` обозначает текущую итерацию цикла (начиная с 0 для первого элемента массива, возвращаемого селектором). `value` обозначает текущий объект, это аналог `$(this)` в цикле `.each`.

В: Почему наш цикл `.each` в функции `hideCode` возвращает `false`?

О: Команда return false в цикле `.each` приказывает прервать цикл и продолжить выполнение кода. При возвращении любого значения, отличного от false, цикл переходит к следующему элементу в списке. В нашем примере скидка уже спрятана, поэтому остальные элементы перебирать необязательно.

Мозговой штурм

Как показать пользователю, на какой области он собирается щелкнуть, ***до*** выполнения щелчка?

Методы могут изменять CSS

Следующая задача — «подсветить» область, на которую наведен указатель мыши, перед щелчком. Для изменения внешнего вида элемента проще всего воспользоваться CSS и классами CSS.

К счастью, jQuery позволяет легко назначать элементам классы CSS и удалять их при помощи простых, удобных методов. Давайте посмотрим, как применить их в нашем решении.

Помните по главам 1 и 2?

Готово к употреблению

Создайте новые файлы отдельно от существующих и проследите за тем, как работают новые методы. Это поможет вам понять, как происходит «подсветка» области перед щелчком.

```
.hover{
   border: solid #f00 3px;
}
.no_hover{
   border: solid #000 3px;
}
```

test_style.css

```
<html>
  <head>
    <link href="styles/test_style.css" rel="stylesheet">
  </head>
  <body>
    <div id="header" class="no_hover"><h1>Header</h1></div>
    <button type="button" id="btn1">Click to Add</button>
    <button type="button" id="btn2">Click to Remove</button>
    <script src="scripts/jquery.1.6.2.js"></script>
    <script src="scripts/my_test_scripts.js"></script>
  </body>
</html>
```

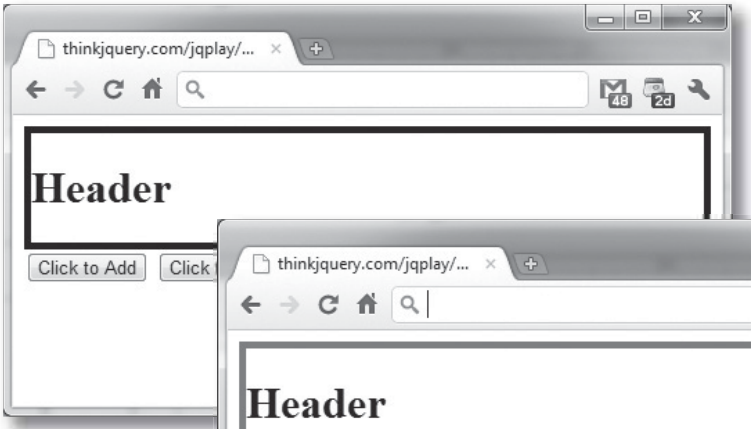
class_test.html

```
$(document).ready(function() {
   $("#btn1").click( function(){
      $("#header").addClass("hover");
      $("#header").removeClass("no_hover");
   });
   $("#btn2").click( function(){
      $("#header").removeClass("hover");
      $("#header").addClass("no_hover");
   });
});
```

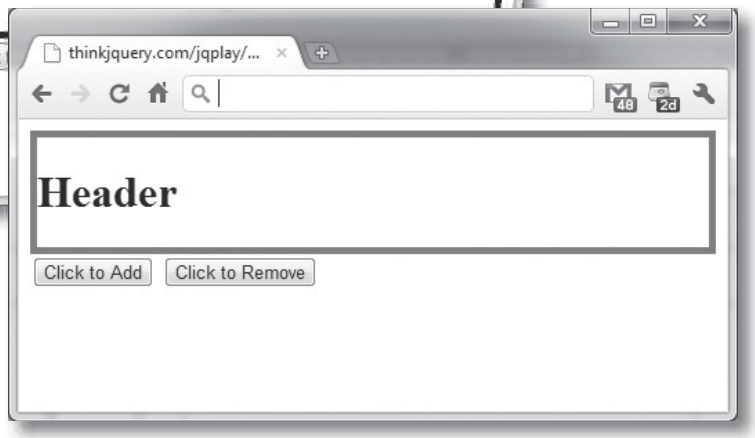
my_test_scripts.js

ТЕСТ-ДРАЙВ

Откройте только что созданный файл *class_test.html* в браузере. При щелчке на кнопке Add класс применяется к области `div` с идентификатором `header`. Кнопка Remove снова удаляет назначенный класс!



До щелчка



После щелчка

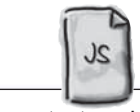
Здорово! Если бы все было так просто... А изменение CSS может срабатывать по другим причинам, кроме события click?

Да, может. И ничуть не сложнее...

Переключение CSS может происходить по событию любого типа, но для нашего решения нужен вполне определенный тип события. Вернитесь к списку на с. 116 и попробуйте определить, какое событие следует использовать в нашем примере.

Добавление события hover

В параметрах события `hover` могут передаваться две функции-обработчика для событий `mouseenter` и `mouseleave`. Эти функции могут быть как именованными, так и анонимными. Внимательно просмотрите код тестового сценария и подумайте, как использовать событие `hover` для изменения внешнего вида элемента при наведении на него указателя мыши.



my_test_scripts.js

```
$(document).ready(function() {
  $("#btn1").click( function(){
    $("#header").addClass("hover");
    $("#header").removeClass("no_hover");
  });
  $("#btn2").click( function(){
    $("#header").removeClass("hover");
    $("#header").addClass("no_hover");
  });
});
```

Упражнение

Измените файлы *my_style.css* и *my_scripts.js* так, чтобы изображения визуально выделялись при наведении на них указателя мыши. Вам понадобится новый класс CSS и две функции-обработчика (после функции `checkForCode`), использующие методы `addClass` и `removeClass` для назначения/отмены класса CSS. Мы уже написали заготовки этих функций; вам остается лишь заполнить пропуски.

my_style.css

```
$(".guess_box").hover(
  function () {
    // Обработчик события mouseenter
    ..... .
  },
  function () {
    // Обработчик события mouseleave
    .....
});
```

my_scripts.js

Упражнение Решение

У нас есть класс CSS, для управления которым используется событие `hover`.

```
.my_hover{
  border: solid #00f 3px;
}
```

my_style.css

Это новый класс.

Назначает класс CSS элементу, на который пользователь наводит указатель мыши, при помощи анонимной функции-обработчика для события mouseenter.

Метод jQuery addClass назначает элементу класс CSS. Он не влияет на другие классы, назначенные элементу ранее.

Анонимная функция-обработчик для события mouseleave.

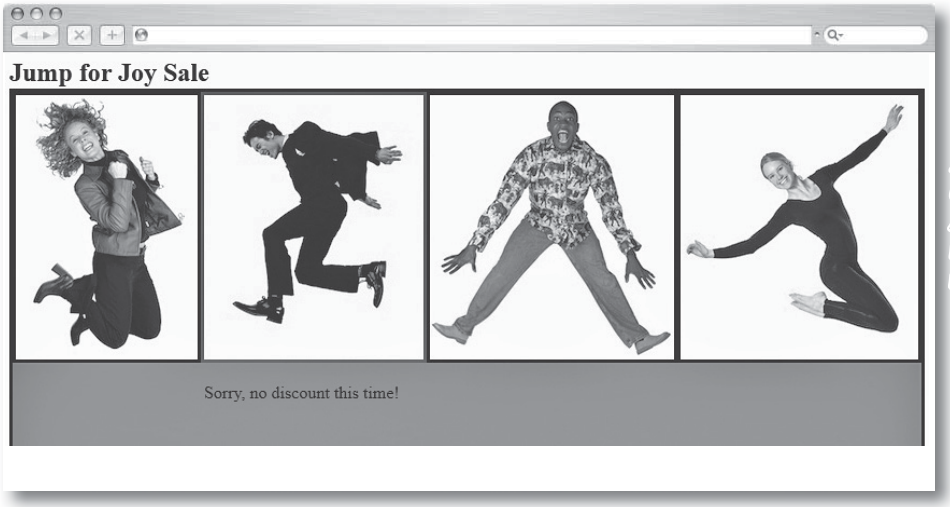
Метод jQuery removeClass отменяет назначение класса CSS элементу.

```
$(".guess_box").hover(
  function () {
    // this is the mouseenter event handler
    $(this).addClass("my_hover");
  },
  function () {
    // this is the mouseleave event handler
    $(this).removeClass("my_hover");
});
```

my_scripts.js

ТЕСТ-ДРАЙВ

Откройте в браузере файл *index.html*, который включает ваш новый файл *my_scripts.js*. Наведите мышь на изображение и посмотрите, изменится ли внешний вид рамки.



Хмм... Рамка изображения меняет цвет, но работа еще не закончена...

Еще немного...

Безусловно, мы продвигаемся вперед, но сообщение все еще выводится в неправильном месте и выглядит не так, как нас просили. Вдобавок в первом сообщении осталось еще одно требование, которое мы не выполнили. Вот как список требований выглядит на данный момент.

- ~~Визуально выделить область перед щелчком, чтобы пользователь точно знал, какой вариант он выбирает.~~
- Вывести код скидки в специальной области страницы. Код должен состоять из текста и числа от 1 до 100.
- После того как посетитель выбрал изображение и щелкнул на нем, следует сообщить, была ли его догадка успешной. Если изображение было выбрано правильно, нужно вывести величину скидки.

Это требование еще из первого сообщения!

Упражнение

Измените код `checkForCode` так, чтобы функция:

1) выводила код скидки в специальной области страницы;

2) создавала код в виде комбинации из букв и числа в интервале от 1 до 100;

3) показывала посетителю, где был спрятан код, если попытка оказалась неудачной.

Чтобы упростить вашу задачу, мы создали классы CSS. Включите их в файл *my_styles.css* для обозначения того, правильно ли было выбрано изображение.

Заодно разместите под четырьмя изображениями элемент span с идентификатором `result`, в котором будет выводиться код скидки.

```
.discount{
  border: solid #0f0 3px;
}
.no_discount{
  border: solid #f00 3px;
}
```

my_style.css

Упражнение
Решение

Функция `checkForCode` дополнена всеми необходимыми составляющими: отдельным местом на экране для вывода кода скидки; самим кодом скидки, состоящим из текста и числа от 1 до 100; и выводом информации о местонахождении скидки после щелчка.

```
function checkForCode(){
   var discount;
   if($.contains(this, document.getElementById("has_discount") ) )
   {
      var my_num = getRandom(100);
      discount = "<p>Your Code: CODE"+my_num+"</p>" ;
   }else{
      discount = "<p>Sorry, no discount this time!</p>" ;
   }
   $(".guess_box").each(function() {
      if($.contains(this, document.getElementById("has_discount")))
      {
      $(this).addClass("discount");
      }else{
      $(this).addClass("no_discount");
      }
      $(this).unbind();
   });
   $("#result").append(discount);
} // End checkForCode function
```

Проверяем, содержит ли текущая область код скидки, с использованием функции jQuery contains.

Используем функцию getRandom для расширения числовой составляющей кода до 100.

Выбираем выводимое сообщение в зависимости от того, нашел пользователь скидку или нет.

Если пользователь угадал – визуально изменяем область, чтобы обозначить местонахождение скидки...

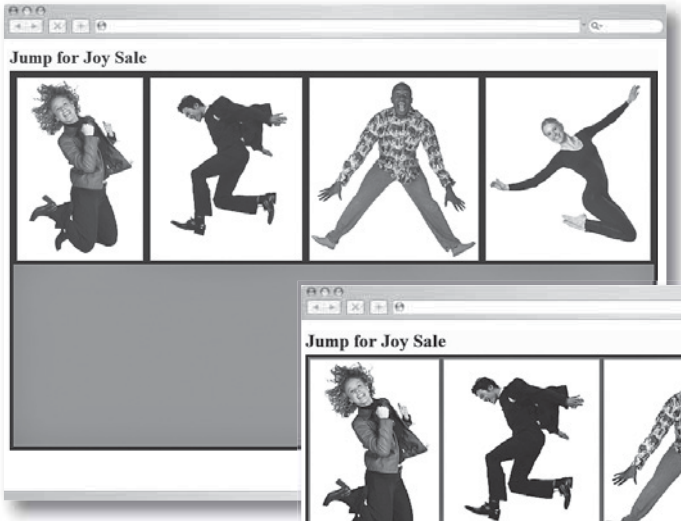
...а если не угадал — используем другое оформление.

Сообщение выводится на странице в специальной области.

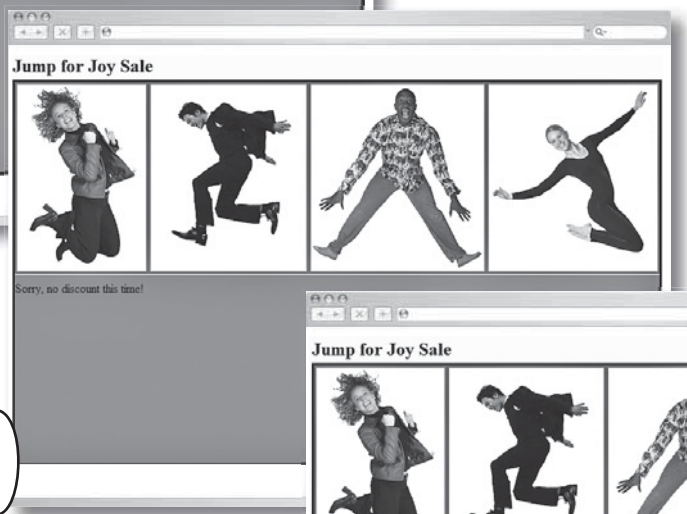
my_scripts.js

Тест-драйв

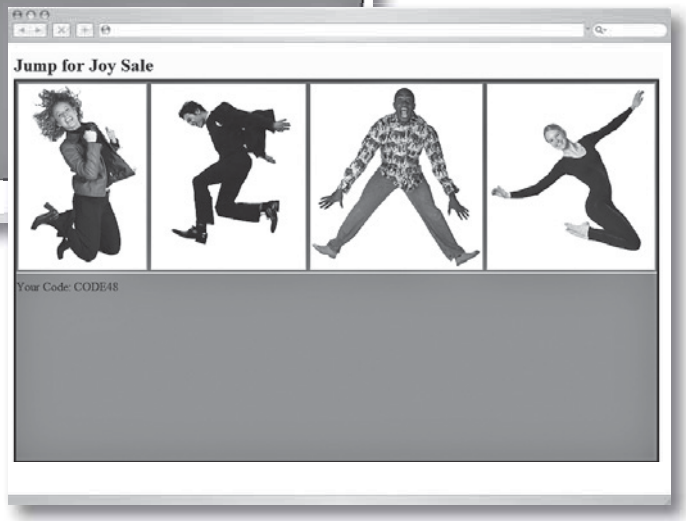
Итак, функция `checkForCode` была успешно изменена. Проверьте, как работают новые возможности сайта. (Файл с окончательной версией кода можно загрузить по адресу *http://thinkjquery.com/chapter03/end/scripts/my_scripts.js*.)



Загруженная страница



Скидки нет



Код скидки

ГЛАВА 3

Ваш инструментарий jQuery

Глава 3 осталась позади, а ваш творческий инструментарий расширился: в нем появились события, пользовательские функции и условные конструкции.

Функции

Фрагменты кода, которые могут использоваться в других местах...

...но только в том случае, если им присвоены имена.

Безымянные функции выполняются только в точке своего определения и не могут использоваться в других местах.

Функции могут получать переменные (называемые аргументами или параметрами), а также могут возвращать результаты своей работы.

Условные конструкции

Проверка логических условий (if XYZ = true) перед выполнением каких-либо операций.

Часто содержат секцию else для случая, если проверяемое условие ложно, но присутствие этой секции необязательно.

События

Объекты, которые упрощают взаимодействие пользователя с веб-страницей.

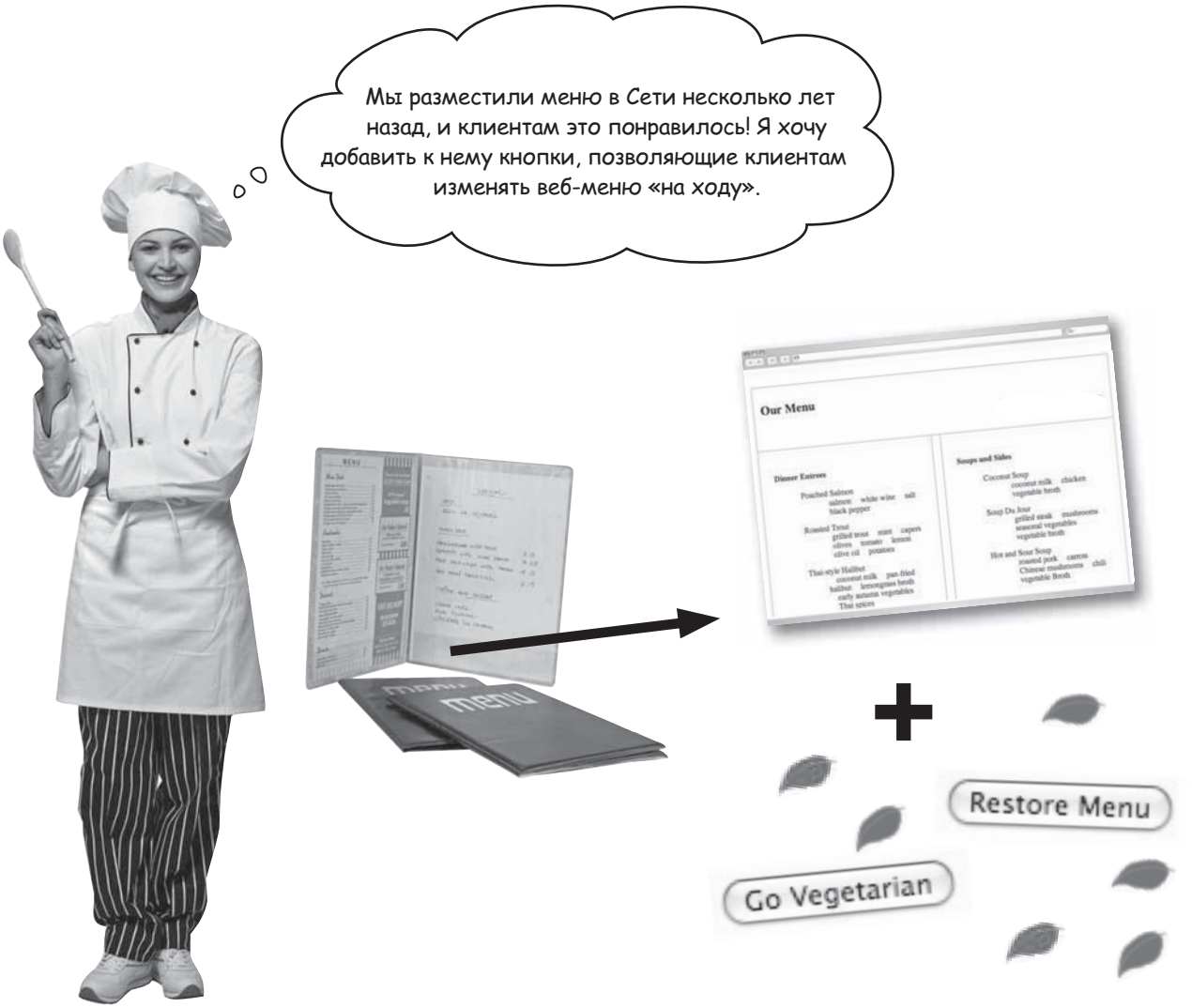
Всего существует около 30 типов событий. Практически все, что происходит в браузере, может стать источником события.

Изменение DOM

Завершение загрузки страницы еще не означает, что ее структура останется неизменной. В главе 1 было показано, как в процессе загрузки страницы строится модель DOM, определяющая ее структуру. В этой главе вы научитесь перемещаться по дереву DOM, работать с иерархией элементов и отношениями «родитель/потомок» для изменения структуры страницы «на ходу» средствами jQuery.

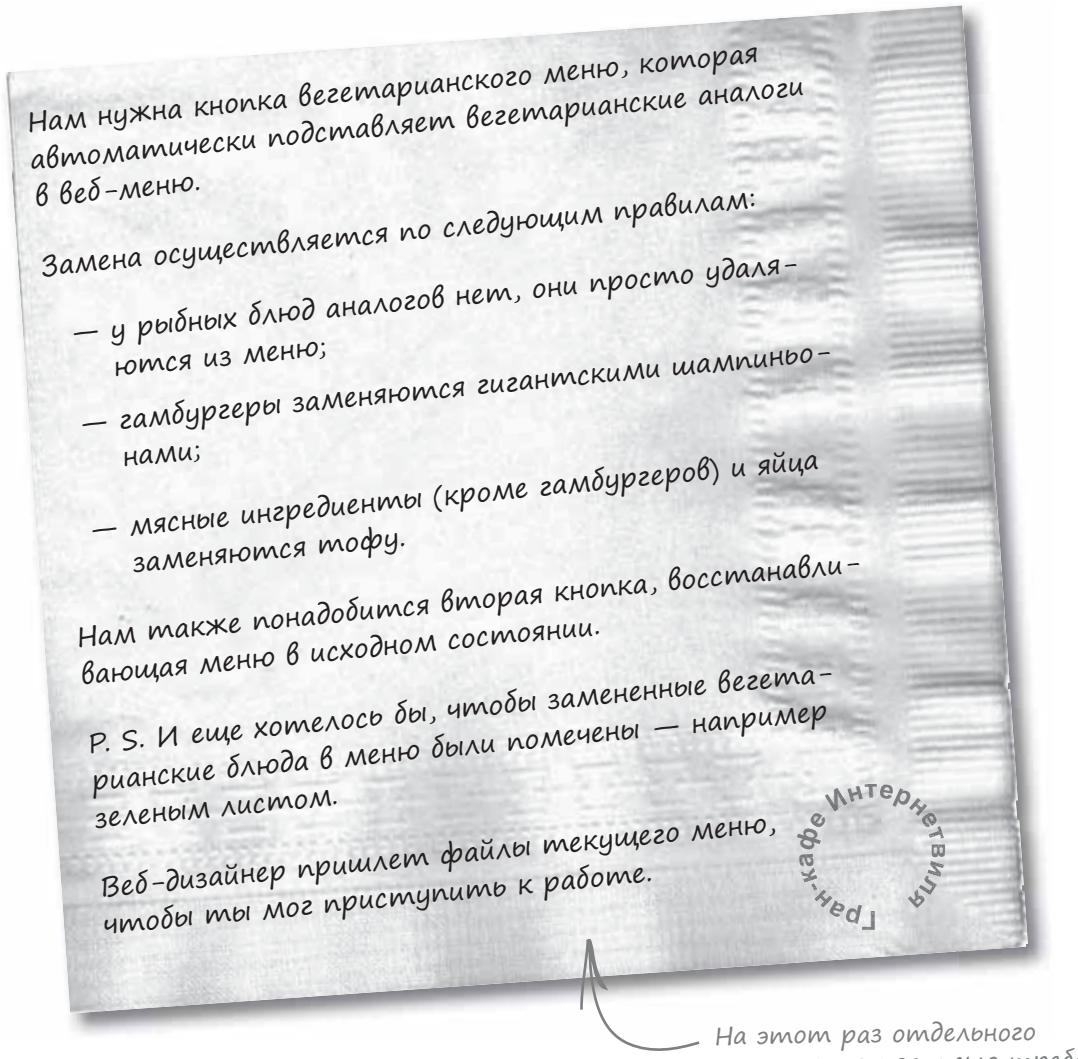
Интерактивное меню

У Александры, шеф-повара небольшого кафе в городке Интернетвиль, есть для вас работа. Ей приходится вести разные веб-страницы для двух разных версий своего меню: обычной и вегетарианской. Александра хочет создать одну страницу, которая бы при необходимости могла изменить меню для клиентов-вегетарианцев.



Вегетарианцы, вперед!

Вот чего хочет Александра:

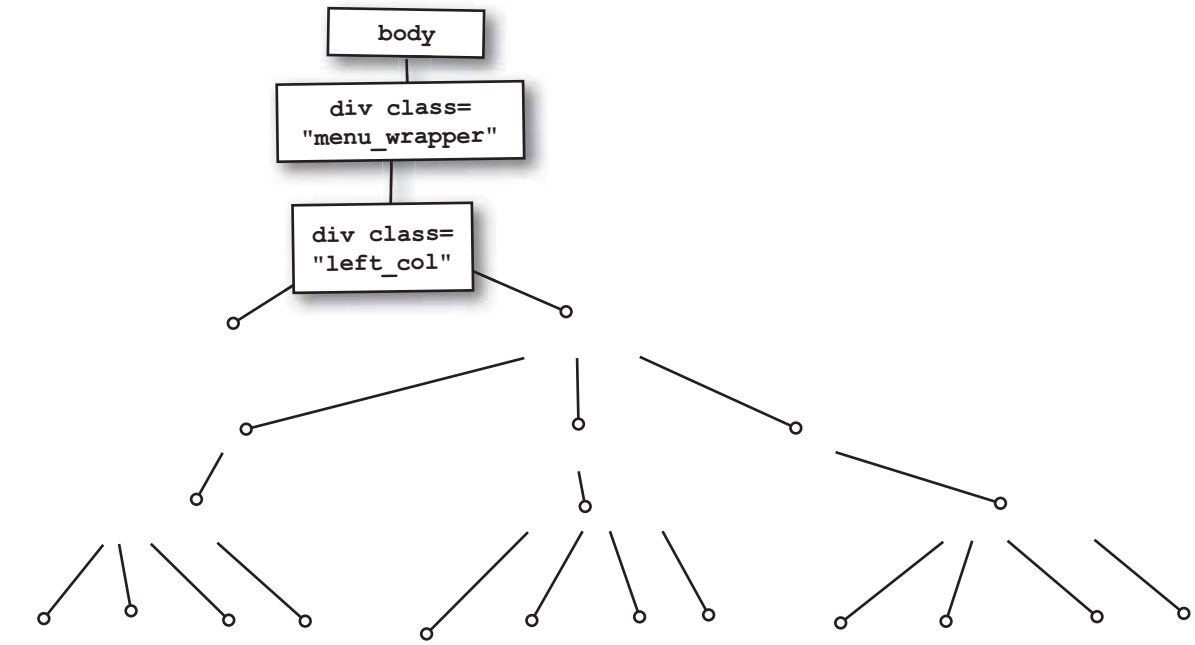


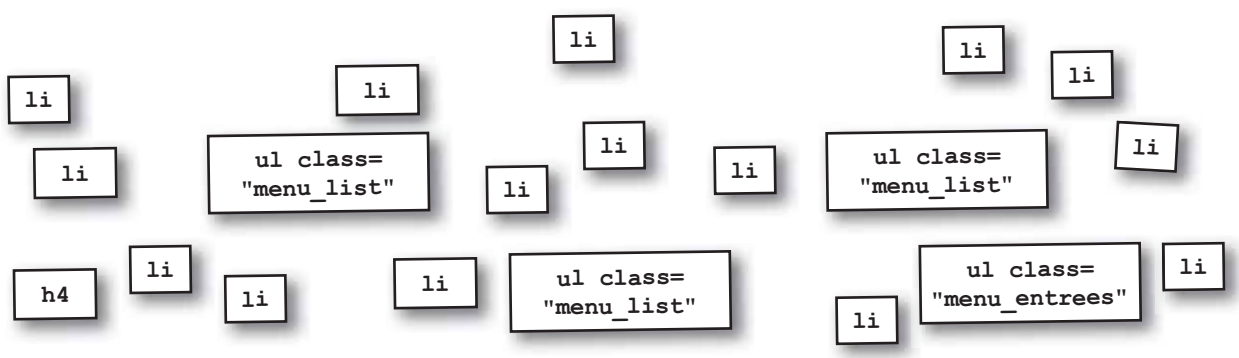
На этот раз отдельного упражнения с записью требований не будет (хотя, наверное, вы уже к ним привыкли?). Тем не менее обязательно изложите суть требований своими словами — это поможет вам четко представлять, что именно мы здесь строим.

Прежде чем браться за код jQuery, давайте посмотрим, какие файлы HTML и CSS нам прислал веб-дизайнер и насколько хороши их стиль и структура.

Развлечения с магнитами

Проанализируйте текущую структуру веб-меню и постройте диаграмму его модели DOM. Ниже вы найдете все магниты с элементами, необходимыми для заполнения всех пропусков. Достройте дерево, используя фрагмент разметки HTML на следующей странице. Позиции для размещения магнитов обозначены кружочками. Мы уже разместили несколько элементов за вас.





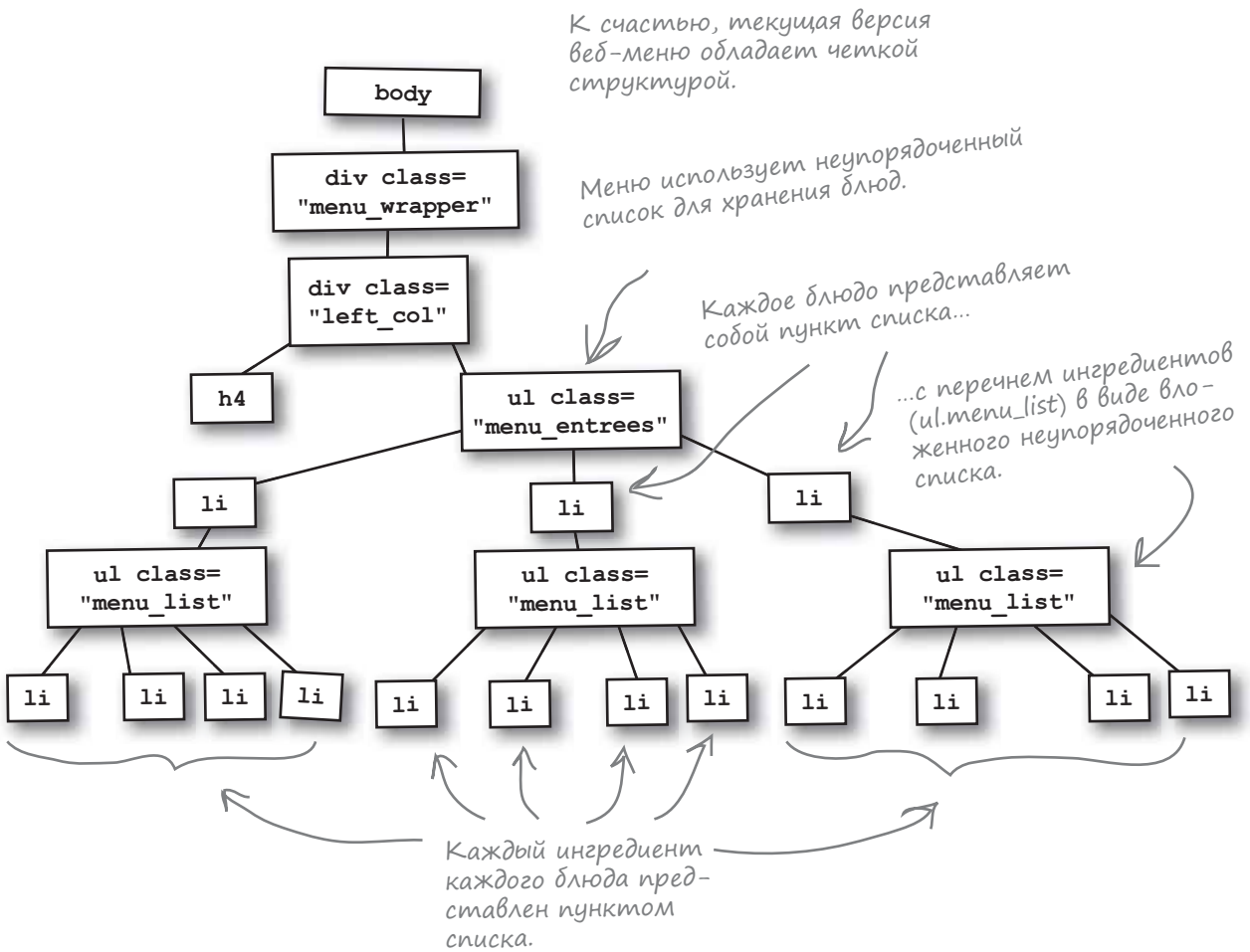
Фрагмент реальной страницы HTML

```
<body>
 <div id="menu_wrapper">
  <div class="left_col">
   <h4>Dinner Entrees</h4>
   <ul class="menu_entrees">
       <li>Thai-style Halibut
              <ul class="menu_list">
                     <li>coconut milk</li>
                     <li>pan-fried halibut</li>
                     <li>early autumn vegetables</li>
                     <li>Thai spices </li>
              </ul>
       </li>
       <li>House Grilled Panini
              <ul class="menu_list">
                     <li>prosciutto</li>
                     <li>provolone</li>
                     <li>avocado</li>
                     <li>sourdough roll</li>
              </ul>
       </li>
       <li>Southwest Slider
              <ul class="menu_list">
                     <li>whole chiles</li>
                     <li>hamburger</li>
                     <li>pepperjack cheese</li>
                     <li>multigrain roll</li>
              </ul>
       </li>
   </ul>
  </div>
 </div>
</body>
```

index.html

Развлечения с магнитами. Решение

Похоже, все ингредиенты создаются как дочерние элементы по отношению к родительским элементам списка блюд. Такую разметку никак не назовешь предельно понятной или однозначной, не правда ли?



```
<body>
 <div id="menu_wrapper">
  <div class="left_col">
   <h4>Dinner Entrees</h4>
   <ul class="menu_entrees">
       <li>Thai-style Halibut
              <ul class="menu_list">
                     <li>coconut milk</li>
                     <li>pan-fried halibut</li>
                     <li>early autumn vegetables</li>
                     <li>Thai spices </li>
              </ul>
       </li>
       <li>House Grilled Panini
              <ul class="menu_list">
                     <li>prosciutto</li>
                     <li>provolone</li>
                     <li>avocado</li>
                     <li>sourdough roll</li>
              </ul>
       </li>
       <li>Southwest Slider
              <ul class="menu_list">
                     <li>whole chiles</li>
                     <li>hamburger</li>
                     <li>pepperjack cheese</li>
                     <li>multigrain roll</li>
              </ul>
       </li>
   </ul>
  </div>
 </div>
</body>
```

Фрагмент реальной страницы HTML

index.html

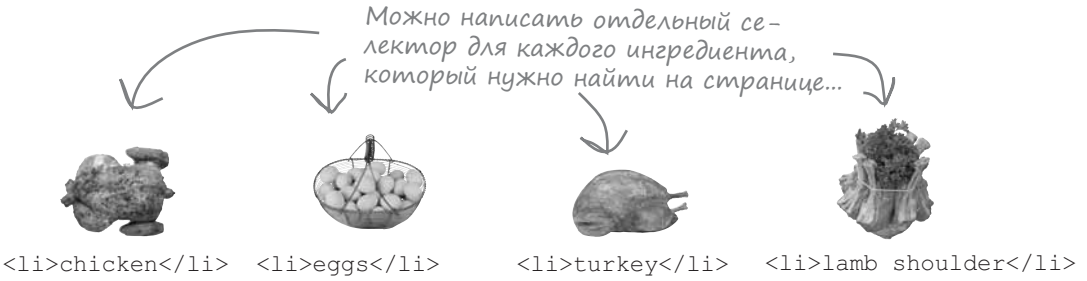
МОЗГОВОЙ ШТУРМ

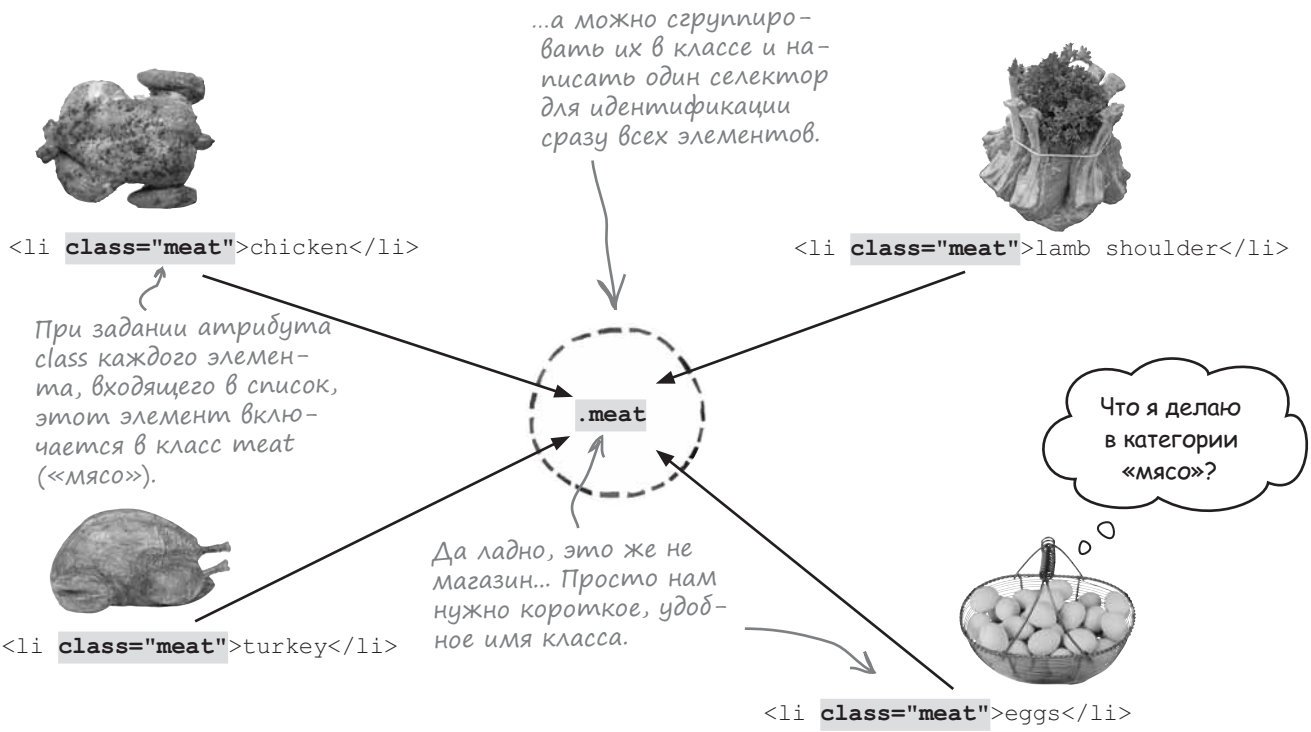
Четкая структура веб-страницы (HTML) существенно упрощает написание кода jQuery. Однако элементы ингредиентов, которые нужно найти, не имеют пометок, которые бы упрощали программирование jQuery. Как же упростить выбор элементов?

Назначение классов элементам

Как было показано во всех предыдущих главах, эффективность поиска элементов веб-страниц средствами jQuery можно повысить за счет правильного определения структуры HTML и CSS. Чтобы структура по-настоящему заработала, следует включить в таблицу стилей классы и идентификаторы и задать соответствующие классы и идентификаторы в атрибутах элементов HTML. Это упростит выбор элементов и сэкономит ваше время при программировании.

В jQuery роль селекторов не ограничивается управлением внешним видом и поведением страницы. Селекторы также используются в jQuery для идентификации элементов страниц.





Упражнение

Найдите ингредиенты, которые должны заменяться вегетарианскими аналогами, и назначьте каждому из них соответствующий класс (fish, meat или hamburger). Если ингредиенту не нужен класс, оставьте строку пустой. Структура кода HTML соответствует внешнему виду меню на странице.

```
<li>Thai-style Halibut
  <ul class="menu_list">
    <li.....>coconut milk</li>
    <li.....>pan-fried halibut</li>
    <li.....>lemongrass broth</li>
    <li.....>vegetables</li>
    <li.....>Thai spices </li>
  </ul>
</li>

<li>Braised Delight
  <ul class="menu_list">
    <li.....>lamb shoulder</li>
    <li.....>cippolini onions</li>
    <li.....>carrots</li>
    <li.....>baby turnip</li>
    <li.....>braising jus</li>
  </ul>
</li>

<li>House Grilled Panini
  <ul class="menu_list">
    <li.....>prosciutto</li>
    <li.....>provolone</li>
    <li.....>avocado</li>
    <li.....>cherry tomatoes</li>
    <li.....>sourdough roll</li>
    <li.....>shoestring fries </li>
  </ul>
</li>

<li>House Slider
  <ul class="menu_list">
    <li.....>eggplant</li>
    <li.....>zucchini</li>
    <li.....>hamburger</li>
    <li.....>balsamic vinegar</li>
    <li.....>onion</li>
    <li.....>carrots</li>
    <li.....>multigrain roll</li>
    <li.....>goat cheese</li>
  </ul>
</li>

<li>Frittata
  <ul class="menu_list">
    <li.....>eggs</li>
    <li.....>Asiago cheese</li>
    <li.....>potatoes </li>
  </ul>
</li>

<li>Coconut Soup
  <ul class="menu_list">
    <li.....>coconut milk</li>
    <li.....>chicken</li>
    <li.....>vegetable broth</li>
  </ul>
</li>

<li>Soup Du Jour
  <ul class="menu_list">
    <li.....>grilled steak</li>
    <li.....>mushrooms</li>
    <li.....>vegetables</li>
    <li.....>vegetable broth </li>
  </ul>
</li>

<li>Hot and Sour Soup
  <ul class="menu_list">
    <li class="meat".....>roasted pork</li>
    <li.....>carrots</li>
    <li.....>Chinese mushrooms</li>
    <li.....>chili</li>
    <li.....>vegetable broth </li>
  </ul>
</li>

<li>Avocado Rolls
    <ul class="menu_list">
    <li.....>avocado</li>
    <li.....>whole chiles</li>
    <li.....>sweet red peppers</li>
    <li.....>ginger sauce</li>
  </ul>
</li>
```

Найдите ингредиенты, которые должны заменяться вегетарианскими аналогами, и назначьте каждому из них соответствующий класс (fish, meat или hamburger). Если ингредиенту не нужен класс, оставьте строку пустой. Структура кода HTML соответствует внешнему виду меню на странице.

```
<li>Thai-style Halibut
  <ul class="menu_list">
    <li>coconut milk</li>
    <li class="fish">pan-fried halibut</li>
    <li>lemongrass broth</li>
    <li>vegetables</li>
    <li>Thai spices </li>
  </ul>
</li>

<li>Braised Delight
  <ul class="menu_list">
    <li class="meat">lamb shoulder</li>
    <li>cippolini onions</li>
    <li>carrots</li>
    <li>baby turnip</li>
    <li>braising jus</li>
  </ul>
</li>

<li>House Grilled Panini
  <ul class="menu_list">
    <li class="meat">prosciutto</li>
    <li>provolone</li>
    <li>avocado</li>
    <li>cherry tomatoes</li>
    <li>sourdough roll</li>
    <li>shoestring fries </li>
  </ul>
</li>

<li>House Slider
  <ul class="menu_list">
    <li>eggplant</li>
    <li>zucchini</li>
    <li class="hamburger">hamburger</li>
    <li>balsamic vinegar</li>
    <li>onion</li>
    <li>carrots</li>
    <li>multigrain roll</li>
    <li>goat cheese</li>
  </ul>
</li>

<li>Frittata
  <ul class="menu_list">
    <li class="meat">eggs</li>
    <li>Asiago cheese</li>
    <li>potatoes </li>
  </ul>
</li>

<li>Coconut Soup
  <ul class="menu_list">
    <li>coconut milk</li>
    <li class="meat">chicken</li>
    <li>vegetable broth</li>
  </ul>
</li>

<li>Soup Du Jour
  <ul class="menu_list">
    <li class="meat">grilled steak</li>
    <li>mushrooms</li>
    <li>vegetables</li>
    <li>vegetable broth </li>
  </ul>
</li>

<li>Hot and Sour Soup
  <ul class="menu_list">
    <li class="meat">roasted pork</li>
    <li>carrots</li>
    <li>Chinese mushrooms</li>
    <li>chili</li>
    <li>vegetable broth </li>
  </ul>
</li>

<li>Avocado Rolls
    <ul class="menu_list">
    <li>avocado</li>
    <li>whole chiles</li>
    <li>sweet red peppers</li>
    <li>ginger sauce</li>
  </ul>
</li>
```

Создание кнопок

Предварительная подготовка в основном завершена — пора вернуться к салфетке с требованиями шеф-повара. На следующем этапе вы должны создать две кнопки:

— Кнопка «Go Vegetarian» автоматически заменяет неподходящие ингредиенты веб-меню вегетарианскими аналогами.

— Вторая кнопка должна восстанавливать меню в исходном состоянии.

Возьми в руку карандаш

Измените разметку и сценарный код, чтобы создать две кнопки, описанные выше. Присвойте кнопке перехода на вегетарианское меню идентификатор `vegOn`, а кнопке восстановления меню — идентификатор `restoreMe`.

```
<div class="topper">
  <h2>Our Menu</h2>
    <ul>
      <li class="nav">.....</li>
      <li class="nav">.....</li>
    </ul>
</div>
```

index.html

```
$(document).ready(function() {
    var v = false;
    .....

        if (v == false){

          v = true}
    });//end button
    .....

        if (v == true){

          v = false;}
    });//end button
});//end document ready
```

my_scripts.js

Возьми в руку карандаш

Измените разметку и сценарный код, чтобы создать две кнопки, описанные выше. Присвойте кнопке перехода на вегетарианское меню идентификатор `vegOn`, а кнопке восстановления меню — идентификатор `restoreMe`.

```
<div class="topper">
  <h2>Our Menu</h2>
    <ul>
      <li class="nav"> <button id="vegOn">Go Vegetarian</button> </li>
      <li class="nav"> <button id="restoreMe">Restore Menu</button> </li>
    </ul>
</div>
```

index.html

Создание элементов button с идентификаторами vegOn и restoreMe.

```
$(document).ready(function() {
    var v = f;
    $("button#vegOn").click(function(){
        if (v == false){

          v = true;}
    });//end button
    $("button#restoreMe").click(function(){
        if (v == true){

          v = false;}
    });//end button
});//end document ready
```

В уточненный селектор входит как тип, так и идентификтор элемента.

С кнопками связывается метод click.

my_scripts.js

Что дальше?

Быстро справились! Две кнопки готовы. Вычеркнем их из списка требований и перейдем к тому, что должна делать кнопка «Go Vegetarian».

Нам нужна кнопка вегетарианского меню, которая автоматически подставляет вегетарианские аналоги в веб-меню.

Нам также понадобится вторая кнопка, восстанавливающая меню в исходном состоянии.

Замена осуществляется по следующим правилам:

— у рыбных блюд аналогов нет, они просто удаляются из меню;

— гамбургеры заменяются гигантскими шампиньонами;

— мясные ингредиенты (кроме гамбургеров) и яйца заменяются тофу.

Упражнение

Сформулируйте (своими словами) описания трех операций, которые должна выполнять кнопка перехода на вегетарианское меню.

1. .....
.....

2. .....
.....

3. .....
.....

Упражнение
Решение

Сформулируйте (своими словами) описания трех операций, которые должна выполнять кнопка перехода на вегетарианское меню.

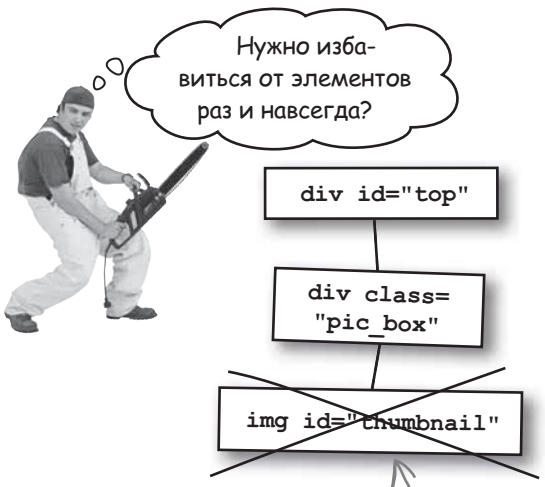
1. *Найти элементы li класса fish и удалить соответствующие блюда из меню.*
2. *Найти элементы li класса hamburger и заменить их гигантскими шампиньонами.*
3. *Найти элементы li класса meat и заменить их тофу.*

Не огорчайтесь, если ваши ответы получились немного другими. Навыки перевода требований в формальные описания операций приходят с практикой.

Начнем с первого пункта: найти элементы `li` класса `fish` и удалить их из меню. В главе 2 мы находили элементы при помощи селекторов классов и удаляли их из DOM методом `remove`.

jQuery также поддерживает метод `detach`. Оба элемента — `detach` и `remove` — удаляют элементы из DOM. Чем отличаются эти два метода и какой из них следует использовать в нашей ситуации?

remove



Метод remove удаляет элемент из DOM.

```
$("img#thumbnail").remove();
```

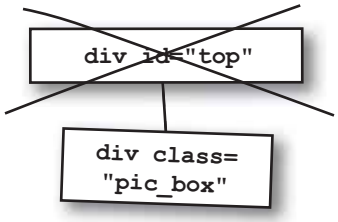
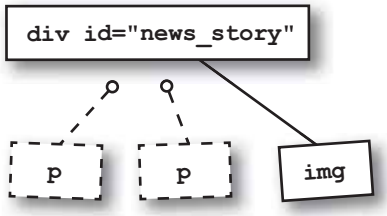
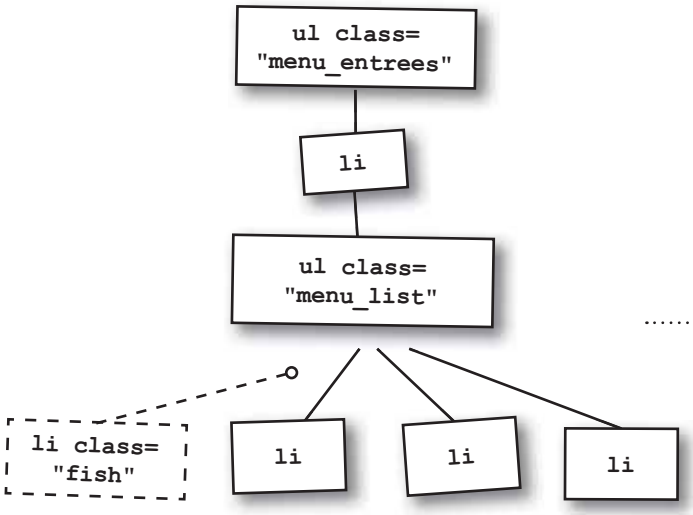
detach

Метод detach извлекает выбранный элемент(-ы) из DOM, но продолжает хранить, чтобы позднее его можно было вернуть на место.

```
$("img#thumbnail").detach();
```

Возьми в руку карандаш

Напишите справа селектор и вызов `remove` или `detach`, который приведет к показанному результату.

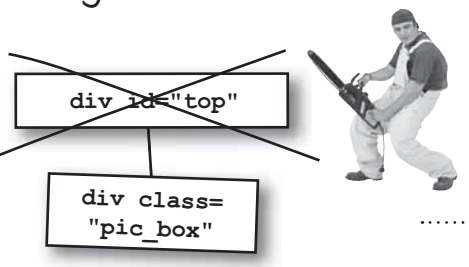
Результат в DOM	Команда jQuery
	.....
	.....
	.....

Возьми в руку карандаш
Решение

Напишите справа селектор и вызов `remove` или `detach`, который приведет к показанному результату.

Когда для элемента выполняется метод remove или detach, все его дочерние элементы также исключаются из дерева.

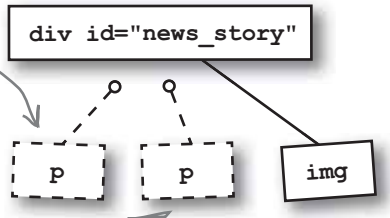
Результат в DOM



Команда jQuery

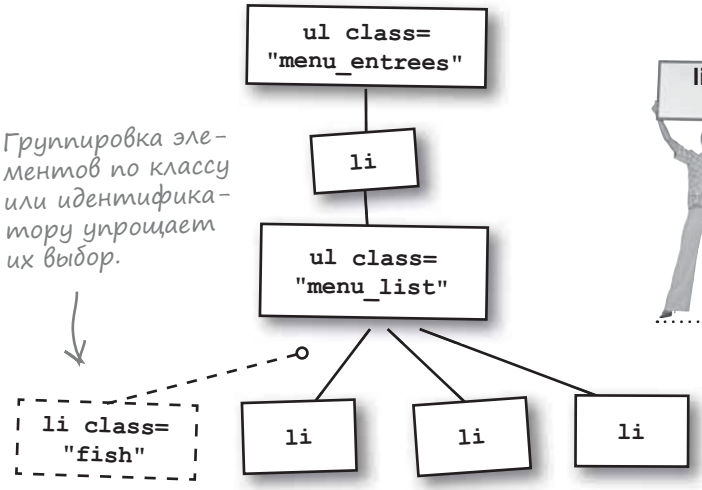
```
$("div#top").remove()
```

Не забывайте, что отсоединяются все элементы, соответствующие селектору.




```
$("div p").detach()
```

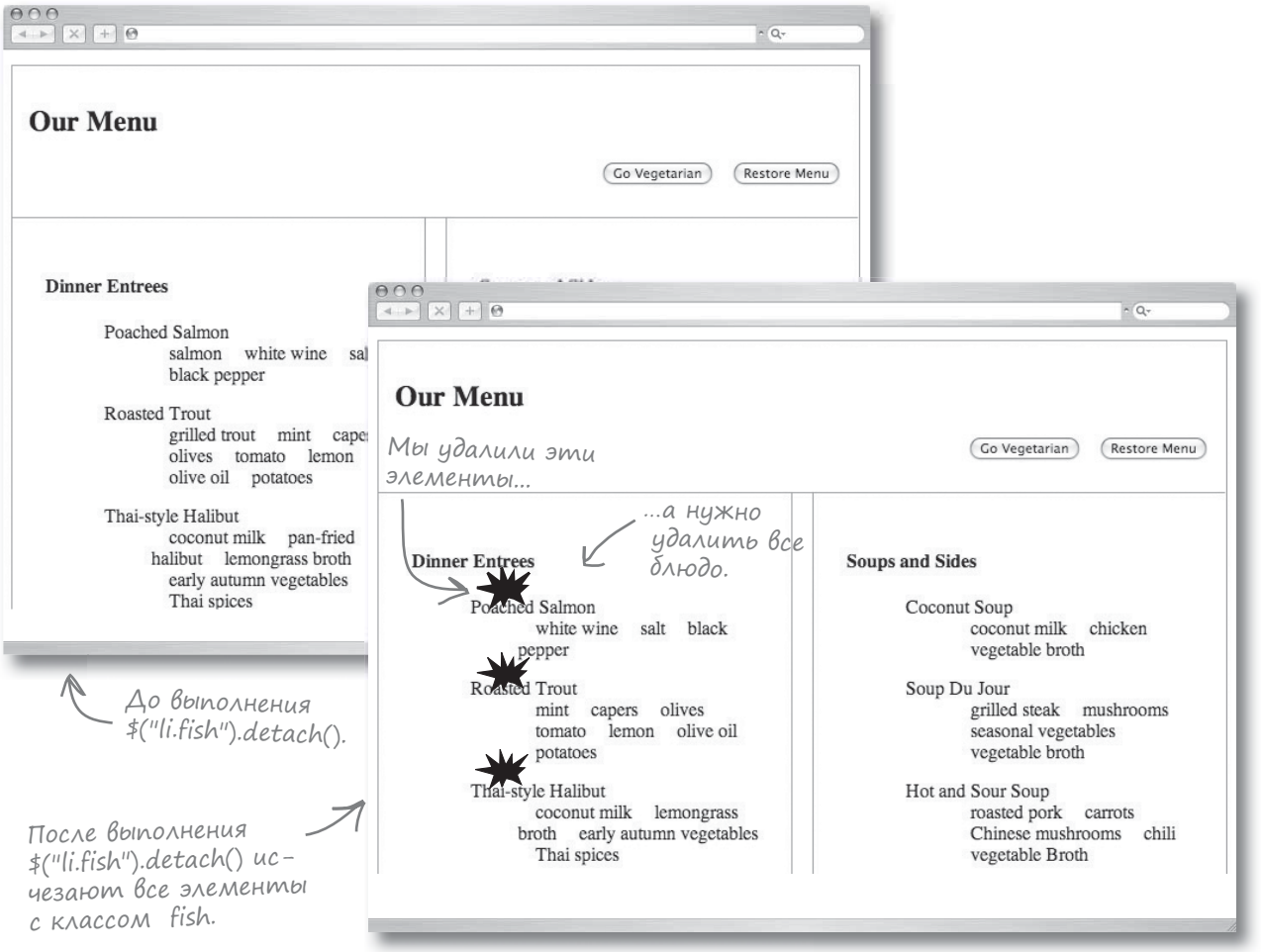
Группировка элементов по классу или идентификатору упрощает их выбор.




```
$("li.fish").detach()
```

ТЕСТ-ДРАЙВ

Включите строку из решения в функцию click кнопки `vegOn` из файла *my_scripts.js*. Откройте страницу в своем любимом браузере и убедитесь в том, что она работает правильно.



Метод `detach` определенно что-то удаляет — но не то, что мы хотели удалить. Мы удалили элементы из списка ингредиентов блюда, а нужно удалить все блюдо, имеющее вложенные элементы класса `.fish`.

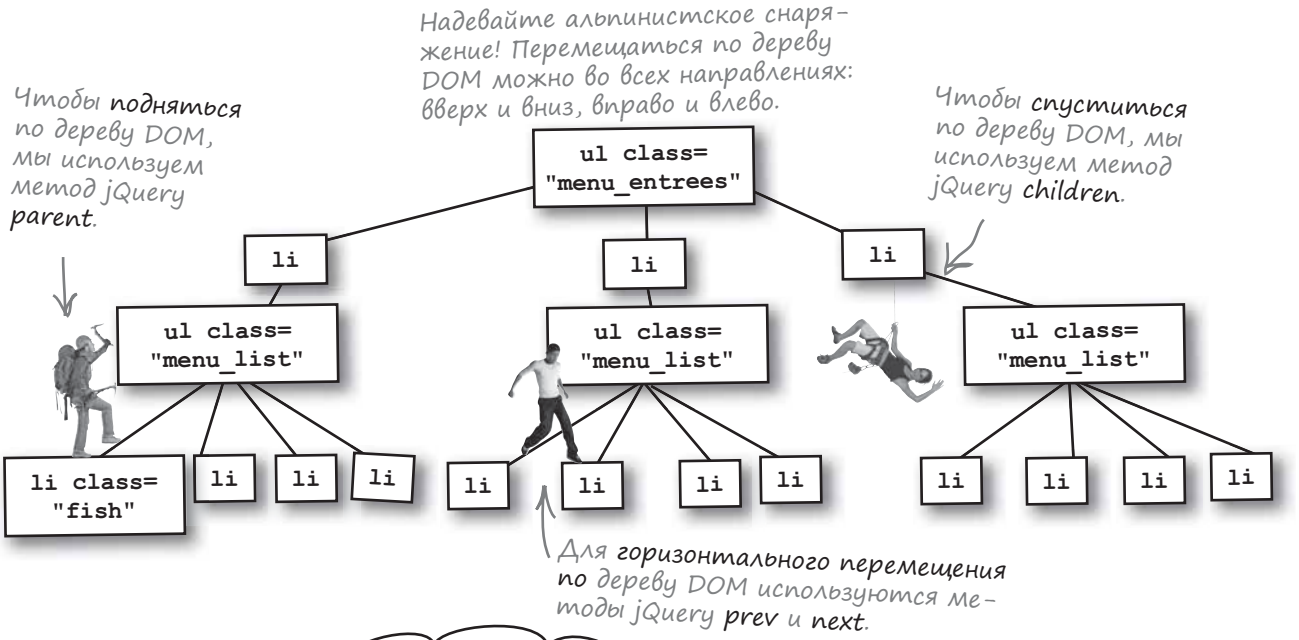
Как приказать DOM исключить из меню все блюдо?

Перемещение по дереву DOM

В главе 1 вы узнали, что модель DOM имеет древовидную структуру. У дерева DOM есть корень, ветви и узлы. Интерпретатор JavaScript, встроенный в браузер, может обходить элементы дерева DOM (и выполнять операции с ними), причем jQuery особенно хорошо подходит для этой работы. Под «обходом дерева» понимаются перемещения вверх и вниз по дереву DOM.

Мы уже выполняли различные операции с DOM, начиная с главы 1. Примером таких операций служит только что рассмотренный метод `detach` (динамическое исключение элементов из DOM).

Но какие еще возможности открывает обход дерева? Чтобы понять, как работает обход, возьмем один раздел меню и представим его в виде дерева DOM.



Лазаем по дереву DOM... Очень мило. Но как это мне поможет с удалением блюд из меню?

Методы обхода дерева DOM позволяют найти элемент и выбрать другие элементы, находящиеся выше, ниже или на одном уровне с ним.

Давайте посмотрим, как же выбираются эти элементы.

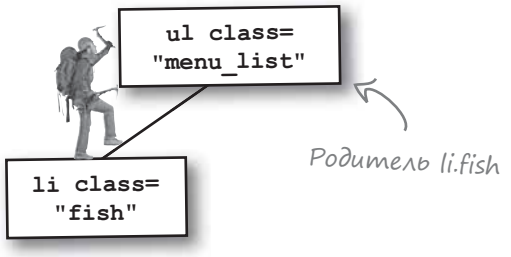
Методы обхода дерева DOM

Чтобы объяснить DOM, что мы хотим удалить блюда, в которые входят элементы класса `fish`, необходимо идентифицировать эти элементы по связи между ними. Методы обхода дерева jQuery предоставляют такую возможность.

Выбираем все элементы класса fish.

Затем переходим к элементу, который находится над этими элементами.

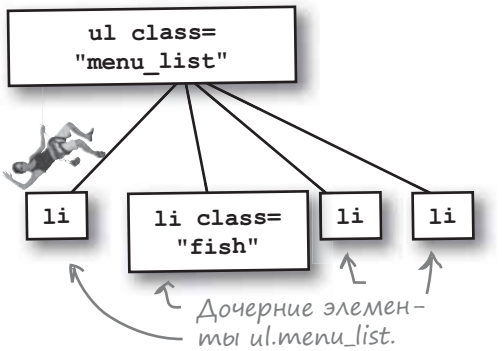
```
$(".fish").parent()
```



Выбираем все элементы класса menu_list.

Затем переходим к элементам, находящимся под этими элементами.

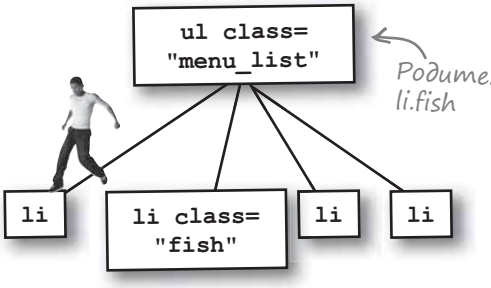
```
$(".menu_list").children()
```



Выбираем все элементы класса fish.

Затем переходим к следующему элементу с тем же родителем, находящимся слева.

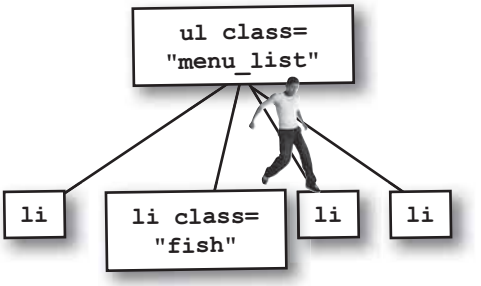
```
$(".fish").prev()
```



Выбираем все элементы класса fish.

Затем переходим к следующему элементу с тем же родителем, находящимся справа.

```
$(".fish").next()
```



МОЗГОВОЙ ШТУРМ

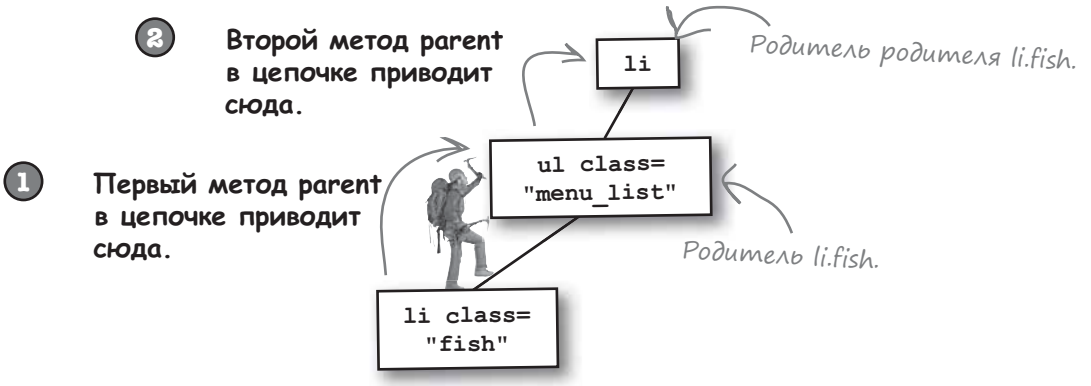
Какой из этих методов поможет убрать из меню блюда, содержащие элементы класса `fish`?

Сцепленные вызовы методов

А если потребуется переместиться еще выше, ниже или глубже? Используйте *сцепление* методов — этот механизм повышает эффективность перемещения по страницам и выполнения операций с ними. Вот как он работает:

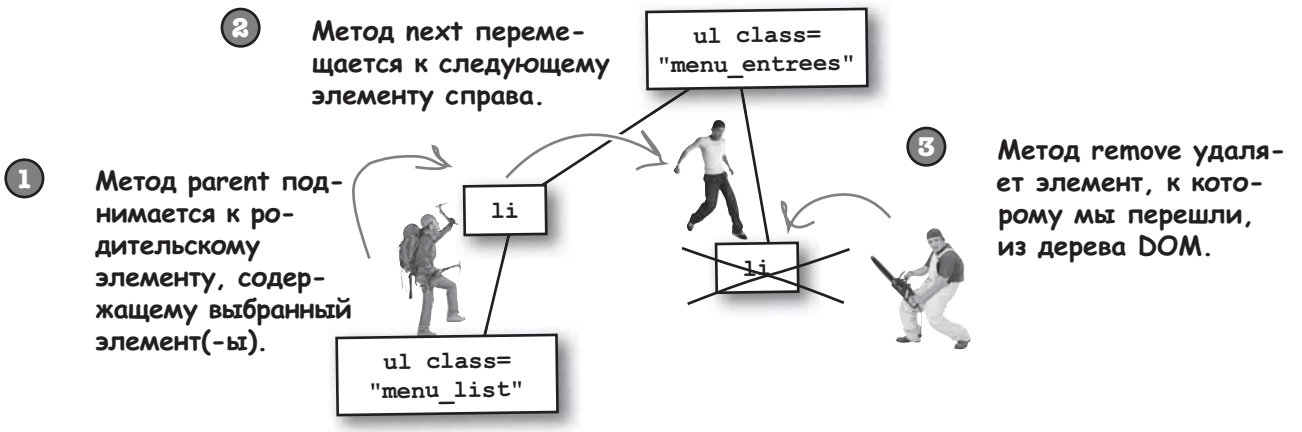
Чтобы подняться еще на один уровень наверх, включите еще один метод в цепочку.

```
$(".fish").parent().parent()
```



Также цепочка может состоять из разных методов.

```
$(".menu_list").parent().next().remove()
```



Упражнение

Загрузите страницу *http://www.thinkjquery.com/chapter04/traversal/* и откройте консоль JavaScript в средствах разработчика вашего браузера. О средствах разработчика разных браузеров рассказано в разделе «Примите к сведению» в начале книги. Выполните каждый из методов обхода со сцепленным методом `detach`, как показано ниже. Запишите, почему этот вызов поможет (или не поможет) в решении нашей задачи.

Важно: Не забывайте ***обновлять страницу*** после выполнения каждой команды.

```
$(".menu_entrees").children().detach()
```

.....

.....

.....

```
$(".menu_list").children().detach()
```

.....

.....

.....

```
$(".fish").parent().detach()
```

.....

.....

.....

```
$(".fish").parent().parent().detach()
```

.....

.....

.....

Упражнение
Решение

Загрузите страницу *http://www.thinkjquery.com/chapter04/traversal/* и откройте консоль JavaScript в средствах разработчика вашего браузера. О средствах разработчика разных браузеров рассказано в разделе «Примите к сведению» в начале книги. Выполните каждый из методов обхода со сцепленным методом `detach`, как показано ниже. Запишите, почему этот вызов поможет (или не поможет) в решении нашей задачи.

Важно: Не забывайте ***обновлять страницу*** после выполнения каждой команды.

```
$(".menu_entrees").children().detach()
```

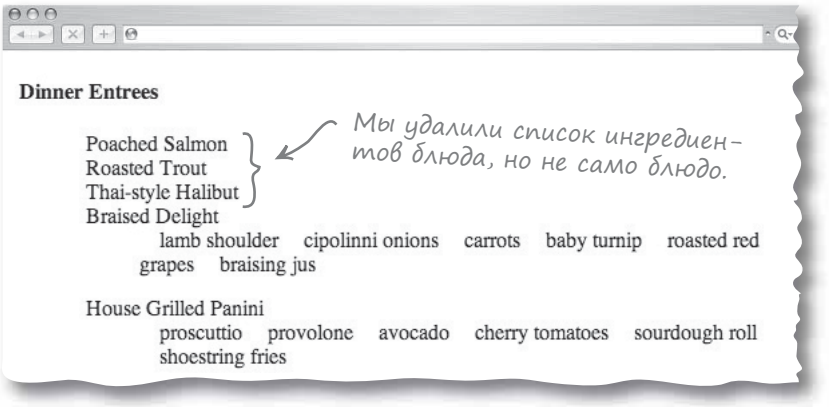
Этот вызов отсоединяет элементы, дочерние по отношению к .menu_entrees. Он не подойдет для удаления блюд, содержащих рыбу, потому что он удаляет ВСЕ блюда в списке. Не то, что нам нужно...


```
$(".menu_list").children().detach()
```

Этот вызов отсоединяет элементы, дочерние по отношению к .menu_list. Он не подойдет для удаления блюд, содержащих рыбу, потому что он удаляет ингредиенты из всех элементов ul.menu_list. Опять не то...

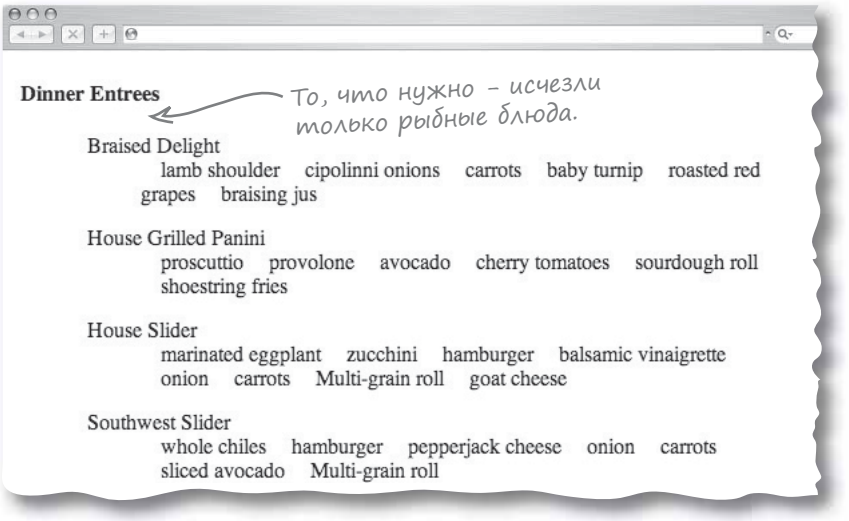
```
$(".fish").parent().detach()
```

Этот вызов отсоединяет родительский элемент .fish. Он не подойдет для удаления блюд, содержащих рыбу, потому что он недостаточно высоко поднимается по дереву DOM. Вместо этого он удаляет элемент ul.menu_list (и все, что находится под ним).




```
$(".fish").parent().parent().detach()
```

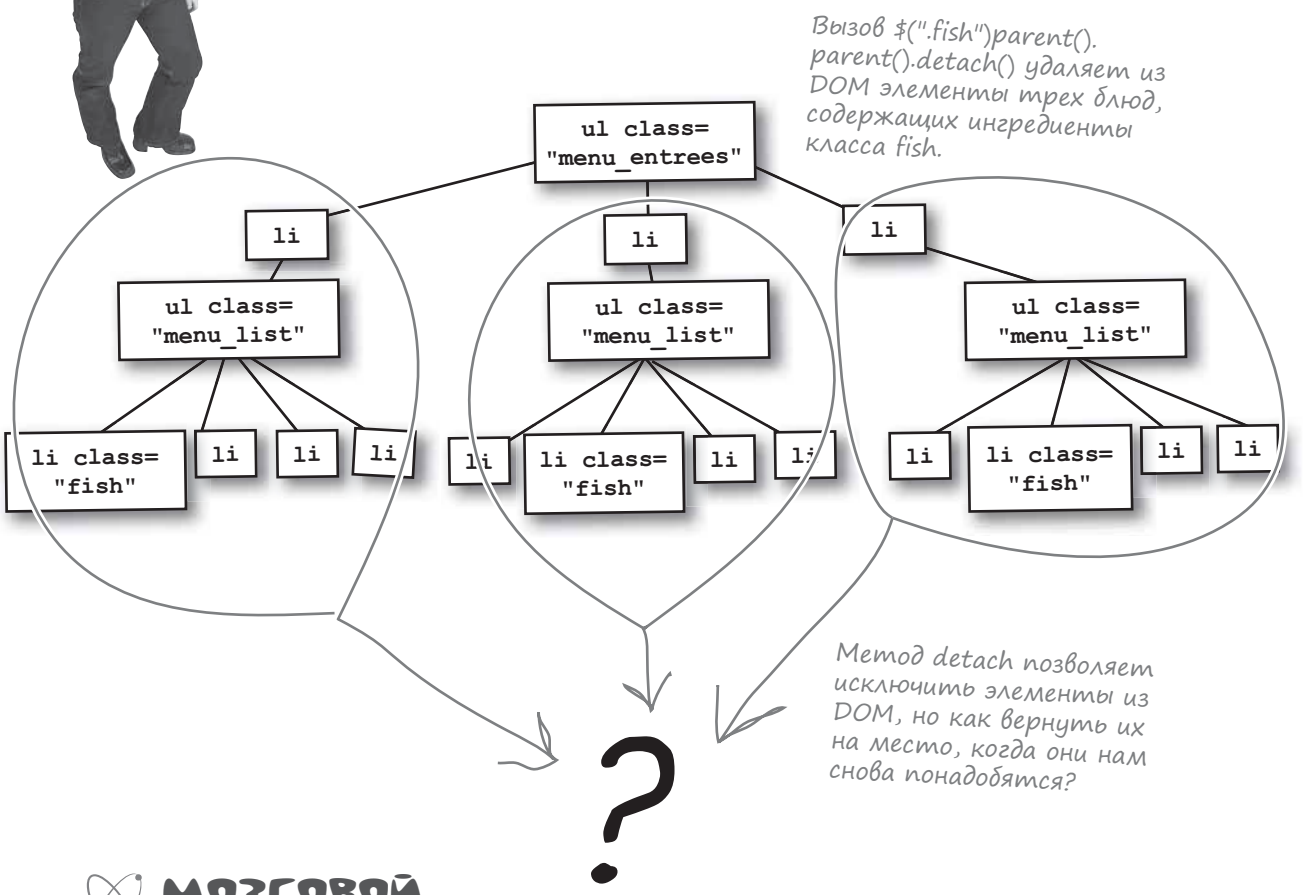
Этот вызов отсоединяет родительский элемент .fish. Он делает именно то, что нам нужно.



Постойте-ка, разве мы не должны восстановить рыбные блюда в меню при программировании второй кнопки?

Верно. Мы не можем просто отсоединить элементы рыбных блюд и забыть о них.

Чтобы восстановление заработало, нам придется слегка доработать наш код.



МОЗГОВОЙ ШТУРМ

Рыбные блюда должны вернуться в дерево DOM. Что для этого с ними нужно сделать?

Возьми в руку карандаш

Мы повидали уже немало конструкций jQuery и JavaScript. Какие из них нам пригодятся для восстановления элементов класса `.fish`? Запишите для каждой конструкции в столбце «Нужно ли использовать?» свой ответ «Да» или «Нет» и объясните, почему вы выбрали (или не выбрали) эту конструкцию. Один ответ мы уже записали, за вами еще три.

	Нужно ли использовать?	Почему?
Завершитель	*Нет*	*Завершитель — простой признак конца команды. Он не имеет отношения к восстановлению отсоединенных элементов.*
Переменная		
Функция		
Селектор		

Возьми в руку карандаш
Решение

Мы повидали уже немало конструкций jQuery и JavaScript. Какие из них нам пригодятся для восстановления элементов класса `.fish`? Запишите для каждой конструкции в столбце «Нужно ли использовать?» свой ответ «Да» или «Нет» и объясните, почему вы выбрали (или не выбрали) эту конструкцию. Ниже приведен наш ответ.

	Нужно ли использовать	Почему?
Завершитель	*Нет*	*Завершитель — простой признак конца команды. Он не имеет отношения к восстановлению отсоединенных элементов.*
Переменная	*Да*	*Переменные используются для хранения информации. Если сохранить отсоединенные элементы в переменной, мы сможем их вернуть в дерево DOM, просто сославшись на переменную.*
Функция	*Нет*	*Функции выполняют операции с данными. Задача восстановления отсоединенных элементов относится к хранению данных, а не к их обработке.*
Селектор	*Нет*	*Селектор выбирает элементы DOM. В нашем случае элементы уже выбраны, а нам нужен способ их сохранения.*

Часто Задаваемые Вопросы

В: А если я захочу избавиться от данных, хранящихся в элементе, но чтобы сам элемент остался на месте?

О: Для удаления содержимого элемента можно воспользоваться методом `empty`. Например, для удаления всего текста из абзацев страницы достаточно выполнить следующую команду: `$("p").empty();`.

В: Можно ли обойти всех родителей элемента?

О: Да. Кроме метода `parent`, jQuery также поддерживает метод `parents`, обходящий всех родителей выбранного элемента. Позднее в этой главе будет приведен пример использования `parents`.

В: А если я хочу найти родительский элемент, ближайший к выбранному?

О: Воспользуйтесь методом `closest`. Как и метод `parents`, метод `closest` поднимается вверх по цепочке родителей, но останавливается при первом совпадении. Например, если вы хотите найти ближайший элемент `ul` над пунктом списка, используйте команду: `$("li").closest("ul")`.

В: Я уже знаю методы `next` и `previous`, а если мне потребуется обойти все одноуровневые элементы дерева DOM?

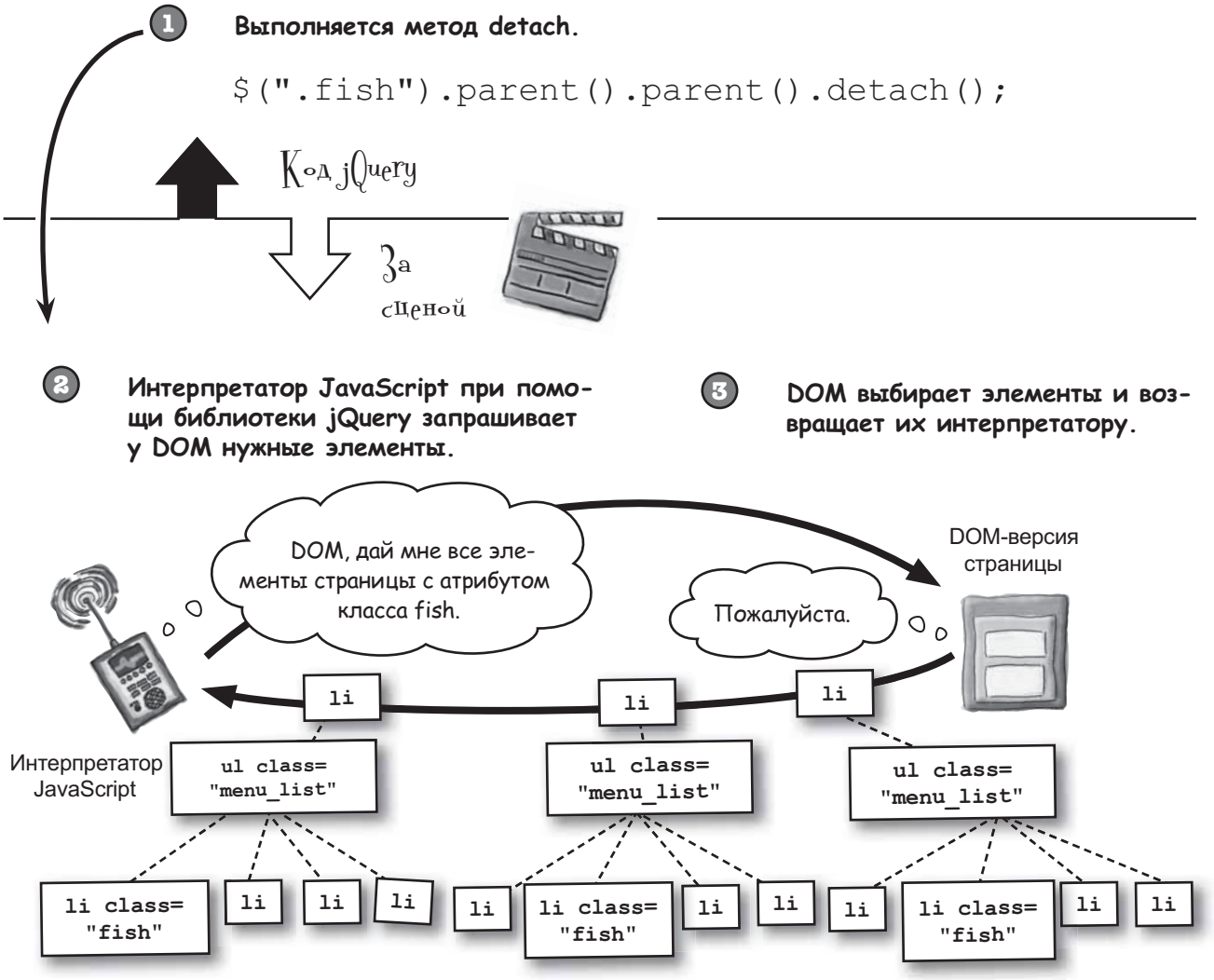
О: К счастью, создатели jQuery предусмотрели и такую возможность. Метод `siblings` обходит все элементы одного уровня с выбранным элементом.

В: Браузер Google Chrome содержит встроенную поддержку jQuery?

О: Нет. Мы можем выполнять код jQuery в средствах разработчика браузера Chrome только потому, что мы включили jQuery в страницу HTML. Если в открытой вебстранице не используется jQuery, вы не сможете выполнять команды jQuery в консоли JavaScript Chrome.

В переменных также могут храниться элементы

Переменные — штука весьма полезная; вот и сейчас они нам пригодятся. Вы уже встречались с переменными в первых трех главах, но тогда они использовались для хранения чисел и текстовых строк. А разве не удобно было бы хранить в переменных еще и элементы DOM? Оказывается, такая возможность существует.



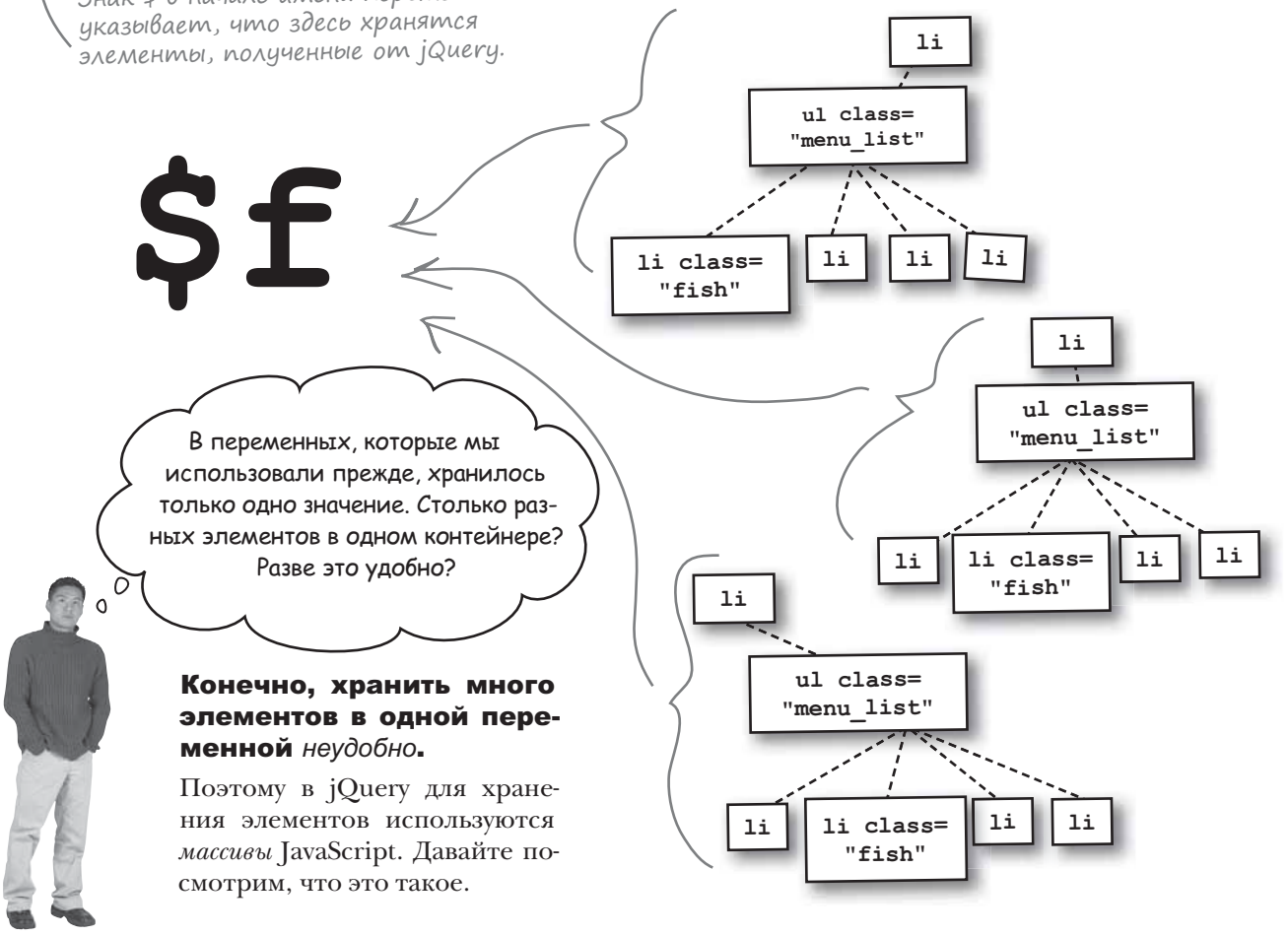
Браузер временно сохраняет полученные элементы в памяти. Если мы захотим воспользоваться ими позднее в своем коде, их нужно сохранить в переменной. Но как это сделать?

И снова знак $...

Сохранить элементы совсем не сложно. Создайте переменную (как для обычных числовых и текстовых данных) и присвойте ей (при помощи знака =) результат команды, возвращающей элементы. Но разве не полезно было бы с первого взгляда видеть, что в переменной хранятся специальные данные — например элементы (вместо чисел и строк)? У программистов jQuery принято начинать имена переменных, предназначенных для хранения возвращаемых jQuery элементов, со знака $. Тогда любой программист, который будет читать ваш код, сразу поймет, что в переменной хранятся данные, полученные от jQuery.

```
$f = $(".fish").parent().parent().detach();
```

Знак $ в начале имени переменной указывает, что здесь хранятся элементы, полученные от jQuery.



В переменных, которые мы использовали прежде, хранилось только одно значение. Столько разных элементов в одном контейнере? Разве это удобно?

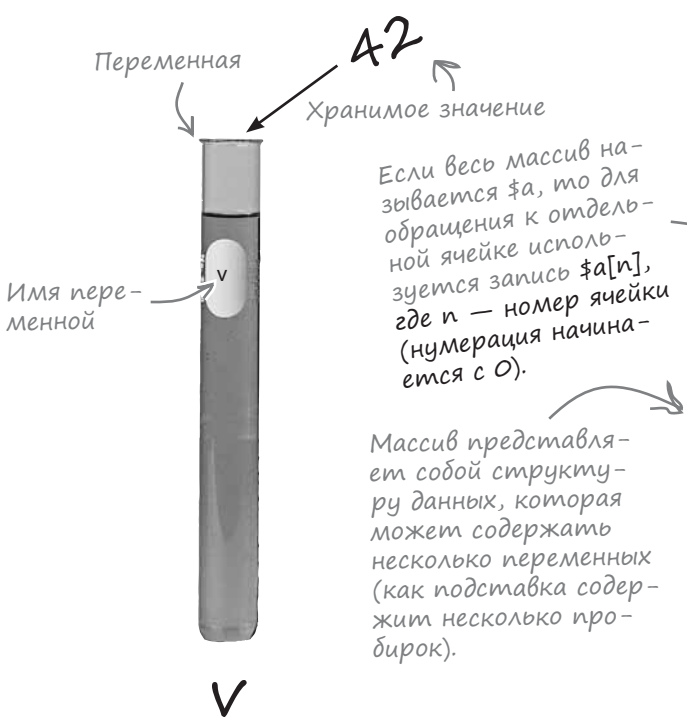
Конечно, хранить много элементов в одной переменной *неудобно*.

Поэтому в jQuery для хранения элементов используются *массивы* JavaScript. Давайте посмотрим, что это такое.

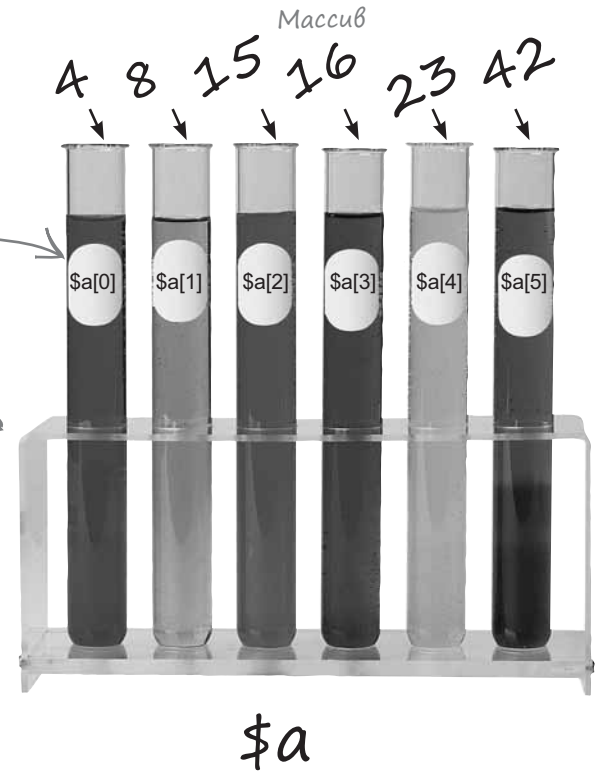
Хранение данных в массивах

Каждый раз, когда мы выбираем элементы DOM и сохраняем их в переменной, jQuery возвращает данные в виде *массива*. В сущности, массив — это переменная с расширенными возможностями хранения данных.

В простой переменной хранится одно значение.



В массиве хранится много значений.



Если весь массив называется $a, то для обращения к отдельной ячейке используется запись $a[n], где n — номер ячейки (нумерация начинается с 0).

Массив представляет собой структуру данных, которая может содержать несколько переменных (как подставка содержит несколько пробирок).

Вы можете сохранить и извлечь данные из любой ячейки. Например, для сохранения значения «15» в третьей ячейке используется следующая команда:

```
$a[2] = 15;
```

Третья ячейка имеет номер 2, потому что нумерация начинается с 0.

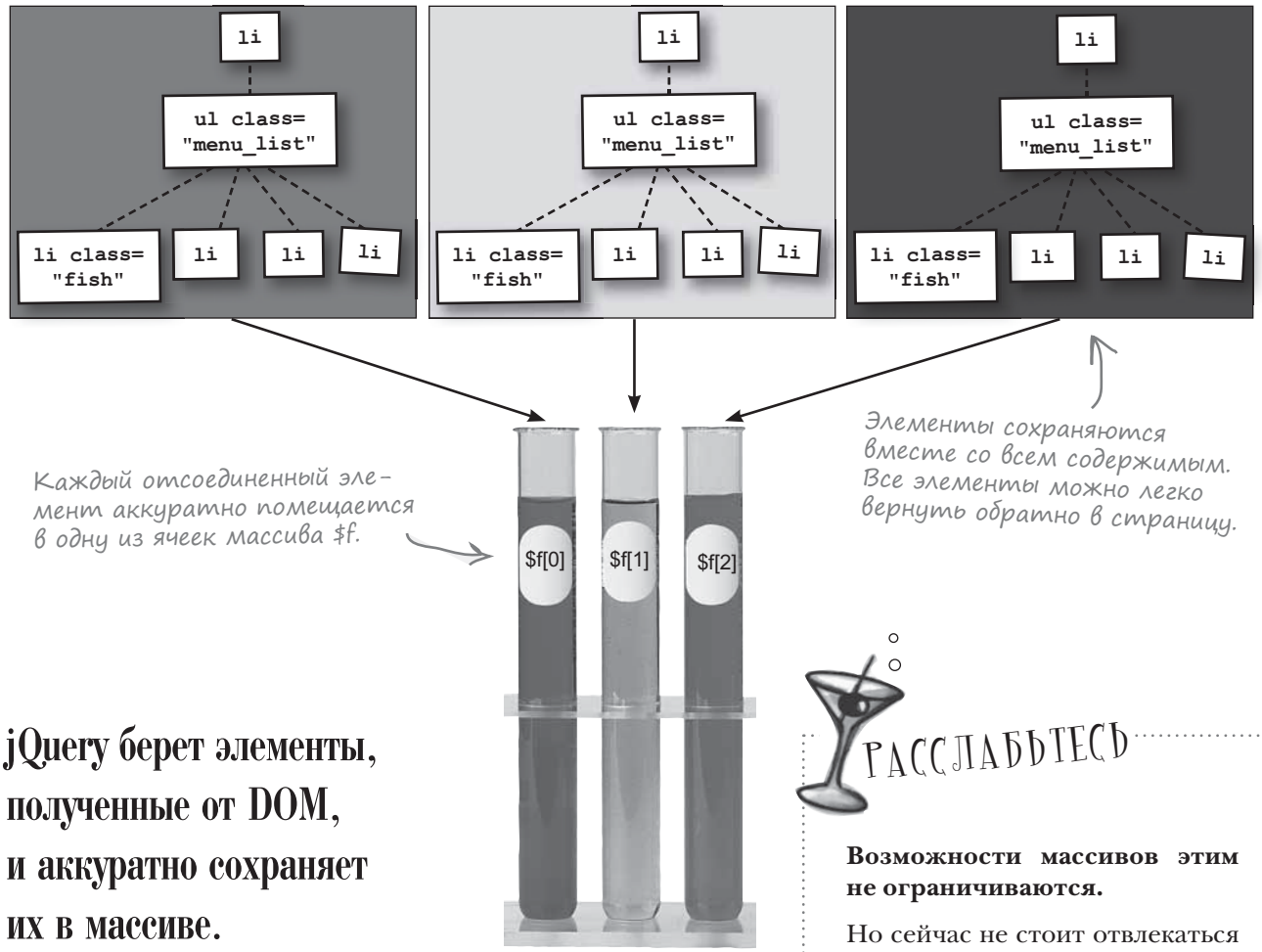
Для любознательных

Имена массивов **необязательно** начинать со знака доллара ($). Обозначение массивов jQuery знаком $ — всего лишь распространенное правило, принятое среди разработчиков jQuery.

Хранение элементов в массиве

Когда мы выбираем и отсоединяем элементы `li` и присваиваем результат операции переменной (`$f`), jQuery берет элементы, полученные от DOM, и аккуратно сохраняет их в массиве JavaScript. А когда позднее потребуется вернуть элементы обратно в массив, эта задача будет решаться намного проще.

```
$f = $(".fish").parent().parent().detach();
```



jQuery берет элементы, полученные от DOM, и аккуратно сохраняет их в массиве.

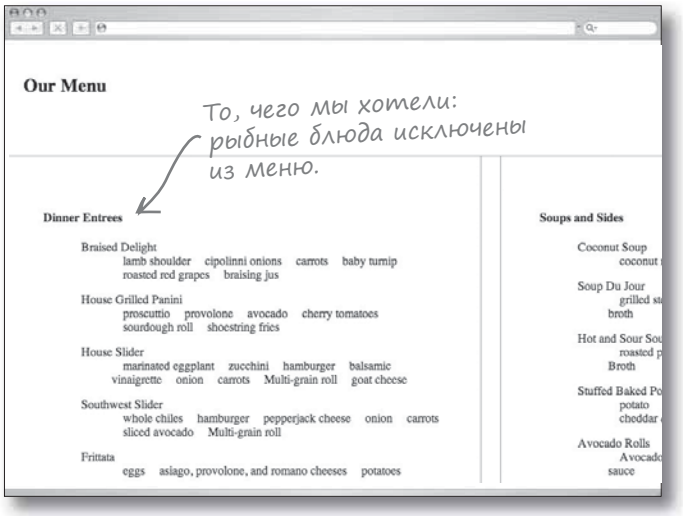
РАССЛАБЬТЕСЬ

Возможности массивов этим не ограничиваются.

Но сейчас не стоит отвлекаться от основной темы. Массивы намного подробнее рассматриваются в главе 6.

ТЕСТ-ДРАЙВ

Включите приведенную на предыдущей странице строку кода, которая отсоединяет родителей элементов `#fish`, в функцию-обработчик `click` кнопки `vegOn` в файле *my_scripts.js*. Затем откройте страницу в своем любимом браузере и убедитесь в том, что все правильно работает.



Получилось! Давайте сверимся со списком.

- [x] 1. *Найти элементы li класса fish и удалить соответствующие блюда из меню.*
- [ ] 2. *Найти элементы li класса hamburger и заменить их гигантскими шампиньонами.*
- [ ] 3. *Найти элементы li класса meat и заменить их тофу.*

Итак, следующая задача — найти блюда, в состав которых входят гамбургеры, и заменить гамбургеры гигантскими шампиньонами.

МОЗГОВОЙ ШТУРМ

Мы уже умеем исключать элементы из DOM, но как динамически заменить содержимое DOM другим содержимым?

Изменение элементов методом replaceWith

Метод `replaceWith` позволяет заменить выбранные элементы новыми элементами. Когда вам потребуется провести подобную замену в DOM, используйте этот удобный метод jQuery. Допустим, вы хотите динамически заменить элемент заголовка 2-го уровня с текстом «Our Menu» заголовком 1-го уровня с текстом «My Menu». Вот как это делается методом `replaceWith`:

Выбираем все элементы h2.

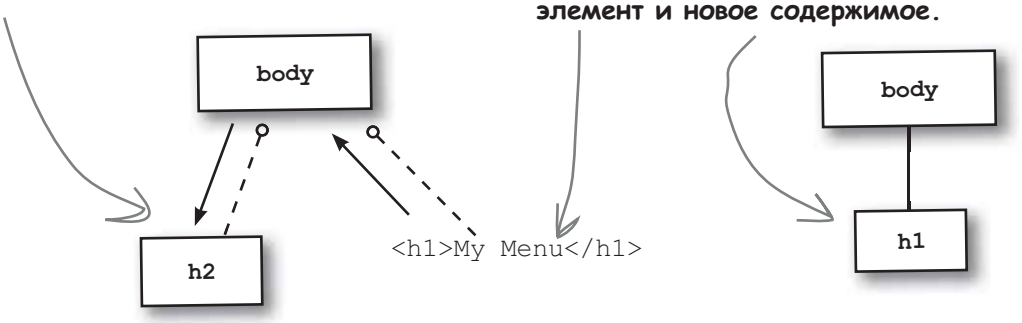
Заменяем выбранные элементы...

...содержимым, заключенным в скобки.

```
$("h2").replaceWith("<h1>My Menu</h1>");
```

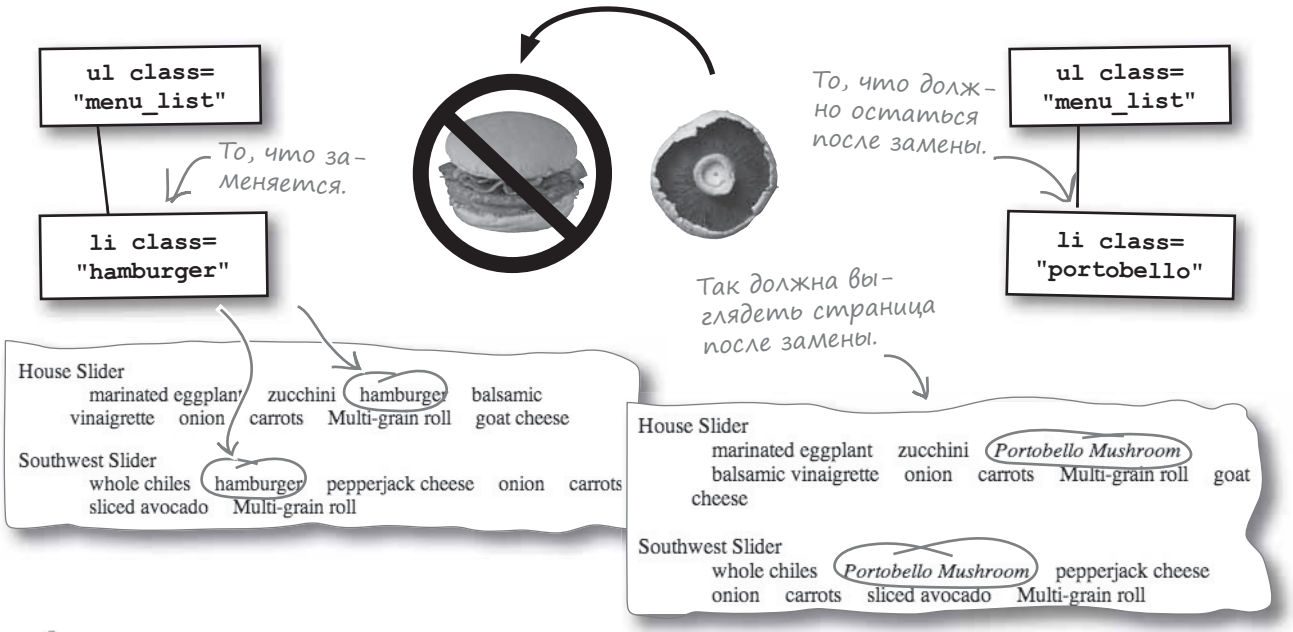
1 **Интерпретатор JS находит элемент заголовка 2-го уровня...**

2 **...и заменяет его содержимым в круглых скобках. В модели DOM появляется новый элемент и новое содержимое.**



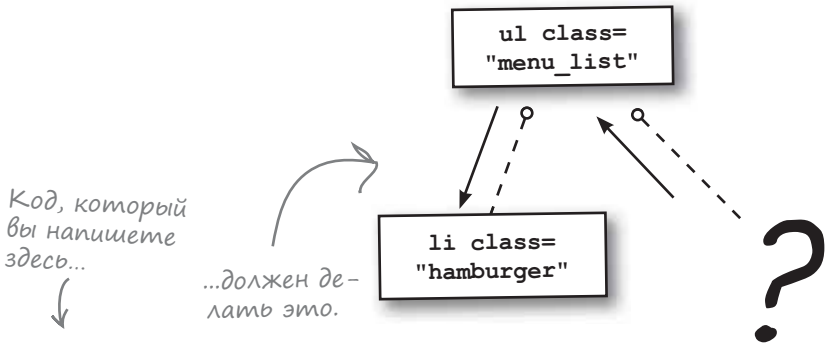
Чем поможет replaceWith?

Итак, мы должны найти элементы `li` класса `hamburger` и заменить их элементами `li` класса `portobello`. Но прежде чем браться за программирование, стоит лишний раз подумать над задачей.



Упражнение

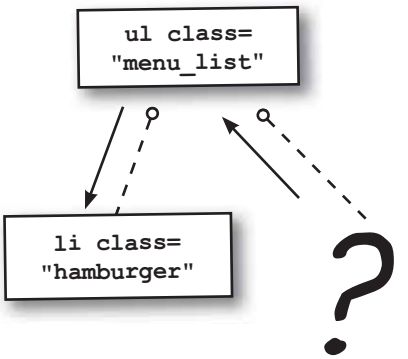
Напишите код, который находит элементы `li` класса `hamburger` и заменяет их элементами `li` класса `portobello`. Следующая схема поможет вам лучше представить суть задачи. Мы уже написали часть решения за вас; допишите остальное.




```
$(              ).replaceWith(                              <em>Portobello Mushroom</em>       );
```

Упражнение
Решение

Напишите код, который находит элементы `li` класса `hamburger` и заменяет их элементами `li` класса `portobello`. Следующая схема поможет вам лучше понять суть задачи. Вот как выглядит наше решение:



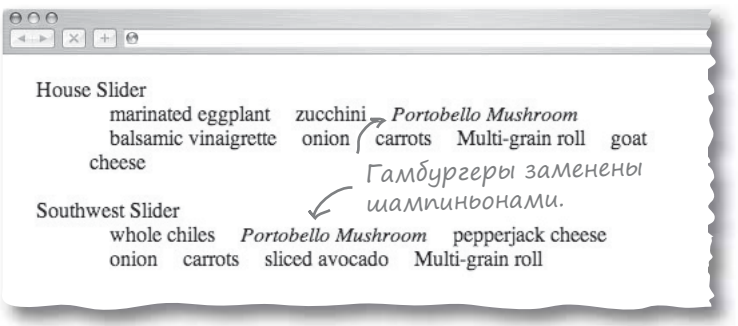
Выбираем все элементы класса hamburger.

Метод replaceWith динамически заменяет выбранное содержимое элементом в круглых скобках. Главное — помнить, что круглые скобки могут содержать код HTML.

```
$(".hamburger").replaceWith("<li class='portobello'> <em>Portobello Mushroom</em></li>");
```

ТЕСТ-ДРАЙВ

Включите вызов `replaceWith` в функцию `click` кнопки `vegOn` в файле *my_scripts.js*. Откройте страницу в своем любимом браузере и нажмите кнопку «Go Vegetarian». Убедитесь в том, что все работает правильно.

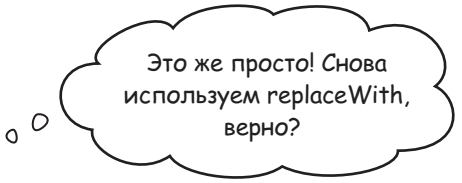


Не торопитесь с replaceWith

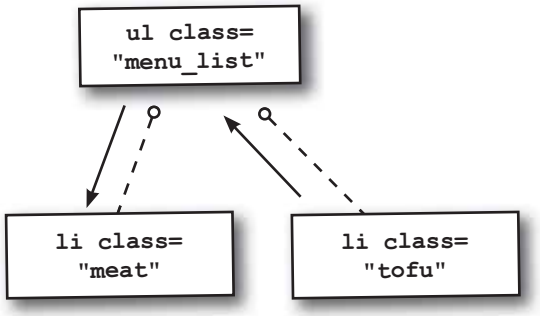
Что там следующее в нашем списке?

- [x] 1. *Найти элементы li класса fish и удалить соответствующие блюда из меню.*
- [x] 2. *Найти элементы li класса hamburger и заменить их гигантскими шампиньонами.*
- [ ] 3. *Найти элементы li класса meat и заменить их тофу.*

Нужно найти элементы класса `meat` и заменить их элементами класса `tofu`.



Вообще-то на этот раз `replaceWith` не подойдет.



МОЗГОВОЙ ШТУРМ

Метод jQuery `replaceWith` — простой и мощный инструмент. Но, к сожалению, для решения этой задачи он не подходит. Почему?

Когда replaceWith не подходит

Метод `replaceWith` хорошо подходит для выполнения замены типа «один к одному» — например, замены класса `hamburger` классом `portobello`.

Замена «один ко многим»

Однако сценарий замены в следующем пункте нашего списка задач не относится к типу «один к одному». Мы должны заменить много *разных* видов ингредиентов (индейка, яйца, говядина, отбивные и т. д.) одним ингредиентом (тофу).

Да, мы можем взять элементы класса meat и заменить их все на tofu.

Замена «многие к одному»

Но когда позднее потребуется выбрать тофу и восстановить старыми элементами, возникает проблема. Модель DOM о них уже не помнит!

Позднее DOM уже ничего не помнит о разных типах мясных ингредиентов.

Тофу можно заменить одним типом мясных ингредиентов, но это совсем не то, что нам нужно.

Таким образом, замену придется выполнять в два этапа.

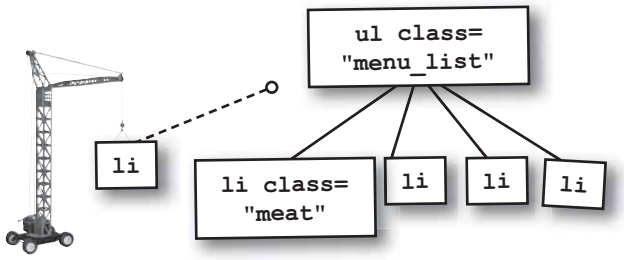
1. **Вставить элементы li класса tofu в DOM после элементов meat.**
2. **Отсоединить элементы класса meat и сохранить их в переменной.**

Вставка HTML в DOM

До настоящего момента мы занимались либо удалением, либо заменой элементов DOM. К счастью, создатели библиотеки jQuery также предоставили различные средства вставки в DOM. Мы рассмотрим два метода вставки: `before` и `after`.

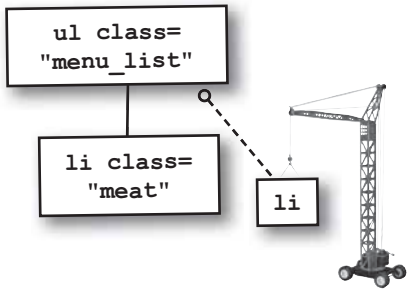
Метод `before` вставляет содержимое перед выбранным элементом:

```
$(".meat").before("<li>Tofu</li>");
```



Метод `after` вставляет содержимое после выбранного элемента.

```
$(".meat").after("<li>Tofu</li>");
```



Возьми в руку карандаш

Напишите код jQuery, реализующий два этапа нашего решения.

1. **Вставить элементы li класса tofu в DOM после элементов meat.**

   .....

2. **Отсоединить элементы класса meat и сохранить их в переменной.**

   .....

Возьми в руку карандаш Решение

Напишите код jQuery, реализующий два этапа нашего решения.

1. **Вставить элементы li класса tofu в DOM после элементов meat.**

```
$(".meat").after("<li class='tofu'><em>Tofu</em></li>");
```

2. **Отсоединить элементы класса meat и сохранить их в переменной.**

```
$m = $(".meat").detach();
```

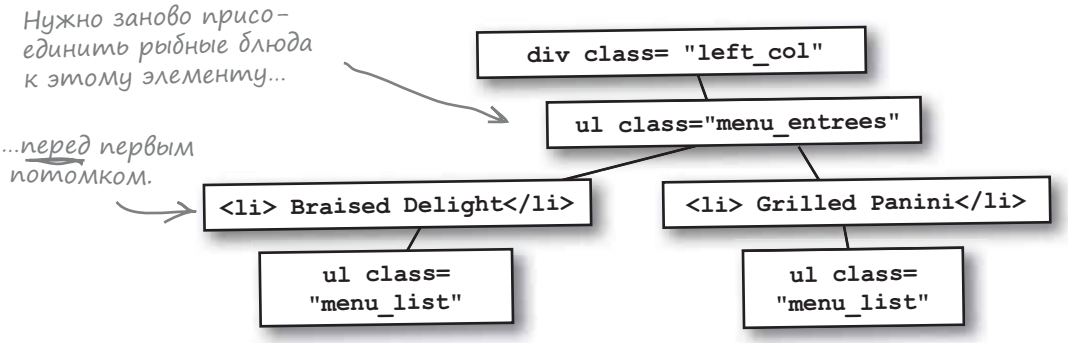
Для кнопки переключения на вегетарианское меню выполнены все пункты списка.

- [x] 1. *Найти элементы li класса fish и удалить соответствующие блюда из меню.*
- [x] 2. *Найти элементы li класса hamburger и заменить их гигантскими шампиньонами.*
- [x] 3. *Найти элементы li класса meat и заменить их тофу.*

Пора переходить к кнопке восстановления меню. Список задач для этой кнопки выглядит так:

- [ ] вернуть рыбные блюда в меню (например, перед первым элементом меню в левом столбце);
- [ ] найти шампиньоны в ингредиентах и заменить их гамбургерами;
- [ ] найти тофу в ингредиентах и заменить его различными видами мясных ингредиентов (в правильном порядке).

Что же необходимо сделать для решения первой задачи?



С использованием `before` все понятно, но как задать *первого* потомка?

Фильтры (часть 1)

К счастью, в jQuery существуют фильтрующие методы (или просто фильтры), которые позволяют сократить выбранное множество элементов для решения таких задач, как поиск первого потомка. Мы рассмотрим шесть фильтров (три на этой странице, три на следующей).

first

Метод `first` исключает из выбранного множества *все элементы, кроме первого.*

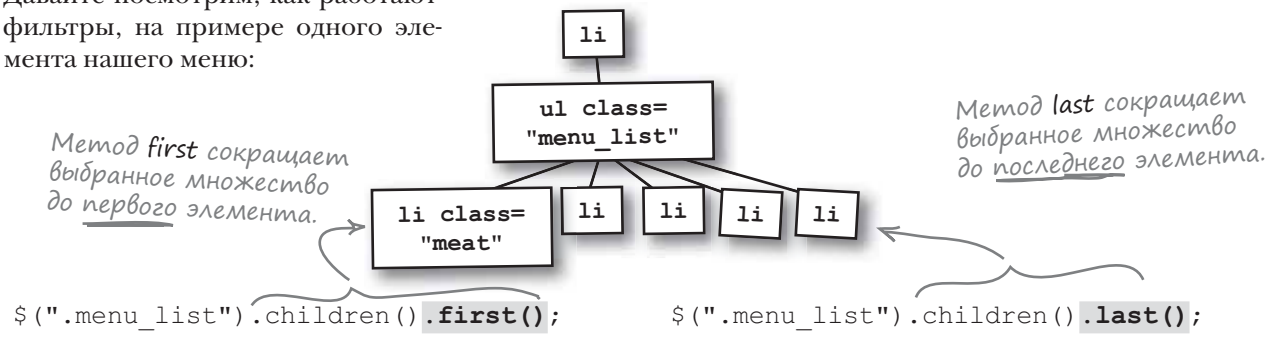
eq

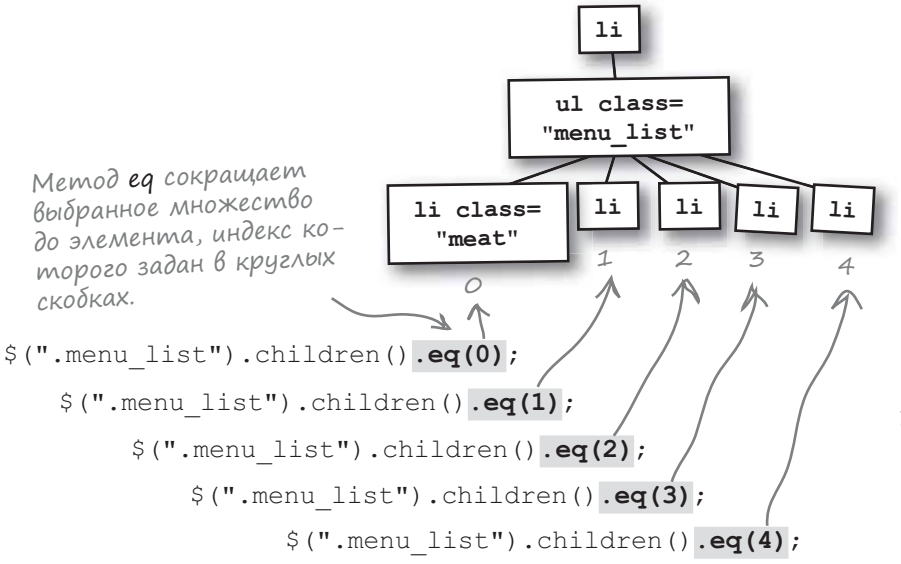
Метод `eq` исключает из выбранного множества *все элементы, кроме элемента, индекс которого равен* числу в круглых скобках.

last

Метод `last` исключает из выбранного множества *все элементы, кроме последнего.*

Давайте посмотрим, как работают фильтры, на примере одного элемента нашего меню:





Вспомните, что jQuery возвращает выбранные элементы в виде массива. Индекс, передаваемый методу eq, соответствует ячейке элемента в массиве.

Фильтры (часть 2)

Теперь посмотрим, что собой представляют методы `slice`, `filter` и `not`.

slice

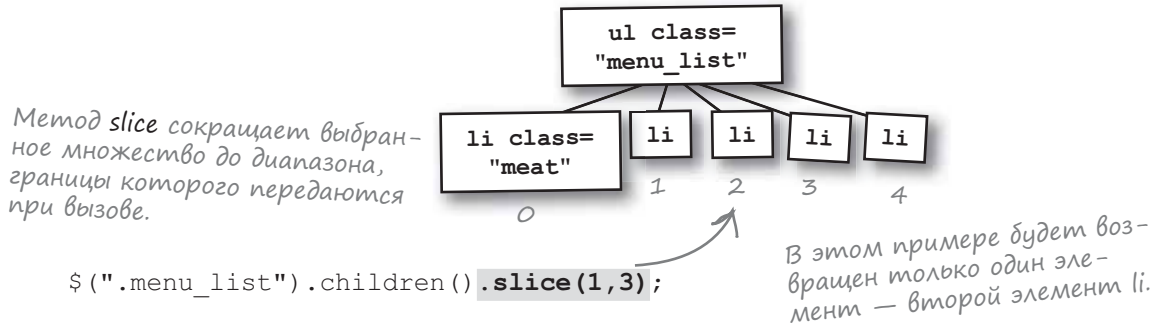
Метод `slice` оставляет в выбранном множестве только те элементы, индексы которых *лежат в диапазоне*, заданном в круглых скобках.

filter

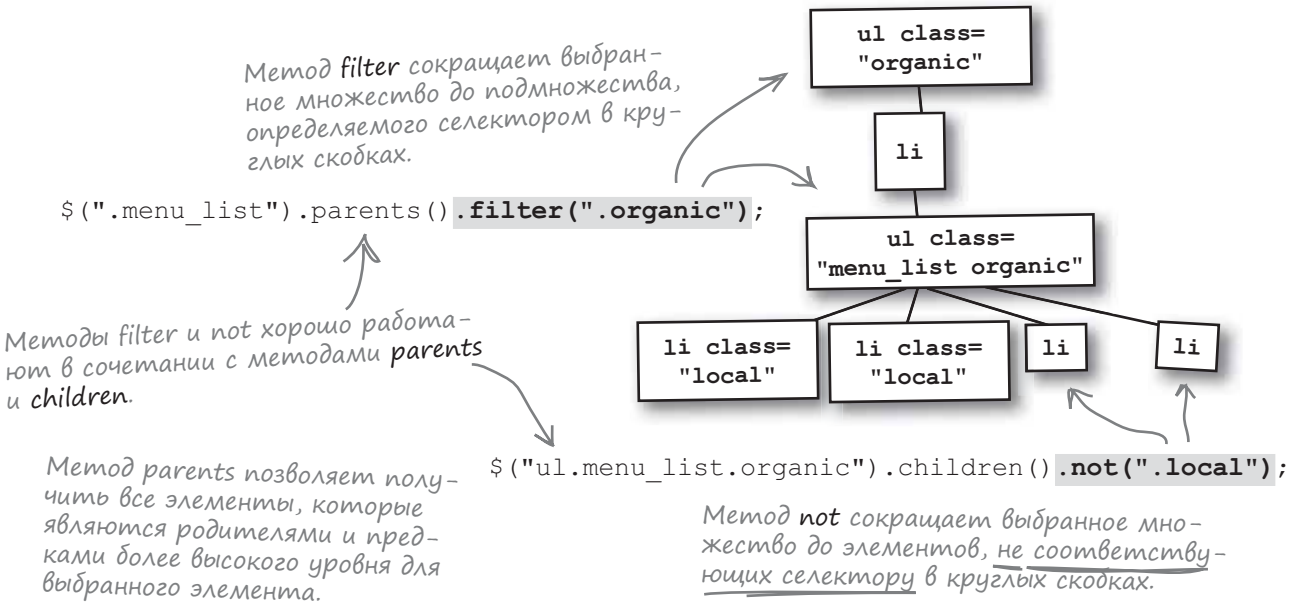
Метод `filter` оставляет в выбранном множестве только те элементы, которые *соответствуют селектору* в круглых скобках.

not

Метод `not` исключает из выбранного множества все элементы, *не соответствующие селектору* в круглых скобках.

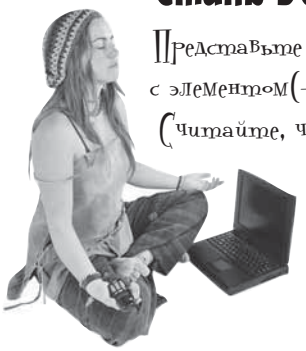


Методы `filter` и `not` позволяют использовать селекторы для определения подмножеств выбранного множества. Селектор передается в аргументе при вызове метода.

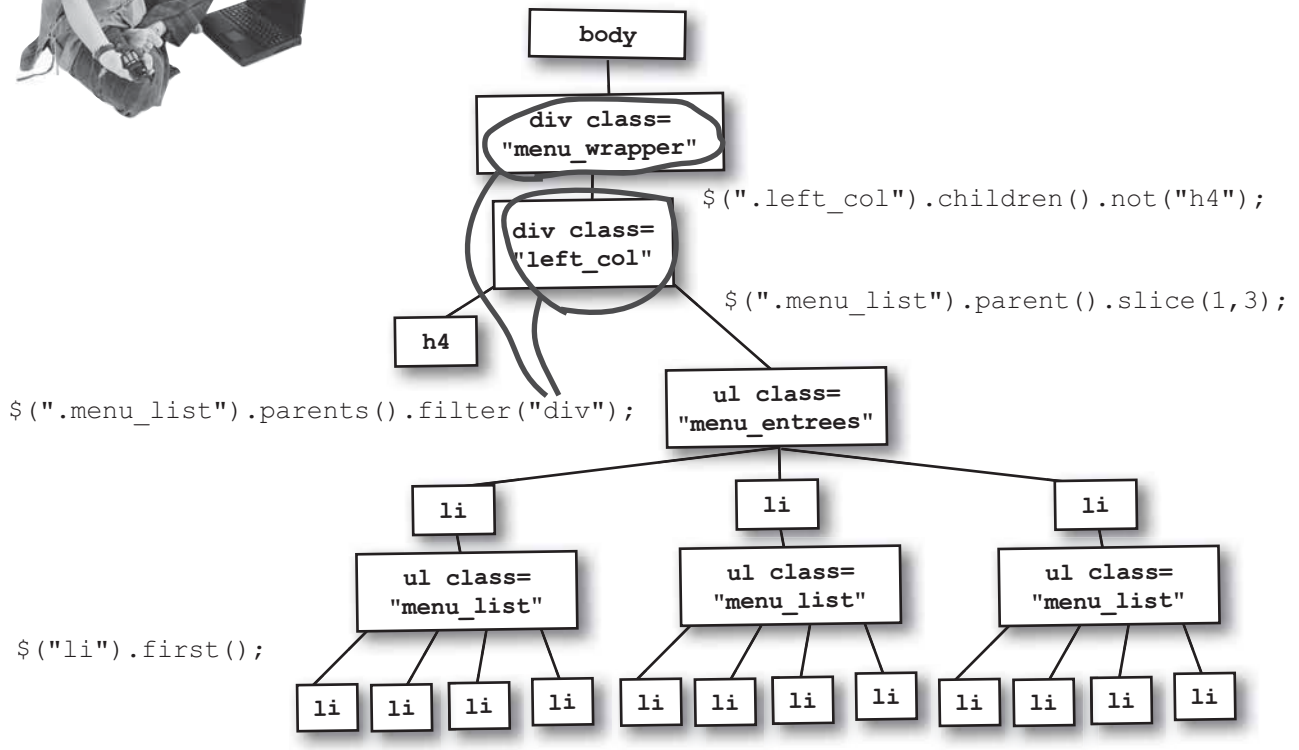


Какой из этих элементов поможет нам определить первого потомка меню?

Стань DOM



Представьте себя на месте дерева DOM. Соедините команды jQuery с элементом(–ами), которые будут выбраны при их выполнении. Считайте, что страница не содержит других элементов. Ответ для первой команды мы написали за вас.

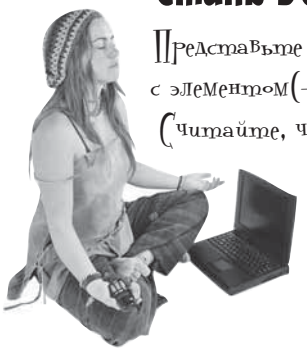


Возьми в руку карандаш

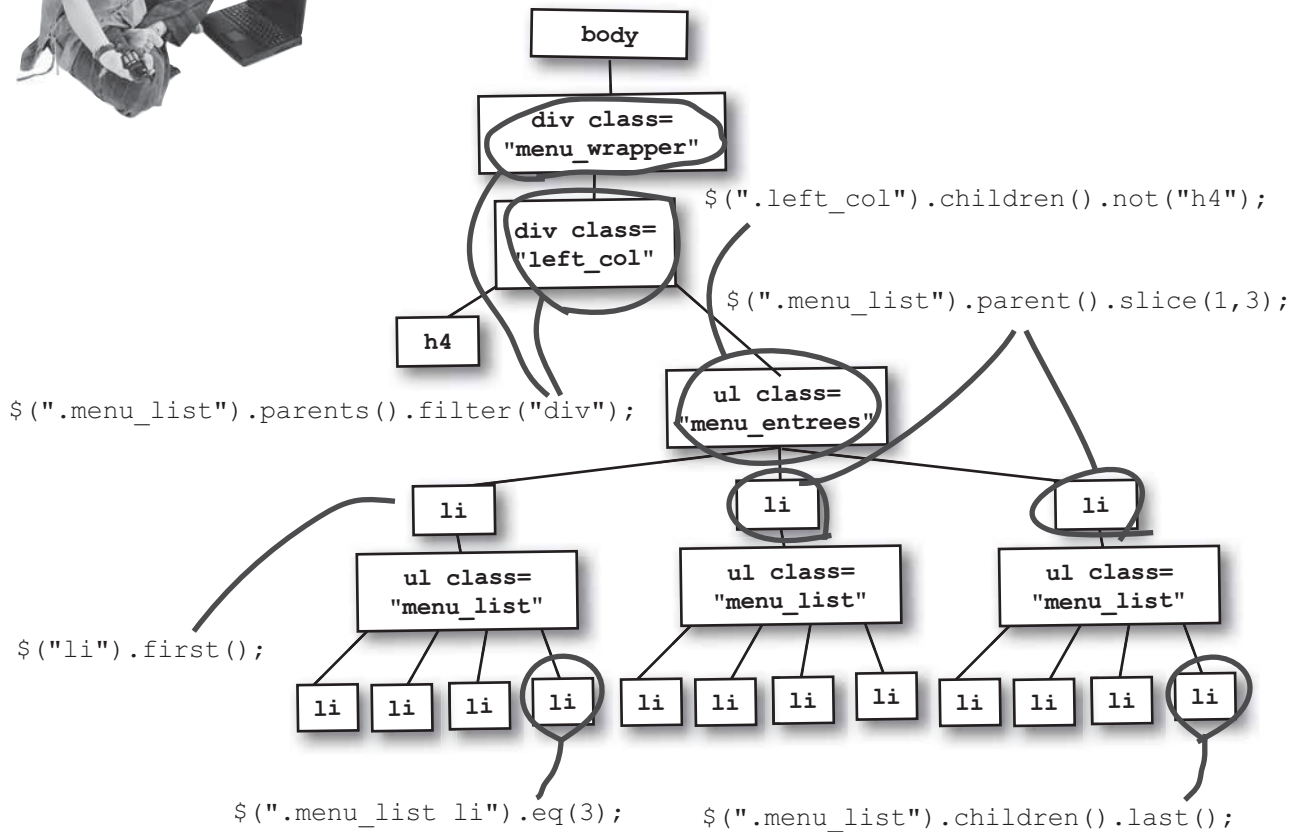
Напишите строку кода jQuery, которая вернет рыбные блюда в меню (например, перед первым элементом, вложенным в `menu_entrees`).

.....`.before($f);`

Стань DOM. Решение



Представьте себя на месте дерева DOM. Соедините команды jQuery с элементом(–ами), которые будут выбраны при их выполнении. Считайте, что страница не содержит других элементов. Ответ для первой команды мы написали за вас.



Возьми в руку карандаш
Решение

Напишите строку кода jQuery, которая вернет рыбные блюда в меню (например, перед первым элементом, вложенным в `menu_entrees`).

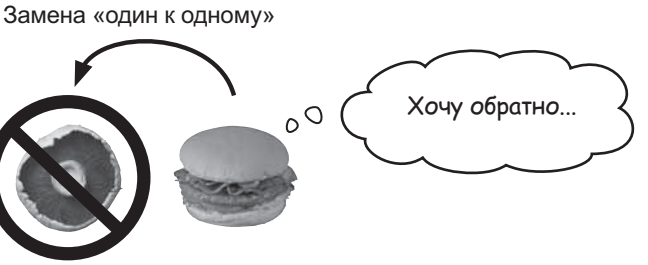
```
$(".menu_entrees li").first().before($f);
```

Верните гамбургер на место

Итак, первое требование в списке кнопки восстановления меню реализовано. Осталось еще два.

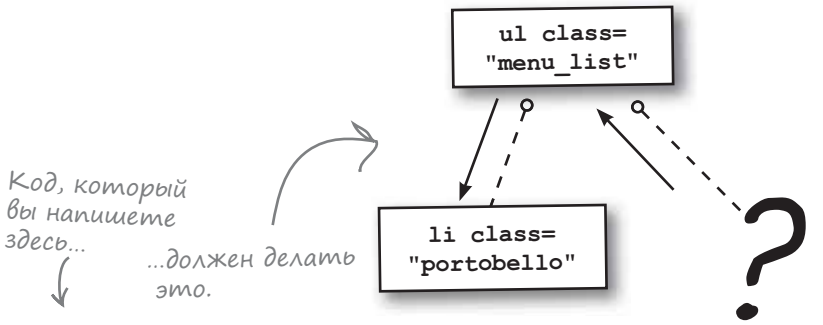
- [x] Вернуть рыбные блюда в меню (например, перед первым элементом меню в левом столбце).
- [ ] Найти шампиньоны в ингредиентах и заменить их гамбургерами.
- [ ] Найти тофу в ингредиентах и заменить его различными видами мясных ингредиентов (в правильном порядке).

Следующая задача кажется смутно знакомой, не правда ли? По сути, мы должны проделать исходную замену в обратном порядке. Почему? Да потому что мы имеем дело с заменой «один к одному», а такие замены хороши своей логической простотой.



Упражнение

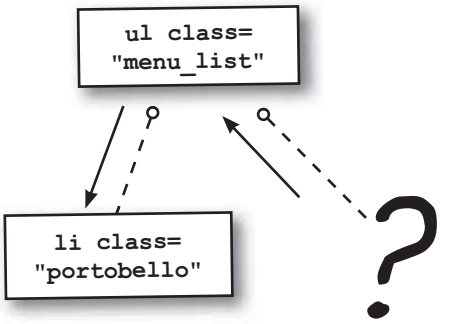
Помните это упражнение? Сейчас мы проделаем его в обратном порядке. Напишите код, который находит элементы `li` класса `portobello` и заменяет их элементами `li` класса `hamburger`. Следующая схема поможет вам лучше представить суть задачи. Мы уже написали часть решения за вас; допишите остальное.




```
$(          ).replaceWith(                                Hamburger      );
```

Упражнение
Решение

Всего одна короткая строка с `replaceWith` — и гамбургеры вернулись на место! Отличная работа!



Выбираем все элементы класса portobello.

Метод replaceWith динамически заменяет выбранное содержимое элементом в круглых скобках.

```
$(".portobello").replaceWith("<li class='hamburger'> Hamburger </li>");
```

И где же мясо?

Мы подошли к последнему пункту списка восстановления меню.

- [x] Вернуть рыбные блюда в меню (например, перед первым элементом меню в левом столбце).
- [x] Найти шампиньоны в ингредиентах и заменить их гамбургерами.
- [ ] Найти тофу в ингредиентах и заменить его различными видами мясных ингредиентов (в правильном порядке).

Вспомните, что мы сделали с элементами `li.meat`? Давайте посмотрим:

Элементы `li.tofu` были вставлены в DOM за элементами `meat`.

```
$(".meat").after("<li class='tofu'><em>Tofu</em></li>");
```

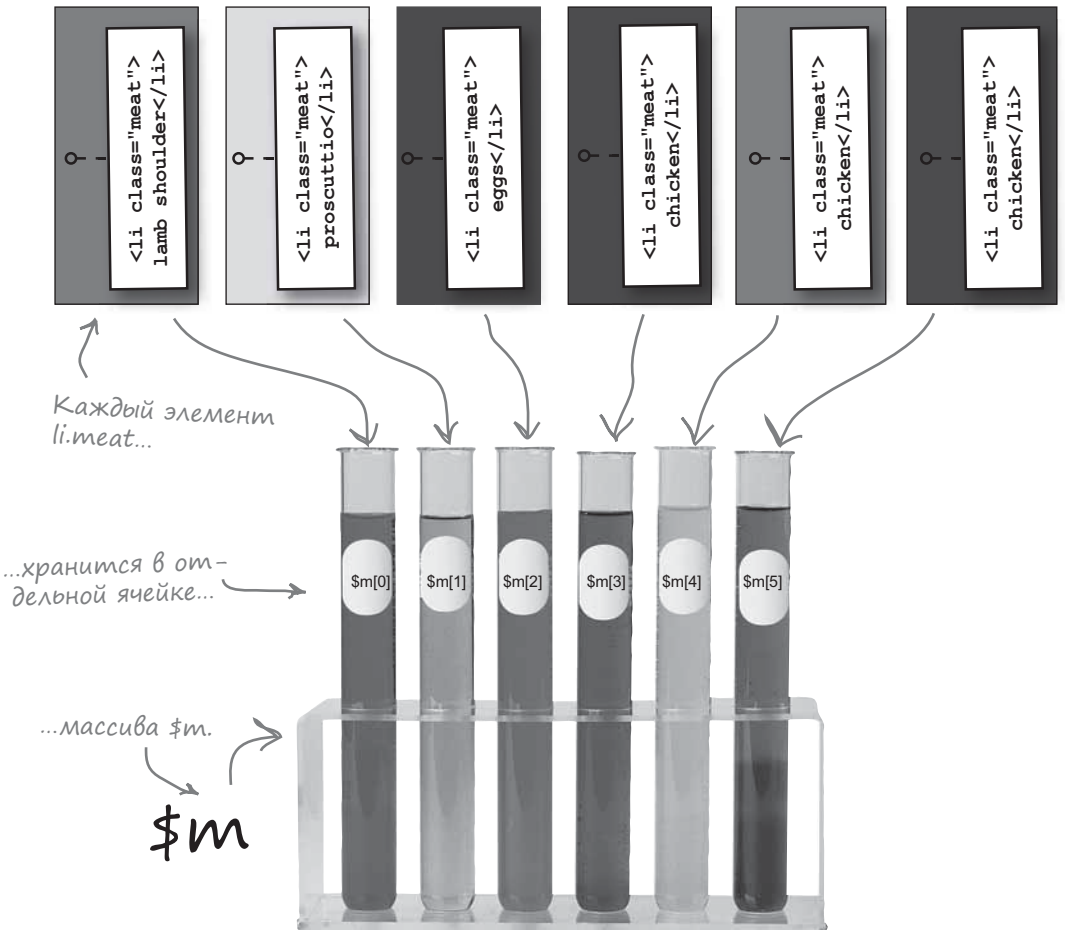
Затем мы отсоединили элементы `li.meat`, но сохранили их в `$m`.

```
$m = $(".meat").detach();
```

Так где взять эти элементы и как вернуть их на место?

Массив отсоединенных элементов

Вспомните, что при хранении элементов jQuery мы начинаем имя переменной со знака $, который показывает, что в переменной хранятся специальные данные. В нашем случае переменная содержит массив jQuery, в котором элементы `$m` хранятся следующим образом:



МОЗГОВОЙ ШТУРМ

Нужно вернуть на место каждый элемент `li.meat`, заменив им элемент `li.tofu`. Мы уже видели много методов, вставляющих элементы в DOM. Какой метод вы выбрали бы для этой задачи?

Метод each и перебор массивов

В главе 3 было показано, как использовать метод `each` для перебора элементов массива. Сейчас мы снова воспользуемся этим методом для перебора всех элементов `meat` в массиве `$m` и их возвращения на исходные места. Но для этого необходимо поближе познакомиться с тем, как работает метод `each`.

Метод each напоминает станок на конвейере.

Индекс отслеживает элемент, с которым работает функция.

$i = 0$

Настоящая сила метода each проявляется при включении в него функции. Эта функция последовательно выполняет некоторую операцию с каждым перебираемым элементом.

Метод each наглядно демонстрирует мощь сценариев jQuery: с его помощью можно последовательно обрабатывать элементы массива.

Ключевым словом this обозначается элемент, с которым работает функция.

Метод each последовательно обрабатывает элементы массива, выполняя некоторую операцию с каждым элементом.

jQuery сохраняет результаты выбора в массиве.

Переменная i отсчитывает обработанные элементы (нумерация начинается с 0).

```
$(".tofu").each(function(i){
  $(this).after(    );
});
```

Для ссылки на текущий обрабатываемый элемент используется запись $(this).

Здесь используется метод after, но для обработки массива элементов можно использовать любой метод jQuery.

Мы хотим вставить элемент `meat` после каждого элемента `li.tofu`. Что должно быть написано в скобках?

Развлечения с магнитами

Разложите магниты в правильном порядке, чтобы они образовали правильный код функции-обработчика кнопки `restoreMenu`. Несколько магнитов уже находятся на своих местах.

```
$("button#restoreMe").click(function(){
    if (v == true){
        v = false;
    }
});
```

my_scripts.js

```
"<li class='hamburger'>Hamburger</li>");
$m[i]);
.before($f);
$(".portobello").replaceWith(
});
$(".tofu").remove();
$(".menu_entrees li").first()
$(this).after(
$(".tofu").each( function(i){
```

Развлечения с магнитами. Решение

Разложите магниты в правильном порядке, чтобы они образовали правильный код функции-обработчика кнопки `restoreMenu`. Несколько магнитов уже находятся на своих местах.

```
$("button#restoreMe").click(function(){
    if (v == true){
        $(".portobello").replaceWith(  "<li class='hamburger'>Hamburger</li>");
        $(".menu_entrees li").first()  .before($f);
        $(".tofu").each( function(i){
            $(this).after(  $m[i]);
        });
        $(".tofu").remove();
        v = false;
    }
});
```

Чтобы вернуть на место элементы meat, мы указываем имя массива $m и индекс, соответствующий элементу tofu, с которым в данный момент работает функция.

my_scripts.js

Вроде... Все?

Мы выполнили все, что требовалось для кнопки восстановления меню. Давайте обновим файлы, и проект можно считать завершенным.

- [x] Вернуть рыбные блюда в меню (например, перед первым элементом меню в левом столбце).
- [x] Найти шампиньоны в ингредиентах и заменить их гамбургерами.
- [x] Найти тофу в ингредиентах и заменить его различными видами мясных ингредиентов (в правильном порядке).

P.S. И еще хотелось бы, чтобы замененные вегетарианские блюда в меню были помечены — например зеленым листом.

Да, вы абсолютно правы.

К счастью, веб-дизайнер уже включил класс `veg_leaf` в файл *my_style.css*. Вот как он выглядит:

```
.veg_leaf{
      list-style-image:url('../images/leaf.png');
      }
```

my_style.css

Упражнение

Напишите команду, которая назначает класс `veg_leaf` родителю родителя элемента класса `tofu`.

.....

Подсказка: в этом вам поможет addClass.

Упражнение
Решение

Пара перемещений по DOM, немного магии `addClass` — и дело сделано!

```
$(".tofu").parent().parent().addClass("veg_leaf");
```

Часто Задаваемые Вопросы

В: С другими фильтрами все понятно, но от `slice` у меня до сих пор голова идет кругом. Можно объяснить подробнее?

О: Метод `slice` действительно непрост. Больше всего сложностей возникает с его параметрами: `slice(start, end)`.

Первый параметр — `start` — является обязательным; без него `slice` работать не будет. Параметр `start` указывает, с какого индекса начинается выделяемый диапазон. Помните, что у первого элемента массива индекс равен 0. Параметр `start` также может принимать отрицательные значения. В этом случае `slice` отсчитывает значения от конца, а не от начала массива.

В: А что делает параметр `end` метода `slice`?

О: Второй параметр метода `slice` — `end` — не является обязательным. Если он отсутствует, то `slice` начинает от позиции, заданной параметром `start`, и выбирает все элементы с индексами, большими `start`. Если забыть о том, что индексы элементов массива начинаются с 0, смысл параметра `end` может быть неочевидным.

В: Похоже, метод `each` способен на многое. А как `each` узнает, с каким элементом он работает?

О: Настоящая сила `each` проявляется в сочетании с ключевым словом `this`. Метод `each` автоматически отслеживает значение индекса и «знает», с каким элементом он работает. Используйте `each` только при выборе нескольких элементов. Для обращения к текущему элементу используйте ключевое слово `this`, но заключите его в сокращенную форму jQuery: `$(this)`.

В: Почему мы включаем «i» или «index» в каждую функцию `each`?

О: Индексная переменная, которой обычно присваивается имя «i» или «index», используется функцией `each` для отслеживания элемента, с которым работает функция. При помощи этой переменной `each` определяет, когда перебор завершен, а обработку следует прекратить.

В: Как провести поиск элементов в массиве jQuery?

О: Для поиска элементов в массиве jQuery используется метод `find`. Допустим, у нас имеется массив элементов `li` в массиве jQuery:

```
var $my_elements = $("li");
```

Чтобы найти в этом массиве все якорные элементы, используйте следующую команду:

```
$my_elements.find("a");
```

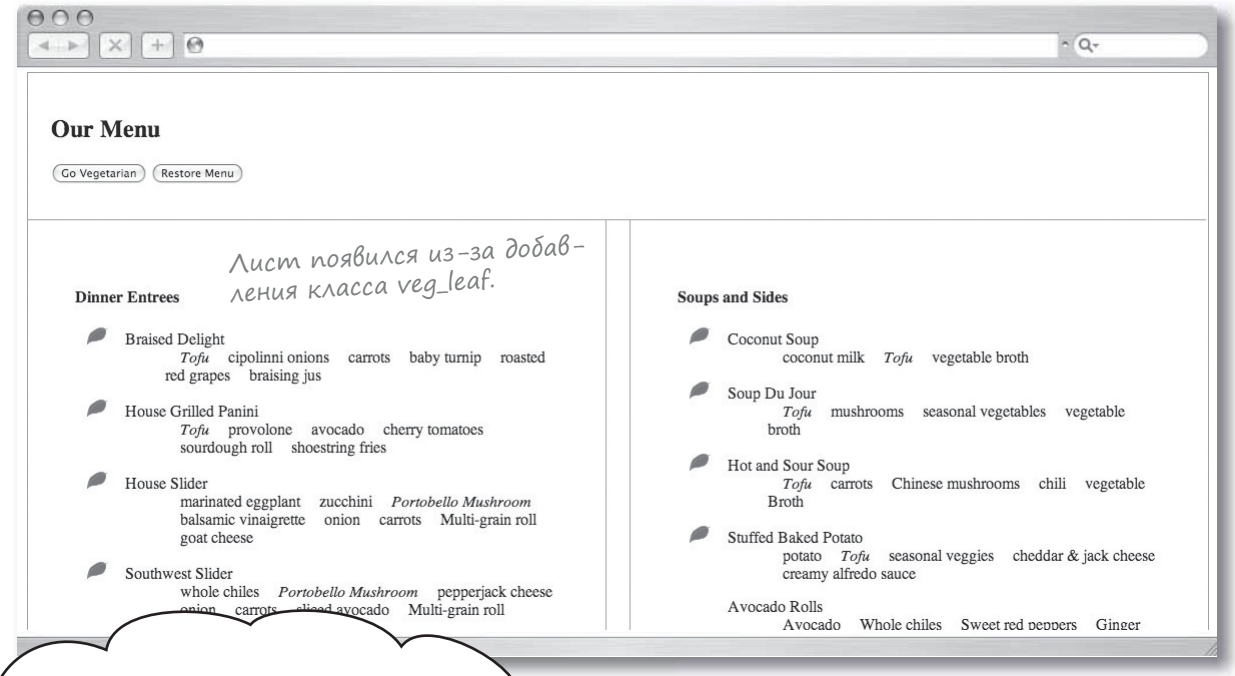
В: Позволяет ли jQuery заключить элемент внутри другого элемента?

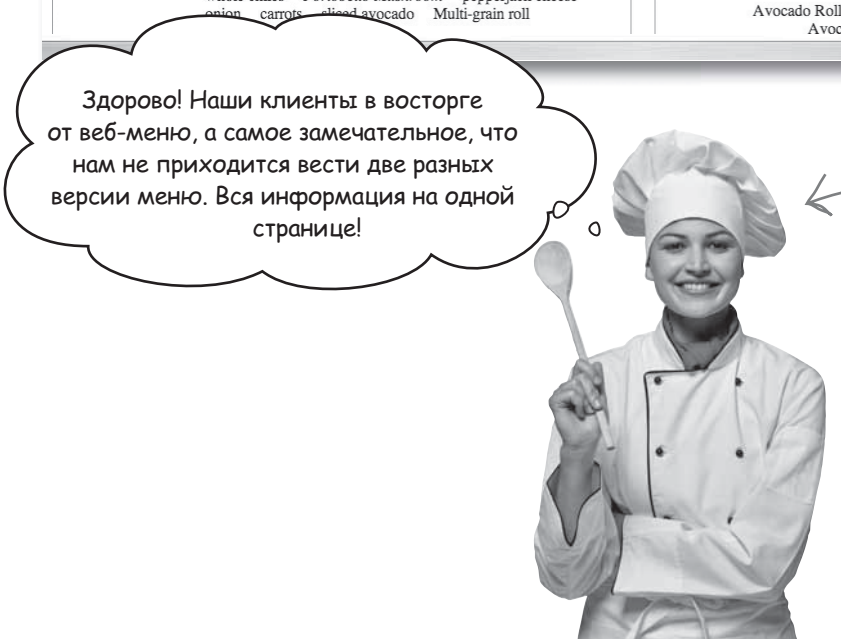
О: Да, позволяет. Допустим, изображение с дентификатором `oreilly` нужно заключить внутри якорного элемента. Это делается так:

```
$("img#oreilly").wrap("<a
href='http://www.oreilly.
com'></a>");
```

ТЕСТ-ДРАЙВ

С тех пор как мы в последний раз обновляли файлы, прошло уже немало времени. Добавьте код кнопки восстановления меню, а также код добавления/удаления класса `veg_leaf` для блюд с вегетарианскими аналогами. Также вы всегда можете загрузить файлы этой главы по адресу *http://www.thinkjquery.com/chapter04/* и сравнить их со своим кодом.





Теперь у Александры будет меньше хлопот с сопровождением сайта — и у нее появится время для приготовления новых блюд.

ГЛАВА 4

Ваш инструментарий jQuery

Глава 4 осталась позади, а ваш творческий арсенал расширился: в нем появились средства обхода и выполнения операций с DOM, массивы и фильтры.

Операции с DOM

Вы можете выполнять операции добавления, замены и удаления элементов из DOM:

detach

remove

replaceWith

before

after

Обход DOM

Перемещение по дереву DOM для выполнения различных операций.

Для перехода к нужной позиции дерева используются отношения между элементами и различные методы (например, parent и child).

Сцепленные вызовы методов эффективны для быстрого перемещения по дереву DOM.

Массивы

В массивах jQuery может храниться любая информация (в том числе и элементы).

Как и в случае с переменными, перед именем массива принято ставить знак $, который показывает, что в массиве хранятся специальные данные jQuery.

Фильтры

Фильтры предназначены для сокращения множества выбранных элементов:

first

equal

last

slice

filter

not

Плавно и изящно

Реализация всяких интересных возможностей — дело замечательное, но если ваша страница не будет хорошо смотреться, люди не станут приходить на сайт. И здесь на первый план выходят визуальные эффекты и анимация jQuery. Вы научитесь организовывать переходы, скрывать и отображать нужные части элементов, изменять размеры элементов на странице — и все это на глазах у ваших посетителей! Вы научитесь планировать выполнение анимаций, чтобы они происходили с различными интервалами, отчего ваша страница будет выглядеть исключительно динамично.

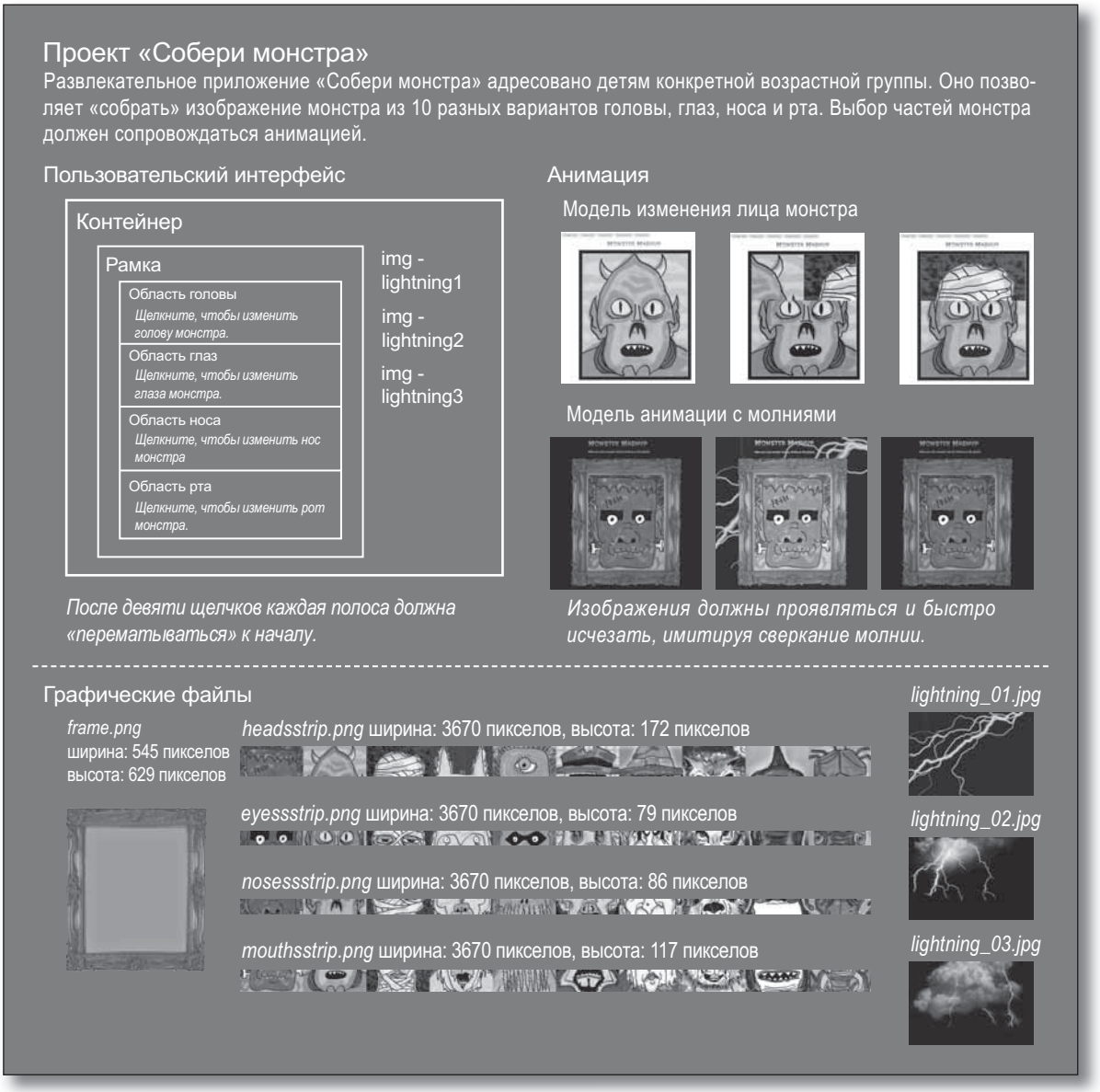
Новый заказ

Фирма DoodleStuff поставляет детям принадлежности для рисования. Несколько лет назад фирма открыла популярный веб-сайт с интерактивными приложениями для детей. Популярность фирмы стала расти настолько быстро, что она не успевает справляться с пожеланиями своих клиентов.

Ориентируясь на новую, более широкую аудиторию, руководитель веб-проектов хочет создать приложение, которое не потребует установки Flash или других дополнительных модулей для браузеров.

Проект «Собери монстра»

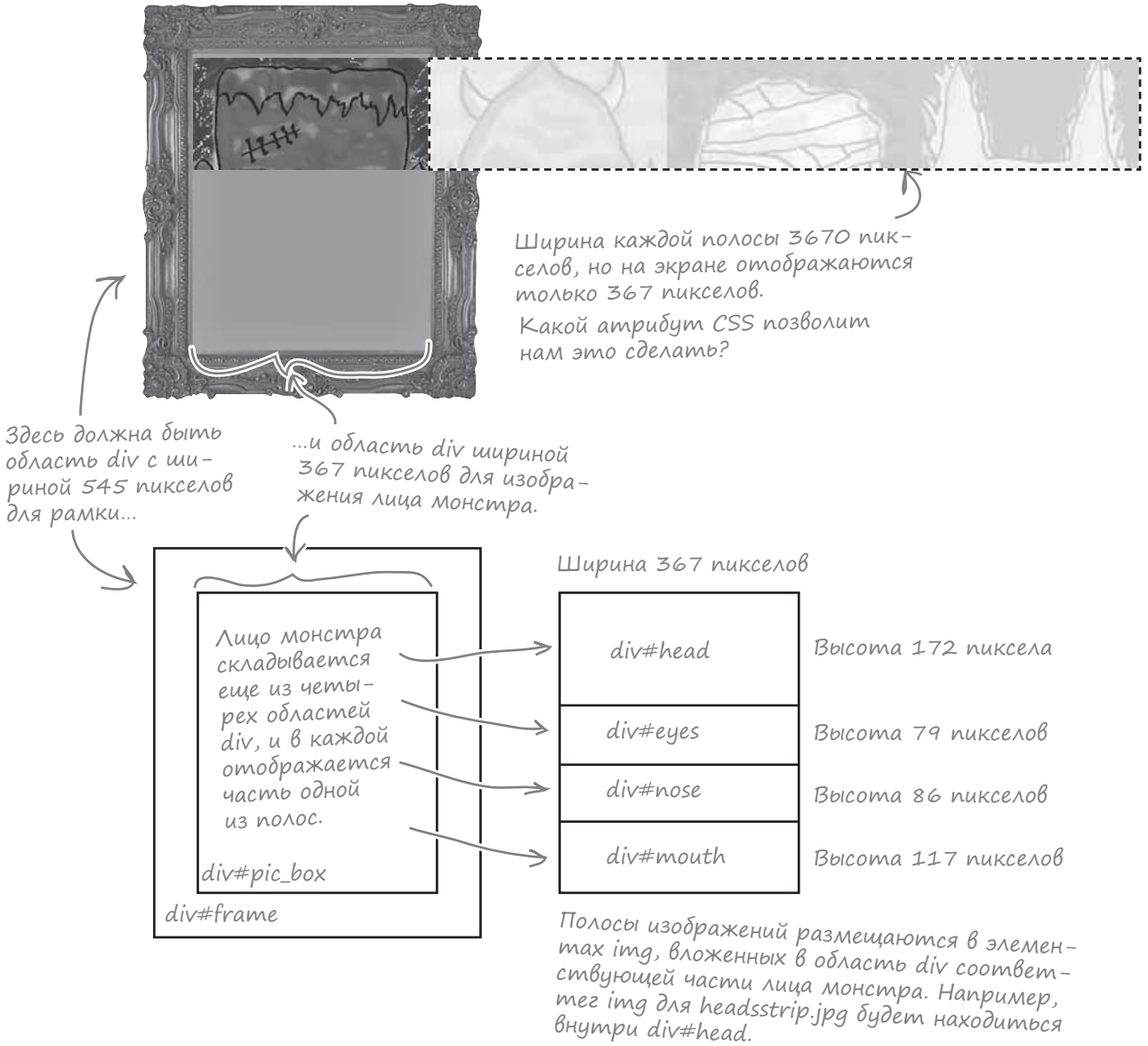
Перед вами схема нового проекта, которую вам дал руководитель веб-проектов, а также переданные веб-дизайнером графические файлы.



У вас имеется подробное описание требований и необходимые графические файлы, но нет разметки HTML и кода CSS — с этого и следует начинать. Что для этого необходимо?

Макет и позиционирование

Мы уже неоднократно говорили о том, как важно тщательно продумать структуру и стиль приложения до перехода к программированию. В данном случае это еще важнее — если не позаботиться о правильном расположении всех визуальных элементов, с эффектами и анимацией *очень скоро* начнутся проблемы. Нет ничего противнее, чем глазеть в свой код jQuery и ломать голову над тем, почему же он не работает так, как нужно вам. Лучше заранее построить эскиз приложения и поразмыслить над тем, что будет происходить на экране.



Упражнение

Запишите в пустых местах файлов HTML и CSS идентификатор CSS, свойство или значение, обеспечивающее нужное размещение элементов приложения «Собери монстра». Если сомневаетесь, перечитайте две предыдущие страницы. Мы заполнили несколько пропусков за вас.

```
body>
<header id="top"><img src="images/Monster_Mashup.png" />
<p>Make your own monster face by clicking on the picture.</p></header>

<div id="frame">
    <div id="pic_box">
        <div id=.....class="face"><img src="images/headsstrip.jpg"></div>
        <div id=.....class="face"><img src="images/eyesstrip.jpg"></div>
        <div id=.....class="face"><img src="images/nosesstrip.jpg"></div>
        <div id=.....class="face"><img src="images/mouthsstrip.jpg"></div>
    </div>
</div>
  <script type="text/javascript" src="scripts/jquery-1.6.2.min.js"></script>
  <script type="text/javascript" src="scripts/my_scripts.js"></script>
</body>
```

index.html

```
#frame {
  position:.....
  left:100px;
  top:100px;
  width:545px;
  height:629px;
  background-image:url(images/frame.png);
  z-index: 2;
  overflow:.....
}

#pic_box{
  position: relative;
  left:91px;
  top:84px;
  .....
  height:460px;
  z-index: 1;
  overflow:.....
}
```

```
.face{
  position:.....
  left:0px;
  top:0px;
  z-index: 0;
}

#head{
  height:172px;
}

#eyes{
  .....
}

#nose{
  .....
}

#mouth{
  .....
}
```

my_style.css

Запишите в пустых местах файлов HTML и CSS идентификатор CSS, свойство или значение, обеспечивающее нужное размещение элементов приложения Monster Mashup. Если засомневаетесь, просмотрите еще раз две предыдущие страницы. Мы заполнили несколько пропусков за вас.

```
body>
<header id="top"><img src="images/Monster_Mashup.png" />
<p>Make your own monster face by clicking on the picture.</p></header>

<div id="frame">
    <div id="pic_box">
        <div id="head" class="face"><img src="images/headsstrip.jpg"></div>
        <div id="eyes" class="face"><img src="images/eyesstrip.jpg"></div>
        <div id="nose" class="face"><img src="images/nosesstrip.jpg"></div>
        <div id="mouth" class="face"><img src="images/mouthsstrip.jpg"></div>
    </div>
</div>
  <script type="text/javascript" src="scripts/jquery-1.6.2.min.js"></script>
  <script type="text/javascript" src="scripts/my_scripts.js"></script>
</body>
```

index.html

```
#frame {
  position: absolute;
  left:100px;
  top:100px;
  width:545px;
  height:629px;
  background-image:url(images/frame.png);
  z-index: 2;
  overflow: hidden;
}

#pic_box{
  position: relative;
  left:91px;
  top:84px;
  width:367px;
  height:460px;
  z-index: 1;
  overflow: hidden;
}
```

При анимации позиции элемента используется абсолютное или относительное позиционирование.

Присваивание overflow значения hidden позволяет скрыть часть полосы изображения, выходящую за границы области pic_box.

Для этого также можно воспользоваться свойством CSS clip.

```
.face{
  position: relative;
  left:0px;
  top:0px;
  z-index: 0;
}

#head{
  height:172px;
}

#eyes{
  height:79px;
}

#nose{
  height:86px;
}

#mouth{
  height:117px;
}
```

my_style.css

Еще немного структуры и стиля

Далее необходимо разобраться со структурными изменениями в файлах HTML и CSS. Включите приведенный ниже код в файлы *index.html* и *my_style.css*. Графические файлы можно загрузить по адресу *www.thinkjquery.com/chapter05*.

Задание!

Добавляем контейнер и вкладываем в него изображение с молниями.

```
<div id="container">
  <img class="lightning" id="lightning1" src="images/lightning-01.jpg" />
  <img class="lightning" id="lightning2" src="images/lightning-02.jpg" />
  <img class="lightning" id="lightning3" src="images/lightning-03.jpg" />
    <div id="frame">
        <div id="pic_box">
           <div id="head" class="face"><img src="images/headsstrip.jpg"></div>
           <div id="eyes" class="face"><img src="images/eyesstrip.jpg"></div>
           <div id="nose" class="face"><img src="images/nosesstrip.jpg"></div>
           <div id="mouth" class="face"><img src="images/mouthsstrip.jpg"></div>
        </div>
    </div>
</div>
```

index.html

```
#container{
    position:absolute;
    left:0px;
    top:0px;
    z-index: 0;
}

.lightning{
    display:none;
    position:absolute;
    left:0px;
    top:0px;
    z-index: 0;
}

body{
    background-color:#000000;
}
p{
    color:#33FF66;
    font-family: Tahoma, Verdana, Arial,
    Helvetica, sans-serif;
    font-size:12px;
}
#text_top {
    position:relative;
    z-index: 4;
}
```

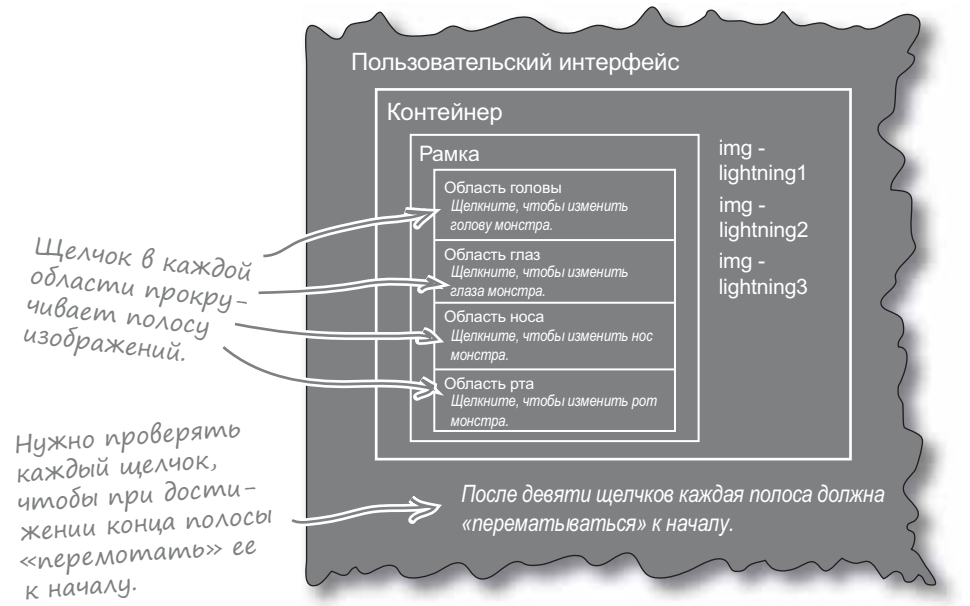
В исходном состоянии молнии должны быть невидимы.

Чтобы использовать анимацию для элемента, следует задать его свойству position значение absolute, fixed или relative.

my_style.css

Проработка интерфейса

Итак, мы разобрались с визуальным макетом приложения «Собери монстра». Давайте проработаем интерактивную часть пользовательского интерфейса, представленную в эскизе. Нам предстоит запрограммировать реакцию разных элементов страницы на щелчки. Собственно, этим мы занимаемся уже четыре главы, так что сейчас это уже пара пустяков.



Часто Задаваемые Вопросы

В: Что это за свойство CSS position? Почему оно необходимо для анимации и эффектов jQuery?

О: Свойство CSS `position` управляет тем, где и как браузерный движок размещает элементы. Многие эффекты jQuery реализуются с использованием свойства `position`. Если вы забыли, как работает это свойство, обращайтесь к превосходному объяснению в центре для разработчиков Mozilla:

http://developer.mozilla.org/en/CSS/position#Relative_positioning

В: Почему для анимации элементов свойству CSS position необходимо задать значение `absolute`, `fixed` или `relative`?

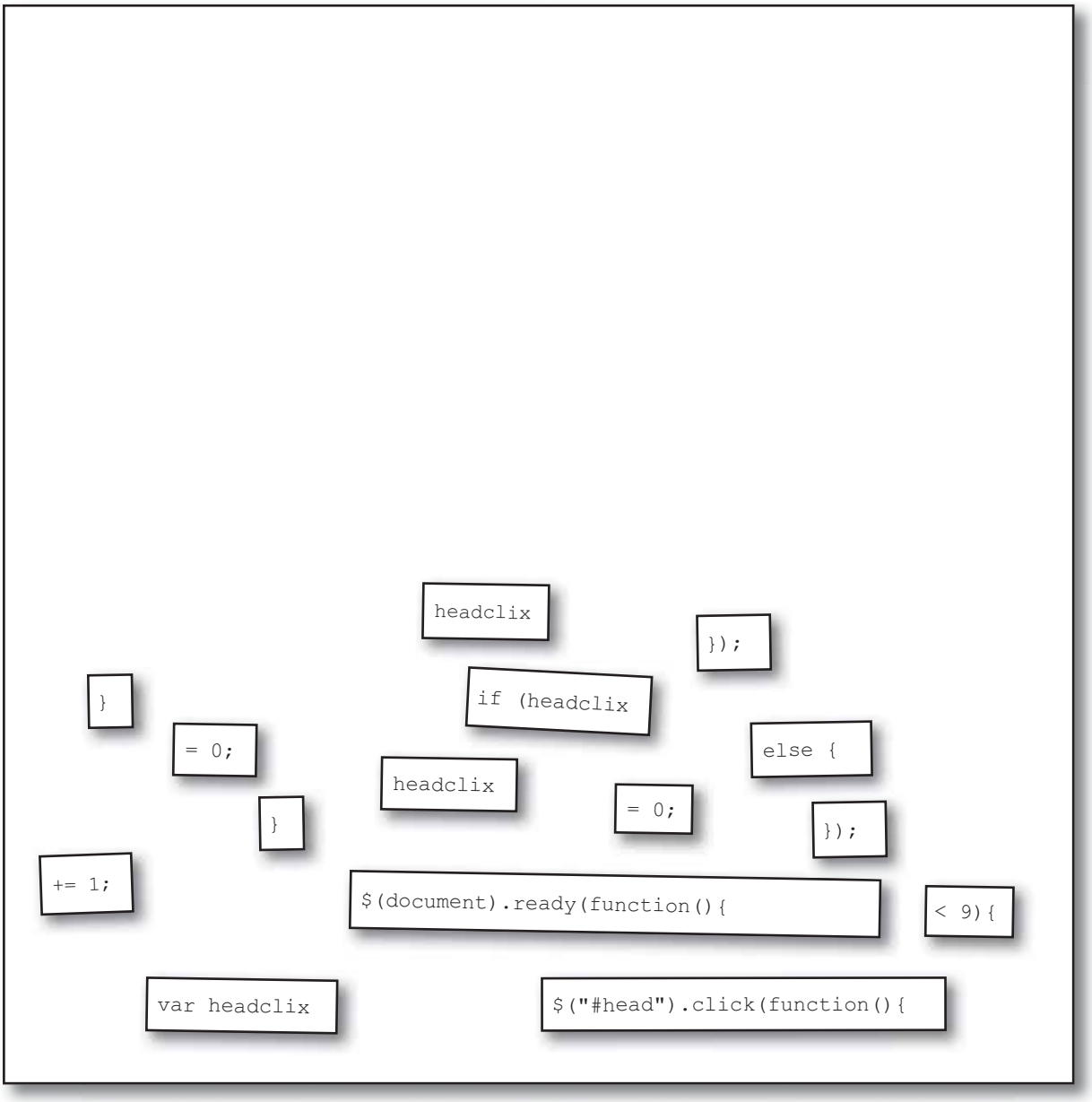
О: Если оставить свойству CSS `position` значение по умолчанию (т. е. `static`), то для элемента будет невозможно задать позицию сторон (`top`, `right`, `left` и `bottom`). При использовании функции `animate` возможность изменения этих позиций необходима, а в режиме `static` это невозможно. Со значениями `absolute`, `fixed` и `relative` — такой проблемы нет.

В: Вы упомянули какой-то «браузерный движок». А *это* что такое?

О: Браузерный движок визуализации — один из важнейших компонентов браузера, который интерпретирует разметку HTML и код CSS и отображает результат в *окне просмотра* браузера. Google Chrome и Safari используют движок визуализации Webkit; Firefox использует Gecko, а Microsoft Internet Explorer — движок, который называется Trident.

Развлечения с магнитами

Расставьте магниты в правильном порядке, чтобы элемент `div#head` реагировал на щелчки. Проследите за тем, чтобы переменные и условные конструкции следовали в правильном порядке для обнаружения девятого щелчка.



Развлечения с магнитами. Решение

Расставьте магниты в правильном порядке, чтобы элемент `div#head` реагировал на щелчки. Проследите за тем, чтобы переменные и условные конструкции следовали в правильном порядке для обнаружения девятого щелчка.

```
$(document).ready(function(){
    var headclix = 0;
    $("#head").click(function(){
        if (headclix < 9){
            headclix += 1;
        }
        else {
            headclix = 0;
        }
    });
});
```

В исходном состоянии переменная равна 0, потому что щелчков еще не было.

Условие ограничивает пользователя девятью щелчками.

Здесь будет размещаться код анимации.

Значение переменной headclix увеличивается на 1.

Если значение headclix больше либо равно 9, сделать следующее:

Здесь будет размещаться код «перемотки» полосы изображений.

После девятого щелчка переменной headclix возвращается значение 0.

Конечно, можно!

Все элементы работают по той же схеме, что и элемент `div#head` (с небольшими изменениями вроде имени переменной).

Возьми в руку карандаш

Допишите сценарий, чтобы он обрабатывал щелчки на других частях монстра (глаза, нос и рот). Вскоре мы дополним обработчики дополнительной функциональностью. Проследите, чтобы переменные и условные конструкции следовали в правильном порядке для обнаружения девятого щелчка.

```
$(document).ready(function(){

        $("#head").click(function(){
             if (headclix < 9){
                    headclix += 1;
                    }
                    else{
                    headclix = 0;
                    }
        });









});
```

my_scripts.js

Возьми в руку карандаш
Решение

Вы обеспечили обработку щелчков на других частях монстра (глаза, нос и рот), а также расположили переменные и условные конструкции в правильном порядке для обнаружения девятого щелчка.

```
$(document).ready(function(){
        var headclix = 0, eyeclix=0, noseclix= 0, mouthclix = 0;
          $("#head").click(function(){
               if (headclix < 9){
                     headclix += 1;
               }
               else{
                     headclix = 0;
               }
          });
          $("#eyes").click(function() {
               if (eyeclix < 9){
                     eyeclix += 1;
               }
               else{
                     eyeclix = 0;
               }
          });
          $("#nose").click(function() {
               if (noseclix < 9){
                     noseclix += 1;
               }
               else{
                     noseclix = 0;
               }
          });
          $("#mouth").click(function() {
               if (mouthclix < 9){
                     mouthclix += 1;
               }
               else{
                     mouthclix = 0;
               }
          });
});
```

В одной строке можно объявить и инициализировать несколько переменных, разделяя их запятыми.

Теперь каждая часть лица монстра реагирует на щелчки, а после девяти щелчков полоса «перематывается» в начало.

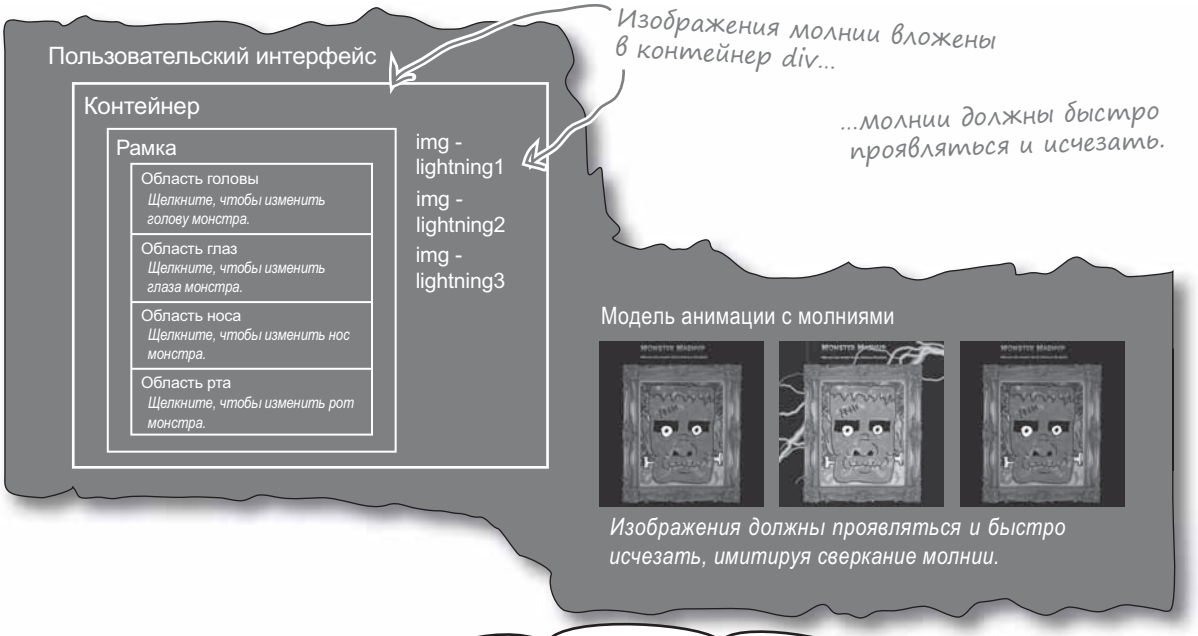
Обратите внимание: все функции click очень похожи друг на друга с минимальными отклонениями. Нельзя ли повторно использовать один фрагмент кода?

Терпение — этим мы займемся в главе 7.

my_scripts.js

Эффект молнии

Теперь займемся эффектом молнии. Прежде чем браться за реализацию эффекта, посмотрим, что об этом говорится в схеме приложения.



Мы уже использовали эффекты изменения прозрачности в главе 1. Нельзя ли воспользоваться теми же средствами в приложении «Собери монстра»?

Возможно. Но нет ли чего-нибудь поинтереснее?

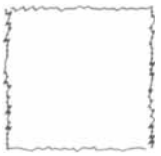
В главе 1 мы воспользовались готовыми эффектами jQuery, а сейчас попробуем разобраться поглубже.

Как jQuery выполняет анимацию элементов?

Во время загрузки файла CSS браузер задает визуальные свойства элементов страницы. При использовании встроенных эффектов jQuery интерпретатор Java изменяет эти свойства CSS, и изменения происходят динамически прямо у вас перед глазами. Но никакого волшебства в этом нет... Все дело в свойствах CSS. Давайте еще раз вернемся к тому, что вы уже видели.

Методы hide, show и toggle изменяют свойство CSS display

hide

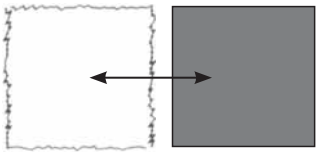


Интерпретатор JS задает свойству CSS `display` выбранного элемента значение `none`; элемент перестает отображаться на странице.

show

Интерпретатор JS изменяет свойство CSS `display` выбранного элемента так, что тот становится видимым.

toggle



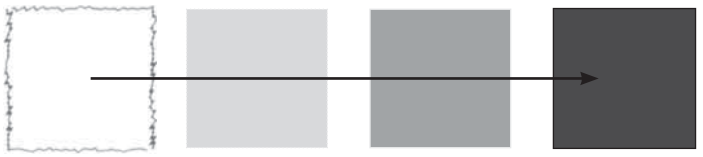
Если элемент скрыт, то интерпретатор JS отображает его, и наоборот.

Эффекты jQuery изменяют свойства CSS «на ходу», в результате чего страница изменяется прямо перед глазами пользователя.

Методы `hide`, `show` и `toggle` работают со свойством `display`. Но нам нужно, чтобы части лица не просто появлялись, а «скользили», а молнии проявлялись и исчезали. Как вы думаете, какие свойства CSS изменяются jQuery в эффектах скольжения и проявления/исчезновения?

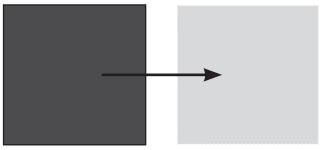
Эффекты изменения прозрачности изменяют свойство CSS opacity

fadeIn



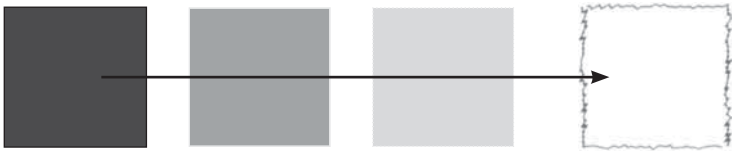
В эффекте проявления `fadeIn` интерпретатор JavaScript изменяет свойство CSS `opacity` выбранного элемента от 0 до 100.

fadeTo



`fadeTo` выполняет анимацию выбранного элемента до заданного уровня прозрачности (в процентах).

fadeOut

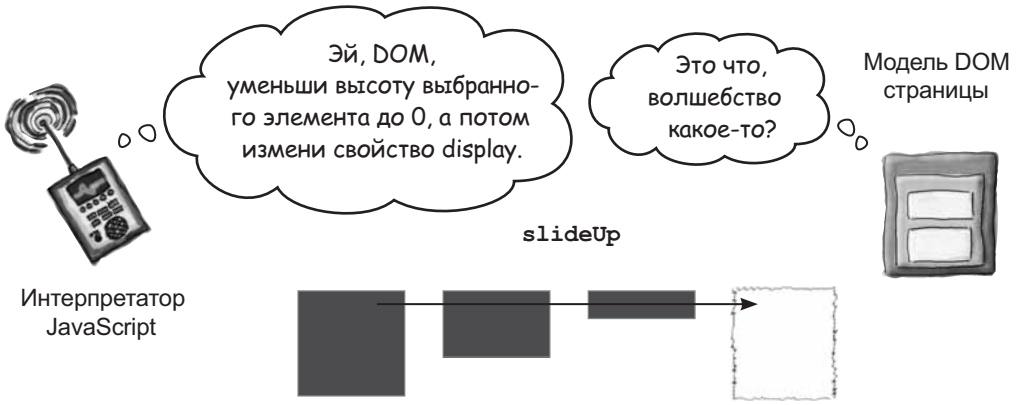


В эффекте исчезновения `fadeOut` интерпретатор JavaScript изменяет свойство CSS `opacity` выбранного элемента от 100 до 0, но сохраняет на странице место, занимаемое элементом.

Для любознательных

Свойство CSS `opacity` по-разному работает в разных браузерах. К счастью, jQuery берет на себя все технические тонкости. Собственно, это единственное, что вам необходимо о них знать!

Эффект скольжения



Интерпретатор JavaScript приказывает DOM уменьшить свойство CSS `height` выбранного элемента до 0, а затем задать свойству `display` значение `none`. По сути, элемент скрывается посредством скольжения.

slideDown

Интерпретатор JavaScript снова делает выбранный элемент(-ы) видимым, для чего его свойство `height` увеличивается от 0 до высоты, заданной в стиле CSS.

slideToggle

Интерпретатор JavaScript проверяет, имеет изображение полную или нулевую высоту, и переключает эффект скольжения в зависимости от результата. Если высота элемента равна 0, то интерпретатор JavaScript использует скольжение вниз, а при полной высоте используется скольжение вверх.

В jQuery включены готовые эффекты только для вертикального скольжения элементов.

В jQuery не существует методов `slideRight` или `slideLeft` (по крайней мере на момент написания книги). Не беспокойтесь, вскоре мы решим эту проблему...

В jQuery не существует методов slideRight и slideLeft.

Возьми в руку карандаш

Какие из готовых эффектов jQuery подойдут для приложения «Собери монстра»? Для каждого вида эффектов напишите, можно ли использовать их в нашем случае, и объясните, почему вы выбрали (или не выбрали) данный эффект.

Эффект	Можно ли использовать?	Почему?
Скрытие/отображение элемента		
Скольжение		
Изменение прозрачности		

Возьми в руку карандаш
Решение

Какие из готовых эффектов jQuery подойдут для приложения «Собери монстра»?

Эффект	Можно ли использовать	Почему?
Отображение/скрытие	Нет	Эти эффекты в нашем приложении бесполезны, нам не нужно анимировать свойство display каких-либо элементов.
Скольжение	Нет	Чуть ближе, и все же не то. Полоса изображений должна скользить влево. Методы SlideUp и slideDown изменяют свойство height, а нам нужно свойство left.
Изменение прозрачности	Да	Эффекты изменения прозрачности позволяют выполнить пункт спецификации, в котором говорится, что изображения молний должны быстро проявляться и исчезать, имитируя электрические разряды.

Как работают эффекты изменения прозрачности

В спецификации говорится, что изображения молний должны быстро проявляться и исчезать, но для имитации сверкания изменения должны быть достаточно быстрыми. Давайте поглубже разберемся в том, как работают эффекты изменения прозрачности, и попробуем реализовать эффект сверкания молнии.

Идентификатор первого элемента img.

Параметр управляет скоростью завершения эффекта.

```
$("#lightning1").fadeIn("fast");
```

Используйте одно из строковых значений: slow, normal или fast...

...или значение в миллисекундах. Например, если указать при вызове значение 1000, то анимация эффекта будет выполнена за одну секунду.

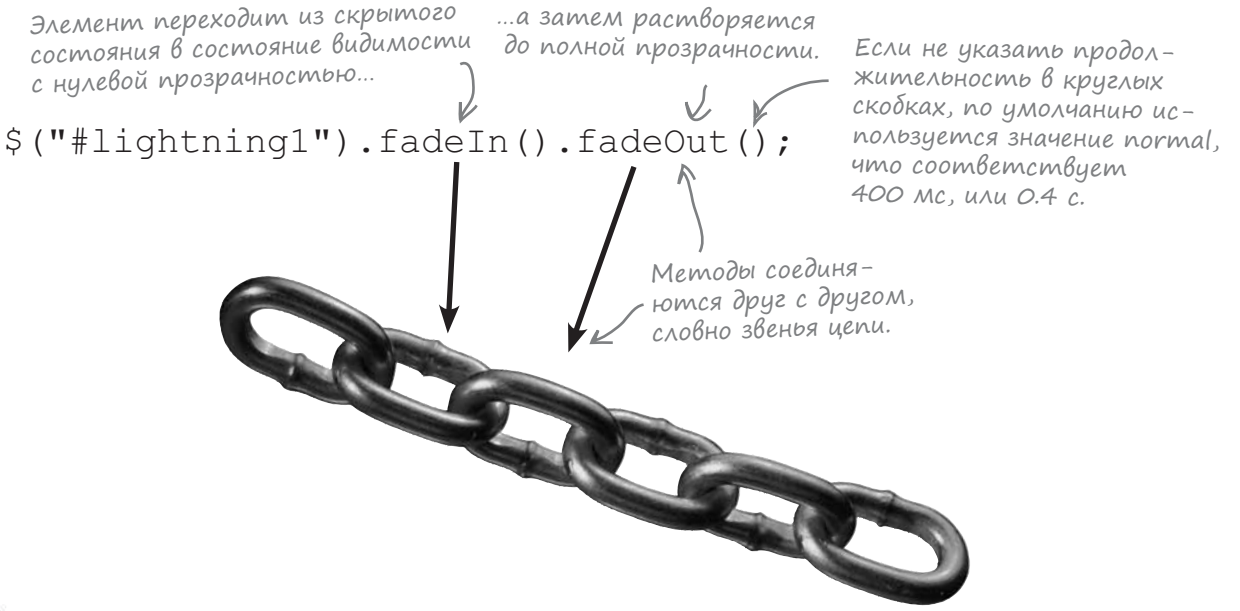
```
$("#lightning1").fadeIn(1000);
```

Не забыть!

1 секунда = 1000 миллисекунд

Комбинированные эффекты

Молнии должны проявляться и исчезать, и так несколько раз. Вместо того чтобы реализовывать повторяющиеся эффекты по отдельности, лучше воспользоваться сцепленными вызовами (которые кратко упоминались в главе 4, когда мы занимались перемещениями по дереву DOM). jQuery поддерживает возможность сцепления вызовов для последовательного применения серии методов к возвращаемому набору элементов. Как вы вскоре убедитесь, сцепление упрощает реализацию сверкания молний и делает запись более компактной.



Упражнение

Напишите пару строк кода jQuery, которые выполняют следующие операции:

1. Элемент `#lightning1` проявляется с продолжительностью в 1/4 секунды.

   .....

2. Присоедините к первому вызову еще один, обеспечивающий исчезновение элемента `#lightning1` за 1/4 секунды.

   .....

Упражнение
Решение

Напишите пару строк кода jQuery, которые выполняют следующие операции:

1. Элемент `#lightning1` проявляется с продолжительностью в 1/4 секунды.

```
$("#lightning1").fadeIn("250");
```

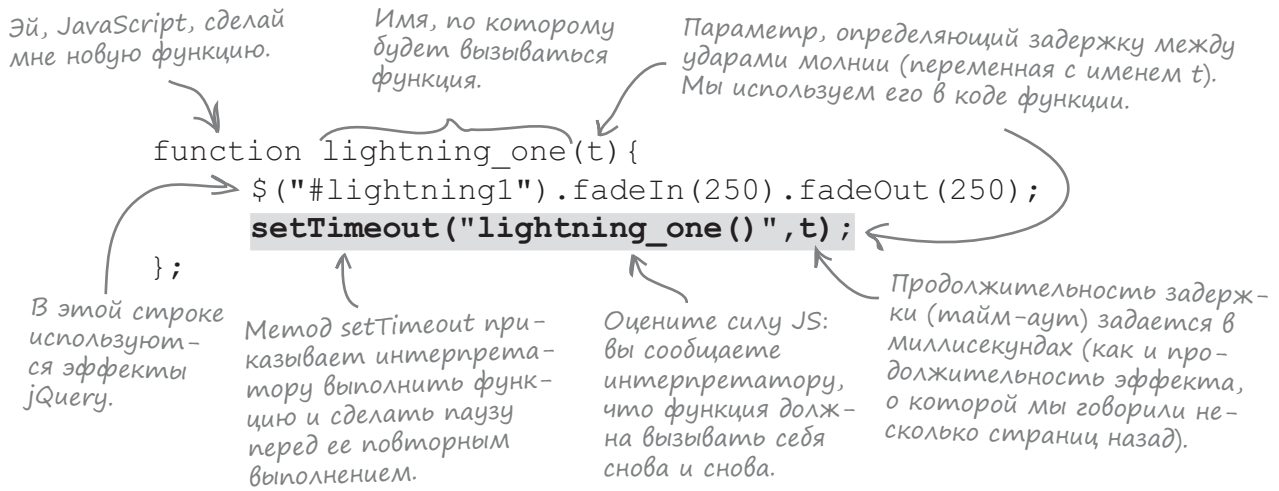
2. Присоедините к первому вызову еще один, обеспечивающий исчезновение элемента `#lightning1` за 1/4 секунды.

```
$("#lightning1").fadeIn("250").fadeOut("250");
```

Задержка при использовании эффектов

Вы знаете, как реализовать проявление и исчезновение молний, но в требованиях к проекту сказано, что разряды должны быть многократными. Во время грозы молнии не следуют непрерывно, между ними есть паузы. А значит, нужно организовать многократное применение эффекта.

Вспомните предыдущие главы: что мы использовали для повторяющихся операций? Правильно: функции! Они появились в главе 3, когда мы создавали универсальную функцию click и генератор случайных чисел. В нашей ситуации можно использовать функции для выполнения эффекта, немного подождать и через некоторое время повторить вызов. Как же будет выглядеть функция для решения этой задачи?



Всего в трех строках кода мы создали функцию анимации молнии, срабатывающую по таймеру, для первого изображения молнии. Теперь напишите аналогичные функции для других изображений молнии.

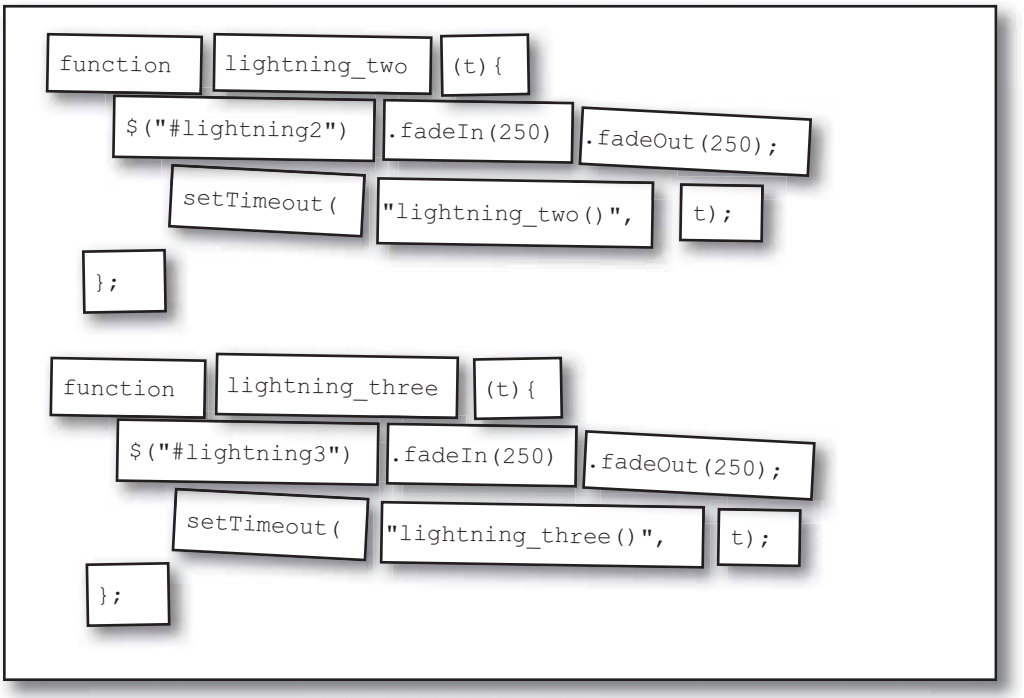
Развлечения с магнитами

Расставьте магниты в правильном порядке. У вас должны получиться функции, повторяющие эффект периодического сверкания молнии, для двух других изображений.

```
function
lightning_two
(t){
};
function
lightning_three
(t){
};
.fadeIn(250)
t);
$("#lightning2")
.fadeOut(250);
setTimeout(
.fadeIn(250)
$("#lightning3")
"lightning_two()",
.fadeOut(250);
"lightning_three()",
setTimeout(
t);
```

Развлечения с магнитами. Решение

Расставьте магниты в правильном порядке. У вас должны получиться функции, повторяющие эффект периодического сверкания молнии, для остальных изображений.



Часто Задаваемые Вопросы

В: Разве `fadeIn().fadeOut()` не то же самое, что `toggle`?

О: Отличный вопрос! Нет, не то же. `toggle` — одиночный метод, который просто переключает выбранный элемент из скрытого состояния в видимое и наоборот, в зависимости от его текущего состояния. Цепочка из `fadeIn` и `fadeOut` создает эффект, при котором выбранный элемент(ы) сначала проявляется, а после завершения эффекта становится невидимым.

В: Новый метод `setTimeout`? Откуда он взялся? Из jQuery или из JavaScript?

О: `setTimeout` — метод JavaScript, который может использоваться для управления некоторыми аспектами анимаций jQuery. Мы поближе познакомимся с `setTimeout` в следующих главах, особенно в главе 7.

Если вы захотите узнать о `setTimeout`, посетите Mozilla Developer's Center: *https://developer.mozilla.org/en/window.setTimeout*. Дополнительную информацию можно найти в превосходной книге Дэвида Фленагана «JavaScript. Подробное руководство».

В: Когда я использую эффект `hide`, элемент просто исчезает. Как замедлить его исчезновение?

О: Чтобы «замедлить» действие `hide`, `show` или `toggle`, передайте значение продолжительности в круглых скобках. Пример использования hide из главы 1:

```
$("#picframe").hide(500);
```

Включение функций в сценарий

Обновите файл сценариев приложения «Собери монстра» кодом из упражнения на предыдущей странице.

В этих строках вызываются функции, выделенные жирным шрифтом в конце.

```
$(document).ready(function(){
    var headclix = 0, eyeclix = 0, noseclix = 0, mouthclix = 0;
    lightning_one(4000);
    lightning_two(5000);
    lightning_three(7000);

    $("#head").click(function(){
        if (headclix < 9){headclix+=1;}
        else{headclix = 0;}
    });

    $("#eyes").click(function(){
        if (eyeclix < 9){eyeclix+=1;}
        else{eyeclix = 0;}
    });

    $("#nose").click(function(){
        if (noseclix < 9){noseclix+=1;}
        else{noseclix = 0;}
    });

    $("#mouth").click(function(){
        if (mouthclix < 9){mouthclix+=1;}
        else{mouthclix = 0;}
    });

});//end doc.onready function

    function lightning_one(t){
        $("#container #lightning1").fadeIn(250).fadeOut(250);
        setTimeout("lightning_one()",t);
    };
    function lightning_two(t){
        $("#container #lightning2").fadeIn("fast").fadeOut("fast");
        setTimeout("lightning_two()",t);
    };
    function lightning_three(t){
        $("#container #lightning3").fadeIn("fast").fadeOut("fast");
        setTimeout("lightning_three()",t);
    };
```

Числа в скобках — длительность задержки (в миллисекундах), передаваемая методу setTimeout. Так мы управляем частотой вспышек.

Мы убрали часть разрывов строк для экономии места. Не беспокойтесь, если в вашем сценарии используется другая разбивка строк.

Определения функций сверкания молний

my_scripts.js

ТЕСТ-ДРАЙВ

Откройте страницу в своем любимом браузере и посмотрите, как работает эффект сверкания молний.

Эффект проявления/исчезновения объединен с методом setTimeout.

Молнии проявляются и исчезают с разными интервалами, чтобы сверкание выглядело реалистичнее.

Итак, к настоящему моменту у вас работают функции click, а три молнии сверкают с разными интервалами. Давайте вернемся к схеме и посмотрим, что еще осталось сделать.

Проект «Собери монстра»

Развлекательное приложение «Собери монстра» адресовано детям целевой возрастной группы. Оно позволяет «собрать» изображение из 10 вариантов головы, глаз, носа и рта. Переключение частей монстра должно сопровождаться анимацией.

Анимация

Модель изменения лица монстра

Модель анимации с молниями

Последняя нерешенная задача этого проекта

Пора сдвигать изображения влево, а ни один из готовых эффектов не подходит. Нельзя ли воспользоваться каким-нибудь другим методом?

Готовые эффекты удобны, но они не позволяют сделать все, что нам нужно.

Пора создать *пользовательский* эффект, который будет сдвигать полосы частей лица монстра влево.

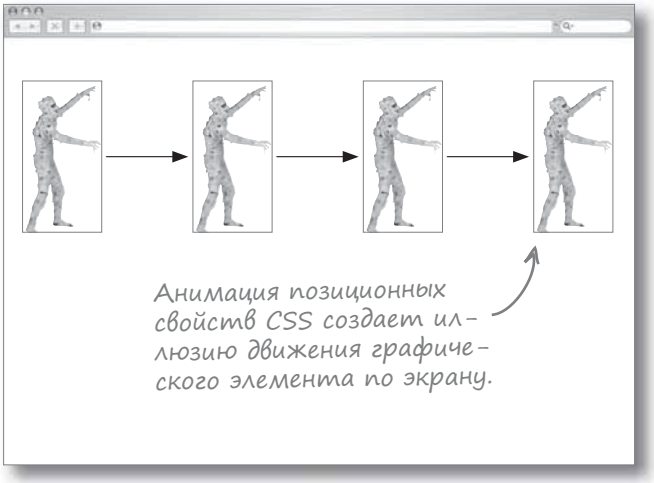
Самодельные эффекты и animate

В jQuery нет методов `slideRight` и `slideLeft`, а на этой стадии проекта без них не обойтись. Означает ли это, что проект «Собери монстра» зашел в тупик?

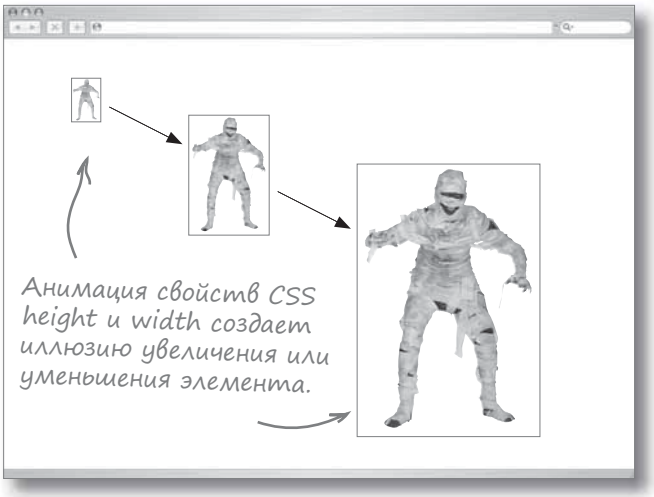
Вовсе нет, в jQuery существует метод `animate` для построения ваших собственных эффектов. С его помощью можно создавать пользовательские анимации, которые по своим возможностям далеко превосходят готовые, стандартные эффекты. Метод `animate` позволяет изменять свойства CSS выбранных элеметов, с возможностью одновременной анимации нескольких свойств.

Давайте посмотрим, что можно сделать с помощью метода `animate`.

Эффекты движения



Эффекты масштабирования



МОЗГОВОЙ ШТУРМ

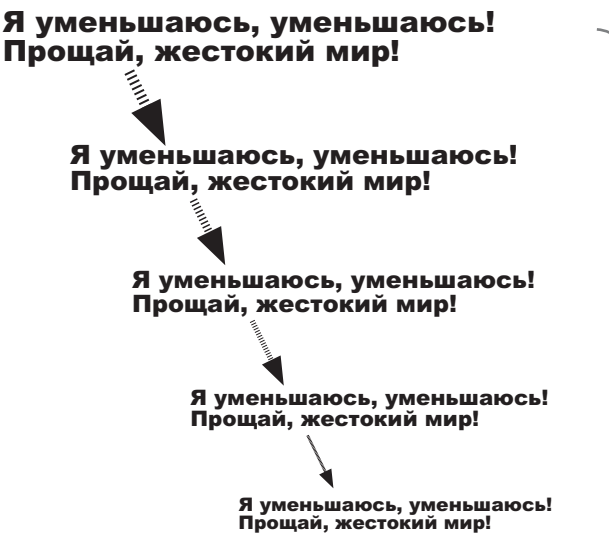
Какое свойство CSS потребуется для анимации сдвига частей лица монстра влево при каждом щелчке?

Что можно анимировать?

Метод `animate` позволяет динамически изменять свойства шрифтов для создания текстовых эффектов. Также возможна одновременная анимация нескольких свойств CSS при одном вызове, что только добавляет интересных возможностей веб-приложениям.

При всей крутизне метода `animate` его возможности небезграничны. В его внутренней реализации используются сложные математические вычисления (о которых вам, к счастью, беспокоиться не нужно), поэтому работать можно только со свойствами CSS, имеющими *числовые значения*. Знайте пределы возможностей, но не ограничивайте свое творческое воображение. Анимация обладает исключительной гибкостью, и с ней можно сделать много интересного.

Текстовые эффекты



Анимация шрифтовых свойств CSS способна создать иллюзию увеличения, уменьшения и полета текста.

Это всего лишь несколько примеров. Для полного описания всех возможностей анимации нам понадобилось бы намного, намного больше места!

Будьте осторожны!

Метод `animate` работает только со свойствами CSS, имеющими числовые значения:

- `borders`, `margin`, `padding`
- element `height`, `min-height` и `max-height`
- element `width`, `min-width` и `max-width`
- `font size`
- `bottom`, `left`, `right` и `top position`
- `background position`
- `letter spacing`, `word spacing`
- `text indent`
- `line height`

Метод animate под увеличительным стеклом

На первый взгляд метод `animate` имеет много общего с другими методами, которые мы уже использовали.

Выбираем элемент(-ы), к которому применяется анимация.

Вызываем метод animate.

Первый параметр animate определяет свойство CSS, к которому применяется анимация.

Второй параметр определяет продолжительность анимации в миллисекундах. С его помощью можно управлять временем завершения анимации.

```
$("#my_div").animate({left:"100px"},500);
```

В этом примере анимация применяется к свойству CSS left...

...для которого задается значение "100px".

Первый аргумент передается всегда, без него анимация работать не будет. Второй параметр необязателен.

Но одна из самых замечательных особенностей `animate` — возможность одновременного изменения нескольких свойств выбранных элементов.

```
$("#my_div").animate({
    opacity: 0,
    width: "200",
    height: "800"
  }, 5000);
```

В этом примере одновременно изменяются свойства прозрачности и размеров элемента.

Будьте осторожны!

Параметры свойств CSS должны задаваться по стандарту DOM, а не по стандарту CSS.

НАПРЯГИ МОЗГИ

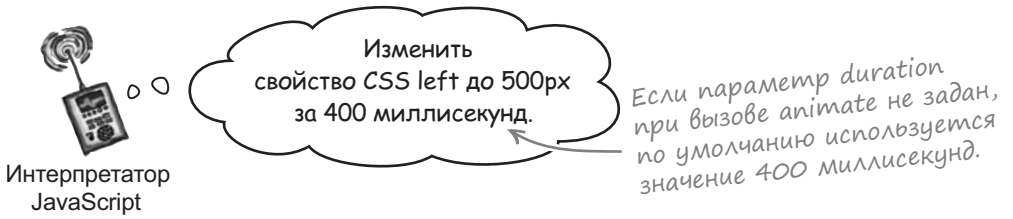
как вы думаете, что при этом происходит в браузере? Как метод `animate` изменяет страницу прямо на глазах у пользователя?

Метод animate изменяет стилевое оформление

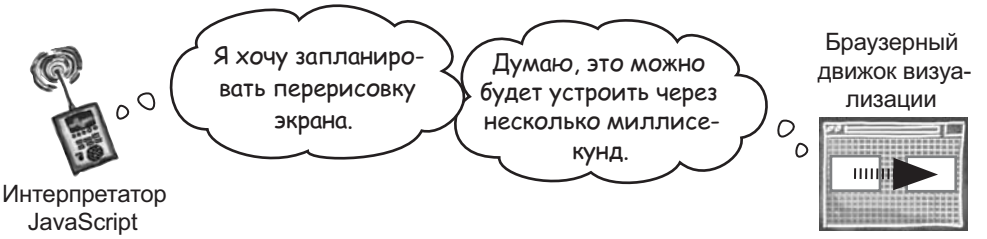
Визуальные эффекты и анимация, которые вы видите на экране телевизора или кинотеатра, используют иллюзию движения. Для достижения этой иллюзии мастера спецэффектов и художники-мультипликаторы берут серию изображений и последовательно воспроизводят их с нужной частотой, вероятно, вы видели, как «оживают» рисунки в уголках тетрадей при быстром пролистывании страниц.

То же самое происходит и на экране браузера, правда, без серии готовых изображений. Вместо этого интерпретатор JavaScript многократно вызывает функцию, изменяющую стилевое оформление элемента. Браузер воспроизводит (*перерисовывает*) изменения на экране. В процессе изменения стилевого оформления создается иллюзия движения или изменения внешнего вида элемента.

1. Интерпретатор JavaScript устанавливает таймер на время выполнения анимации.

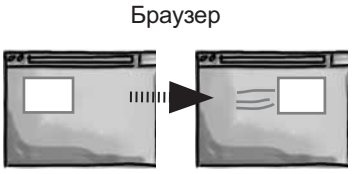


2. Интерпретатор JavaScript приказывает браузерному движку визуализации изменить свойство CSS, заданное в параметрах метода animate. Движок обеспечивает визуальное отображение этих свойств CSS на экране.



3. Интерпретатор многократно вызывает функцию, изменяющую свойство CSS элемента (до завершения работы таймера). При каждом выполнении функции изменения воспроизводятся на экране.

4. Пока браузер отображает изменения в элементе, посетитель видит иллюзию движения.



Соедините каждый фрагмент кода пользовательской анимации с его описанием.

```
$("#my_div").animate({top: "150px"}, "slow")
```

Анимация одновременного изменения левого и правого полей всех абзацев.

```
$("p").animate({
       marginLeft:"150px",
       marginRight:"150px"
});
```

Анимация изменения правой стороны `#my_div` до 0 за половину секунды.

```
$("#my_div").animate({width: "30%"}, 250)
```

Анимация межбуквенных интервалов всех абзацев с продолжительностью по умолчанию (400 миллисекунд).

```
$("#my_div").animate({right: "0"}, 500)
```

Анимация одновременного изменения внутренних полей и ширины `#my_div`.

```
$("p").animate({letterSpacing:"15px"});
```

Анимация изменения положения верхней стороны `#my_div` с продолжительностью slow.

```
$("#my_div").animate({
       padding: "200px",
       width: "30%"
}, "slow")
```

Анимация изменения высоты всех изображений с продолжительностью fast.

```
$("img").animate({height: "20px"}, "fast")
```

Анимация изменения ширины `#my_div`, продолжающегося 1/4 секунды.

Кто и что делает?

Решение

Соедините каждый фрагмент кода пользовательской анимации с его описанием.

```
$("#my_div").animate({top: "150px"}, "slow")
```

Анимация изменения положения верхней стороны #my_div с продолжительностью slow.

```
$("p").animate({
       marginLeft:"150px",
       marginRight:"150px"
});
```

Анимация одновременного изменения левого и правого полей всех абзацев.

```
$("#my_div").animate({width: "30%"}, 250)
```

Анимация изменения ширины #my_div, продолжающегося 1/4 секунды.

```
$("#my_div").animate({right: "0"}, 500)
```

Анимация изменения правой стороны #my_div до 0 за половину секунды.

```
$("p").animate({letterSpacing:"15px"});
```

Анимация межбуквенных интервалов всех абзацев с продолжительностью по умолчанию (400 миллисекунд).

```
$("#my_div").animate({
       padding: "200px",
       width: "30%"
}, "slow")
```

Анимация одновременного изменения внутренних полей и ширины #my_div.

```
$("img").animate({height: "20px"}, "fast")
```

Анимация изменения высоты всех изображений с продолжительностью fast.

Откуда и куда?

Следует помнить, что метод `animate` изменяет *текущее* значение свойства CSS до значения, заданного *первым параметром*. Чтобы пользовательская анимация была эффективной, необходимо тщательно учесть, какое значение хранится в CSS в данный момент. В предыдущем примере значение left элемента `#my_div` изменялось до 100px. То, что при этом произойдет на экране, полностью зависит от текущего значения свойства `left` элемента `#my_div`.

Текущее значение свойства


```
#my_div{
  left: 20px;
}
```

Заданное значение свойства

В анимации используется абсолютная позиция.

```
$("#my_div").animate({left:"100px"});
```

Если свойство имеет другое текущее значение, то и результат будет другим.

Текущее значение свойства


```
#my_div{
  left: 200px;
}
```

Заданное значение свойства


```
$("#my_div").animate({left:"100px"});
```

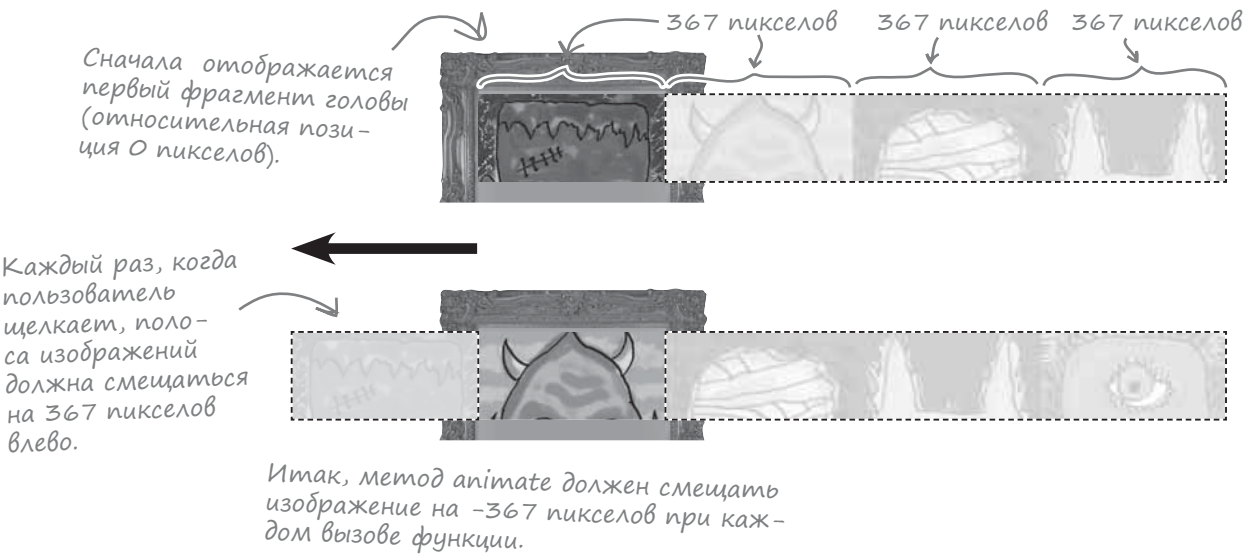
Прекрасно, но как это нам поможет с приложением «Собери монстра»?

Все относительно.

Чтобы части лица монстра скользили именно так, как нам нужно, нужно продумать их *текущие позиции* и то, как они должны изменяться *относительно их положения на момент последнего изменения.*

Абсолютные и относительные перемещения элементов

Вспомните, что полосы изображений, которые должны отображаться, вложены в элемент `div` с идентификатором `#pic_box`. Свойству `left` элемента `div#pic_box` в текущем коде CSS задается значение 91px. Давайте подумаем, как должны двигаться полосы изображений для реализации эффекта горизонтального скольжения.



Вернемся к примеру абсолютного позиционирования на предыдущей странице:

Команда задает свойству left элемента #my_div абсолютное значение 100 пикселов.

```
$("#my_div").animate({left:"100px"});
```

Но как приказать animate смещать элемент на -367 пикселов при каждом вызове?

```
$("#head").animate({left:"???"});
```

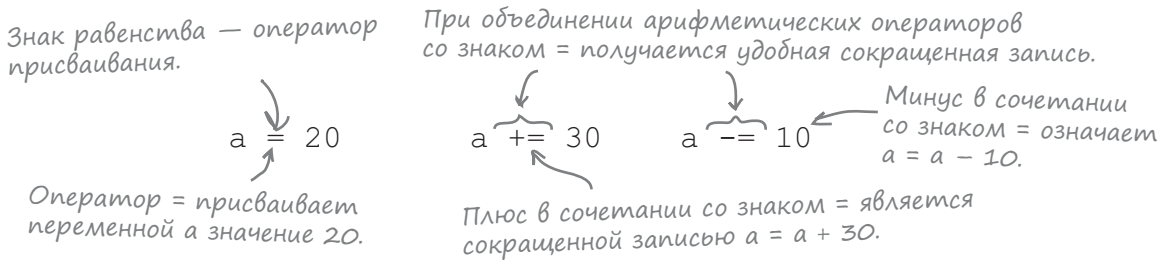
Относительная анимация = каждый раз переместить на столько

При абсолютной анимации элемент перемещается в абсолютную позицию в визуальной системе координат. При относительной анимации перемещение элемента задается относительно *последней позиции, в которую он переместился в процессе анимации.*

Но как обозначить относительное перемещение элемента при вызове метода `animate`?

Комбинированные операторы

В JavaScript существуют специальные операторы, которые сдвигают элемент(ы) на одинаковую величину при каждом вызове `animate` — *операторы присваивания*. Они обычно используются для присваивания, при котором новое значение переменной получается увеличением старого значения на некоторую величину. Впрочем, звучит все это намного сложнее, чем на самом деле.

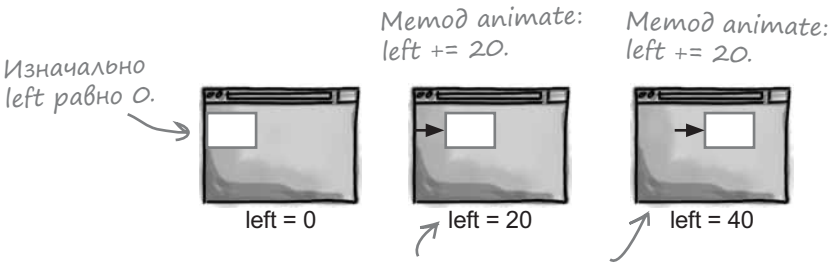


Комбинированные операторы помогают создавать относительную анимацию: новое значение вычисляется по формуле «текущее значение *плюс или минус* смещение в пикселах».

Элемент с идентификатором box смещается на 20 пикселов при каждом вызове метода animate.

```
$("#box").animate({left:"+=20"});
```

Вот что происходит с `#box` при каждом вызове метода `animate`:



Другие комбинированные операторы присваивания:

- `a *= 5` — это сокращенная запись операции `a = a * 5`.
- `a /= 2` — это сокращенная запись операции `a = a/2`.

Упражнение

Напишите пару строк кода jQuery, которые выполняют следующие операции:

1. Элемент `#head` перемещается на 367 пикселов влево при каждом вызове `animate`. Перемещение выполняется за 0,5 секунды.

.....

2. Элемент `#head` перемещается обратно в исходную позицию (`left:0px`). Перемещение выполняется за 0,5 секунды.

.....

Упражнение
Решение

Напишите пару строк кода jQuery, которые выполняют следующие операции:

1. Элемент `#head` перемещается на 367 пикселов влево при каждом вызове `animate`. Перемещение выполняется за 0,5 секунды.

```
$("#head").animate({left:"-=367px"},500);
```

2. Элемент `#head` перемещается обратно в исходную позицию (`left:0px`). Перемещение выполняется за 0,5 секунды.

```
$("#head").animate({left:"0px"},500);
```

Абсолютная анимация переводит голову монстра в исходное состояние, имитируя «перемотку».

Часто Задаваемые Вопросы

В: Некоторые пользователи считают, что анимация мешает нормальному восприятию веб-страницы. Что делать, если я захочу разрешить пользователю отключать анимацию?

О: Справедливое замечание, некоторые пользователи считают, что анимация раздражает и усложняет работу с программой. Если вы хотите, чтобы пользователи могли отключить анимацию, свяжите функцию click кнопки (вы уже знаете, как это делается) со следующей строкой кода:

```
$.fx.off = true;
```

Также для остановки анимации в jQuery может использоваться метод `stop`. Обе возможности описаны на сайте jQuery:

http://api.jquery.com/jQuery.fx.off/
http://api.jquery.com/stop/

В: Вы пишете: «Параметры свойств CSS должны задаваться по стандарту DOM, а не по стандарту CSS». Что бы это значило?

О: Отличный вопрос! Метод `animate` получает параметры по стандарту DOM («синтаксис DOM»), а не по стандарту CSS. Различия лучше пояснить на конкретном примере. Чтобы задать толщину границы `div` в синтаксисе CSS, можно использовать следующую запись:

```
div {
border-style:solid;
border-width:5px;
}
```

Предположим, что вы хотите анимировать изменение толщины границы. В jQuery для этого свойство задается в синтаксисе DOM:

```
$("div").animate({borde
rWidth:30},"slow");
```

Обратите внимание: в синтаксисе CSS имя свойства записывается в виде `border-width`, тогда как в синтаксисе DOM используется запись `borderWidth`.

Различия между двумя вариантами записи более подробно рассматриваются в следующей статье:

http://www.oxfordu.net/webdesign/dom/straight_text.html

В: А если я захочу анимировать изменение цвета?

О: Для выполнения анимации цветовых переходов необходимо использовать библиотеку jQuery UI, которая поддерживает больше эффектов, чем jQuery. Мы будем рассматривать jQuery UI в главе 10, но без рассмотрения эффектов. Когда вы научитесь загружать и включать jQuery UI в ваше веб-приложение, с анимацией цветов особых проблем уже не будет.

Включение вызовов animate в сценарный код

Измените файл сценариев приложения «Собери монстра» — включите в него код, созданный нами в упражнении на предыдущей странице.

Задание!

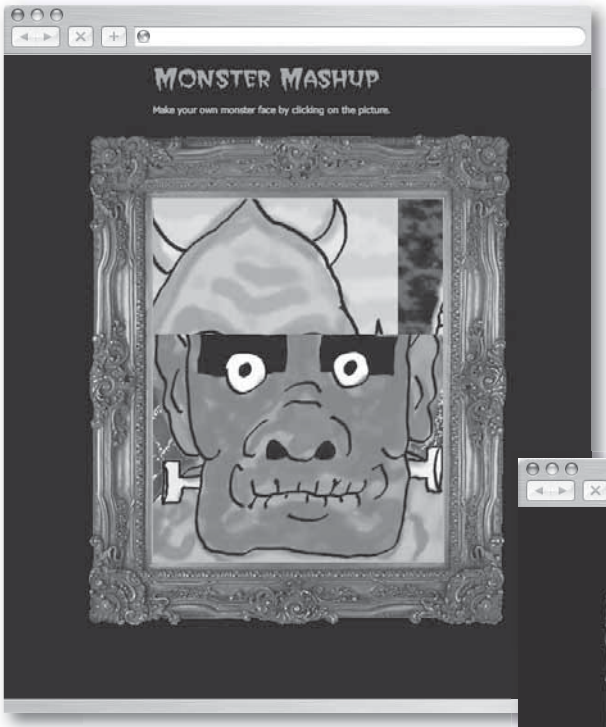
```
$("#head").click(function(){
        if (headclix < 9){
                $(this).animate({left:"-=367px"},500);
                headclix+=1;
        }
        else{
                $(this).animate({left:"0px"},500);
                headclix = 0;
        }
});

$("#eyes").click(function(){
        if (eyeclix < 9){
                $(this).animate({left:"-=367px"},500);
                eyeclix+=1;
        }
        else{
                $(this).animate({left:"0px"},500);
                eyeclix = 0;
        }
});

$("#nose").click(function(){
        if (noseclix < 9){
                $(this).animate({left:"-=367px"},500);
                noseclix+=1;
        }
        else{
                $(this).animate({left:"0px"},500);
                noseclix = 0;
        }
});

$("#mouth").click(function(){
        if (mouthclix < 9){
                $(this).animate({left:"-=367px"},500);
                mouthclix+=1;
        }
        else{
                $(this).animate({left:"0px"},500);
                mouthclix = 0;
        }
```

Здесь можно использовать ключевое слово this, так как мы находимся внутри функции элемента, на котором был сделан щелчок.

my_scripts.js

ТЕСТ-ДРАЙВ

Откройте страницу в своем любимом браузере и убедитесь в том, что все работает правильно.

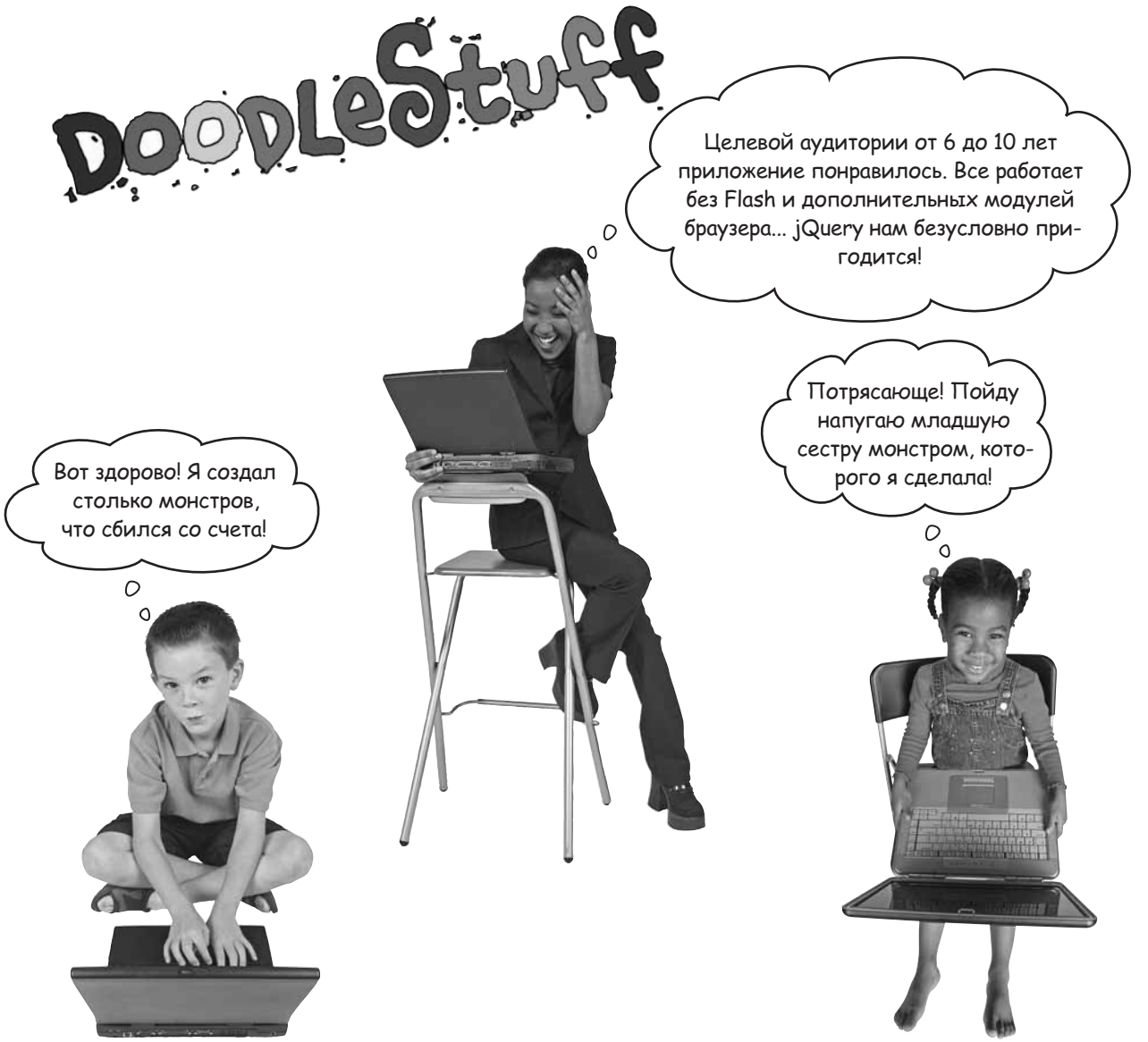


Мы реализовали пользовательский эффект горизонтального скольжения.

Несколько щелчков мышью... и пользователь может создать собственное лицо монстра.

Смотри, мама! Работает без Flash!

Руководитель веб-проектов доволен результатами работы. Мы использовали готовые эффекты jQuery в сочетании с пользовательскими эффектами, созданными специально для конкретного приложения.



ГЛАВА 5

Ваш инструментарий jQuery

Глава 5 осталась позади, а ваш творческий инструментарий расширился: добавились эффекты изменения прозрачности и скольжения, а также пользовательская анимация.

Эффекты изменения прозрачности

Изменение уровня прозрачности элементов:

fadeIn

fadeOut

fadeTo

Эффекты скольжения

Изменение высоты элементов:

slideUp

slideDown

slideToggle

animate

Метод позволяет создавать пользовательские анимации, когда готовых эффектов jQuery недостаточно.

Выполняет анимацию свойств CSS во времени.

Работает только со свойствами CSS, имеющими числовые значения.

Перемещение элементов бывает абсолютным или относительным.

Комбинированные операторы присваивания (+=, -=) упрощают относительную анимацию.

~~Люк~~ jQuery, я твой отец!

Силы jQuery небезграничны. Хотя эта библиотека написана на JavaScript, к сожалению, она позволяет сделать не все, на что способен язык-родитель. В этой главе мы рассмотрим некоторые возможности JavaScript, необходимые для создания современных сайтов, и их применение в jQuery для создания пользовательских списков и объектов, а также средства перебора списков и объектов, которые существенно упростят вашу работу.

Программируем блэкджек

Слухи о вашем мастерстве jQuery распространяются молниеносно. Смотрите, вам пришло сообщение из клуба «Head First», в котором вас просят помочь с разработкой очередного развлекательного приложения.

От: **Head First Lounge**
Тема: **Блэкджек на сайте**

Привет!

Это твои знакомые из клуба «Head First». Надеемся, ты поможешь нам разработать новое приложение для нашего сайта.

Нам бы ОЧЕНЬ хотелось иметь приложение для игры в блэкджек. Справишься?

В идеале игрок щелкает на странице и получает две карты, с возможностью потребовать сдачи дополнительных карт.

Правила нашего клуба, которые нам хотелось бы использовать:

1. Туз ВСЕГДА дает 11 очков (а не 1).

2. Если сумма очков игрока превышает 21 — перебор, игрок проиграл, игра закончена.

3. Если сумма очков на руках игрока 21 — блэкджек, игра закончена.

4. Если сумма очков игрока не превышает 21, а на руках у него пять карт, то игра закончена, а игрок побеждает.

Если ни одно из этих условий не выполнено, игрок может взять следующую карту (или остановить сдачу).

Если оно из правил/условий выполнено, игра закончена.

Игрок должен иметь возможность сбросить текущую игру и начать заново.

Мы не хотим, чтобы игроку приходилось перезагружать страницу вручную. Игра должна все делать сама.

Можешь сделать это для нас? Мы будем вечно благодарны!

--
Клуб
Head First

Джим: Парни, вы прочитали сообщение из клуба «Head First»?

Фрэнк: Да, они хотят разместить на своем сайте упрощенную реализацию блэкджека. Мне кажется, это тривиально.

Джим: Элементарно? Но ведь это блэкджек! Нам понадобится колода карт, дилер, счетчик суммы очков... и так далее. И ты думаешь, что мы с этим справимся?

Джо: Просто не будет, но я думаю, что мы справимся. Как ты и сказал, кто-то должен сдавать карты. Мы напишем для этого функцию. Вероятно, в этой функции можно будет использовать наш старый генератор случайных чисел.

Джим: Кстати, да. А что с картами? Ведь их в колоде 52.

Фрэнк: Мы можем просто составить большой список и каждый раз выбирать случайную карту из списка.

Джим: Но как избежать повторного выбора одной карты?

Фрэнк: Пожалуй, я знаю, как это сделать...

Джим: Ого, впечатляет! А как насчет запоминания того, какие карты уже вышли? И подсчета очков?

Фрэнк: Ты меня подловил. Я пока не знаю, как это сделать.

Джо: Спокойно! В JavaScript и jQuery найдется немало полезных возможностей, которые нам помогут.

Джим: А зачем JavaScript? Мы можем использовать переменные или массивы jQuery для информации о картах. Я думал, нам не придется лезть в дебри JavaScript, раз уж мы используем jQuery...

Фрэнк: Переменные не решат проблем, в них в любой момент времени может храниться только конкретное значение: строка, текст или отдельный элемент страницы. А массивы jQuery позволяют хранить несколько значений, но это могут быть только элементы DOM, возвращаемые селектором.

Джо: Да, ты прав. Нам потребуется что-то более гибкое.

Фрэнк: Скажем, возможность определения *наших собственных* структур и типов переменных.

Джо: Снова в точку! А для создания структур нам понадобится JavaScript...

Объекты и хранение данных

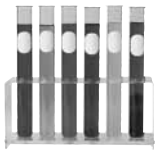
До настоящего момента мы использовали для хранения данных переменные и массивы. Переменные обеспечивают самый простой вариант хранения: одно значение связывается с одним именем. Хранение данных в массивах более эффективно: несколько значений связываются с одним именем.

Переменная

В переменной хранится одно значение.

```
var a = 42;
```

Массив



В массиве хранится несколько значений.

```
var v = [2, 3, 4]
```

Объекты — еще более гибкая схема хранения данных. Они используются, когда для описания некоторой сущности необходимо сохранить несколько переменных. Внутри объекта переменные называются *свойствами*. Объект может содержать функции, предназначенные для взаимодействия со свойствами объекта. Такая функция, созданная внутри объекта, называется *методом*.

Объект

```
planeObject={
    engines:"4",
    type:"passenger",
    propellor: "No"};
```

Данные объекта хранятся в свойствах.

Данные, описывающие самолет, объединяются в одну логическую группу.

```
leopardObject={
      num_spots:"23",
      color:"brown"};
```

Имя свойства связывается...

...со значением.

Для обращения к свойствам объекта используется «точечная запись»:

```
planeObject.engines;
leopardObject.color;
```

Объект *Его свойство*

Объекты используются для хранения нескольких значений, описывающих некоторую сущность.

МОЗГОВОЙ ШТУРМ

Какими свойствами может обладать объект карты?

Построение собственных объектов

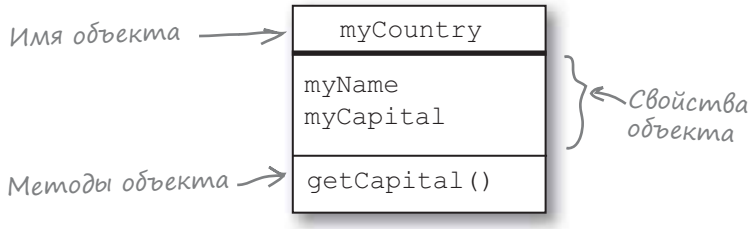
Объекты позволяют создавать ваши собственные переменные, которые работают именно так, как нужно вам. Вы можете создать объект, предназначенный для однократного использования, или же создать «шаблон» для построения объектов, который может использоваться снова и снова. О повторном использовании объектов мы поговорим чуть позже, а пока обсудим создание «одноразовых» объектов, а также рассмотрим некоторые термины и схемы, связанные с объектом.

Существует стандартный способ описания объектов с использованием диаграмм UML (Unified Modeling Language). UML — международный стандарт описания объектов в объектно-ориентированном программировании.

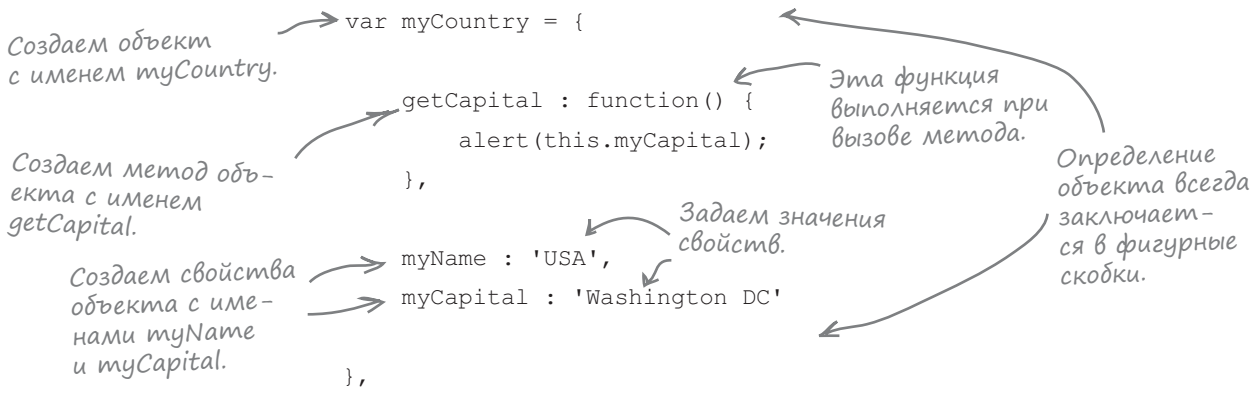
Переменная, связанная с объектом, называется *свойством* объекта. Функция, связанная с объектом, называется *методом* объекта. Объекты, рассчитанные на одноразовое использование, создаются ключевым словом `var`, как и все остальные переменные, встречавшиеся нам до настоящего момента.

UML-диаграмма объекта

Эта структура покажет, как устроен ваш объект. Прежде чем браться за написание кода, полезно свериться с диаграммами UML.



А вот как этот объект определяется в программном коде:



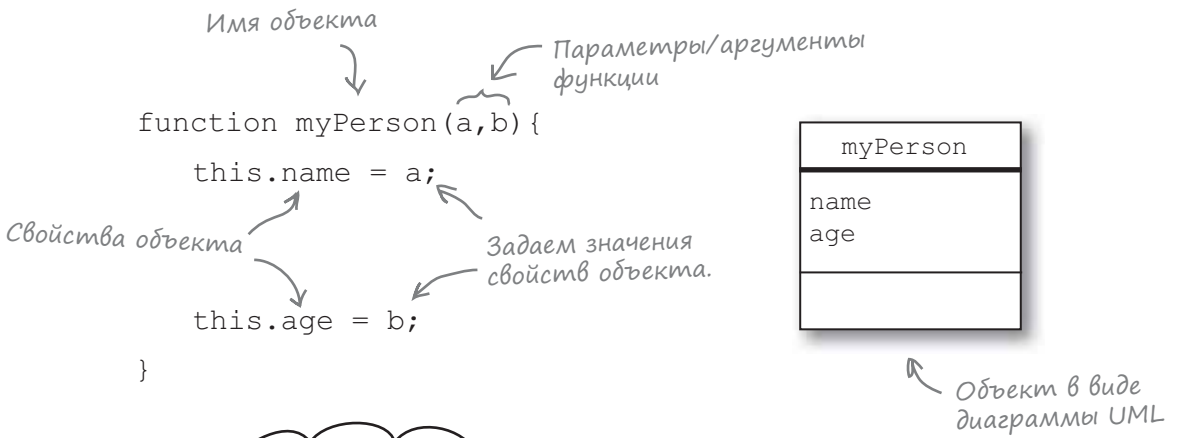
Оказывается, практически все данные в jQuery и JavaScript — объекты.

Это относится к элементам, массивам, функциям, числам и даже строкам, — все они имеют методы и свойства.

Создание объектов для повторного использования

У объектов есть одна интересная особенность: они могут иметь одинаковую структуру, но содержать разные значения свойств (или переменных). Как и в случае с функциями, рассчитанными на многократное использование (как мы делали в главе 3), можно создать «шаблон» (или *конструктор*) объекта, чтобы использовать его повторно. Конструктор объекта также может использоваться для создания *экземпляров* объекта.

Конструктор — всего лишь функция, поэтому для создания конструктора объекта следует использовать ключевое слово `function` вместо ключевого слова `var`. Далее ключевое слово `new` используется для создания нового экземпляра объекта.



МОЗГОВОЙ ШТУРМ

Как вы думаете, где можно использовать такие объекты?

Взаимодействие с объектами

После создания экземпляра объекта (независимо от того, создали ли его вы или кто-то другой) все операции с ним выполняются при помощи оператора «точка» (.). Чтобы вы лучше поняли, как работает этот механизм, давайте повнимательнее присмотримся к только что определенным объектам `myCountry` и `myPerson`.

Оператор «точка» используется по схеме имя_объекта.имя_метода() или имя_объекта.имя_свойства().

Вызов метода getCapital

На странице будет выведен текст «Washington DC».

```
myCountry.getCapital();
```

Обращение к свойству myName

```
alert(myCountry.myName);
```

На странице выводится текст «USA».

Новый экземпляр объекта создается командой «new».

Значения, переданные в аргументах функции new, задают значения свойств.

Создание новых экземпляров объекта myPerson с именами actor1 и actor2

```
var actor1 = new myPerson('Jack', '42');
var actor2 = new myPerson('Mary', '33');
```

Обращение к свойству name экземпляра объекта myPerson с именем actor1

```
alert(actor1.name);
alert(actor2.age);
```

Обращение к свойству age property экземпляра объекта myPerson с именем actor2 (в данном случае значение равно 33)

На странице выводится текст «Jack».

Кажется, я начинаю понимать... Я могу создать объект для представления карт в колоде?

Да! Отличная мысль.

Давайте подготовим страницу HTML и посмотрим, как создать объект, представляющий карту.

Подготовка страницы

Создайте файлы HMTL и CSS по приведенной ниже информации. Не забудьте создать файл *my_scripts.js* в папке *scripts*. На следующих страницах мы включим в него достаточно большой объем кода. Графику для всей главы можно загрузить по адресу: *http://thinkjquery.com/chapter06/images.zip*.


```
#controls{
    clear:both;
}
#my_hand{
    clear:both;
    border: 1px solid gray;
    height: 250px;
    width: 835px;
}
h3 {
    display: inline;
    padding-right: 40px;
}
.current_hand{
     float:left;
}
```

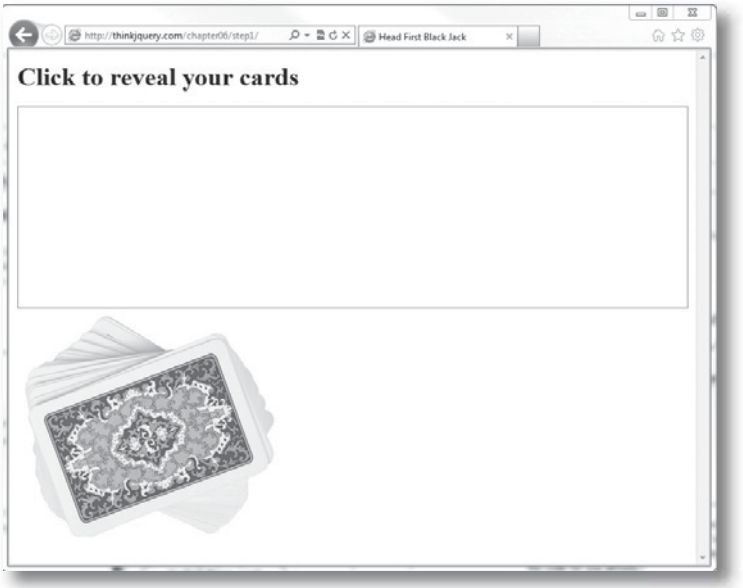
my_style.css

```
<!DOCTYPE html>
<html>
    <head>
        <title>Head First Black Jack</title>
        <link href="styles/my_style.css" rel="stylesheet">
    </head>
    <body>
        <div id="main">
            <h1>Click to reveal your cards</h1>
            <h3 id="hdrTotal"></h3><h3 id="hdrResult"></h3>
            <div id="my_hand">
            </div>
            <div id="controls">
                <div id="btnDeal">
                    <img src="images/deck_small.jpg">
                </div>
            </div>
        </div>
        <script src="scripts/jquery-1.6.2.min.js"></script>
        <script src="scripts/my_scripts.js"></script>
    </body>
</html>
```

index.html

Тест-драйв

Откройте страницу *index.html* в своем любимом браузере, ознакомьтесь с общей структурой страницы.



Упражнение

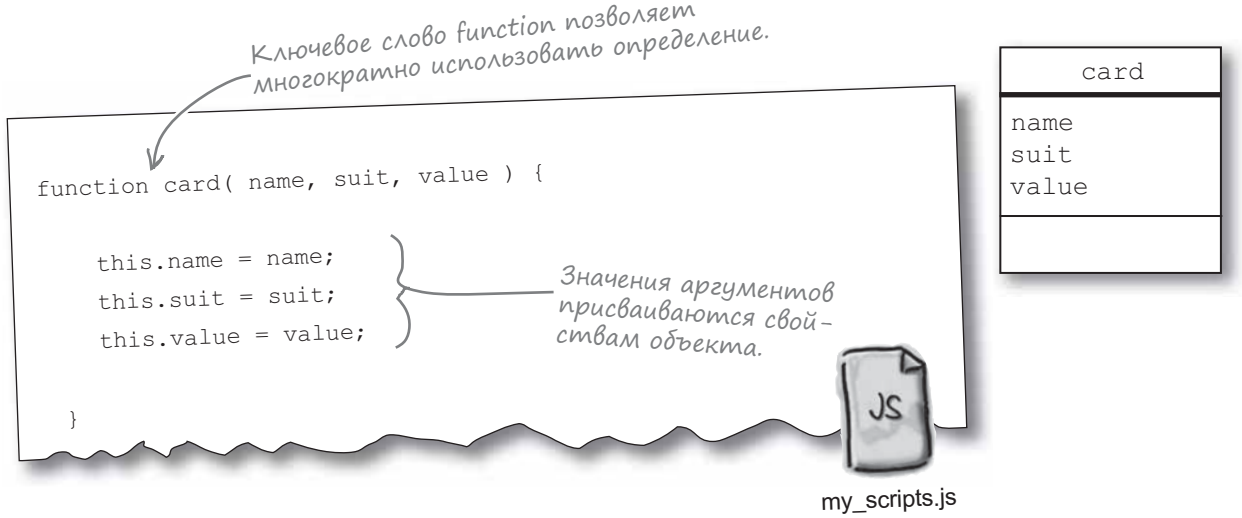
Используя приведенную ниже UML-диаграмму объекта card, определите объект `card` с тремя параметрами `name`, `suit` и `value`. Значения параметров должны присваиваться соответствующим свойствам объекта. Этот конкретный объект пока не содержит ни одного метода. Часть кода мы уже написали за вас.

```
function card(                          ) {



}
```

card
name suit value

Упражнение
Решение

Ниже приведено определение объекта `card`. Включите его в файл *my_scripts.js* (раздел `$(document).ready(function(){ });`). Пока это весь код, который содержится в файле.



Часто Задаваемые Вопросы

В: Чем «одноразовые» объекты отличаются от «многоразовых», пригодных для повторного использования?

О: «Одноразовый» объект — всего лишь особая форма объявления переменной, пригодной для хранения нескольких блоков информации. «Многоразовые» объекты представляют собой шаблоны, по которым можно создать сколько угодно экземпляров нужного объекта, причем каждый экземпляр содержит свой набор информации, описывающей объект.

В: Кажется, вы используете разные способы задания свойств. Это так?

О: Да, так. Значения свойств могут задаваться как с использованием оператора присваивания (=), так и с использованием двоеточия (:), как мы сделали в примере. Оба способа являются действительными и взаимозаменяемыми.

В: Что еще мне нужно знать об объектах?

О: Объекты JavaScript — достаточно сложная тема. Позднее в этой книге мы будем использовать запись JSON (JavaScript Object Notation). В этой записи обращение к свойствам производится несколько иным способом, который также может применяться к объектам JavaScript (так называемое «обращение по ключу»). Вместо записи:

```
object.my_property
```

используется запись:

```
object['my_property']
```

Результат остается неизменным — обращение к значению свойства

```
my_property.
```

В: Откуда появился стандарт UML?

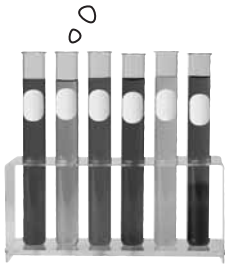
О: Стандарт UML родился в середине 90-х, когда компании пытались разработать наглядный способ описания объектов. С тех пор UML прошел несколько циклов развития, а несколько конкурирующих частных фирм пытались утвердить свою версию в качестве общепринятого стандарта. К счастью, единый стандарт существует, и каждый разработчик со знанием UML сможет читать и понимать диаграммы и информацию из других источников UML.

Получается, объект card будет очень полезным, но нам придется как-то отслеживать отыгранные карты, верно?

Верно.

Нам понадобится механизм хранения сданных карт и обращения к ним при сдаче. К счастью, мы уже знаем, как это делается...

Это снова я! Мы уже встречались в главе 4.



И снова массивы

Несколько элементов данных можно объединить в массив. Ячейки массива необязательно имеют отношение друг к другу, но такой способ хранения данных упрощает работу с ними. В главе 4 мы уже видели, как селектор jQuery возвращает данные и сохраняет их в массиве. Сейчас мы воспользуемся JavaScript, чтобы работа с массивами стала еще эффективнее.

Ячейки массива могут содержать данные любого типа: строки, числа, объекты, даже элементы HTML! Существует несколько способов создания массивов:

Создание пустого массива ключевым словом new

```
var my_arr1 = new Array();
```

Создание массива ключевым словом new с заданием значений ячеек

```
var my_arr2 = new Array('USA', 'China', 'Japan', 'Ireland');
```

```
var my_arr3 = ['USA', 'China', 'Japan', 'Ireland'];
```

Создание массива без ключевого слова new, но с заданием значений ячеек (перечисляемых в квадратных скобках [ ])

И как упоминалось ранее, массивы тоже являются объектами, а значит, обладают методами и свойствами. Популярное свойство объекта `length` возвращает количество ячеек в массиве. Для обращения к свойству `length` используется запись вида `имя_массива.length`.

Будьте осторожны!

Не существует различий между способами создания массивов.

На практике они выбираются вперемешку в зависимости от назначения массива. Чтобы найти список методов объекта массива, используйте в поисковой системе строку «JavaScript array methods».

Обращение к ячейкам массива

В отличие от создания массивов, существует только один способ обращения к информации, хранящейся в ячейках. Индексирование массивов начинается с 0 — первой ячейке массива соответствует порядковый номер (или индекс) 0. Мы уже использовали индексы в главе 3; если вы забыли, что это такое, вернитесь к ней и освежите память.

Первой ячейке массива соответствует индекс 0, второй — индекс 1 и т. д.

При обращении к ячейкам массива всегда используются квадратные скобки [ ].

```
alert( my_arr2[0] ) ;
// В окне сообщения выводится строка USA
```

Имя массива, созданного на с. 257

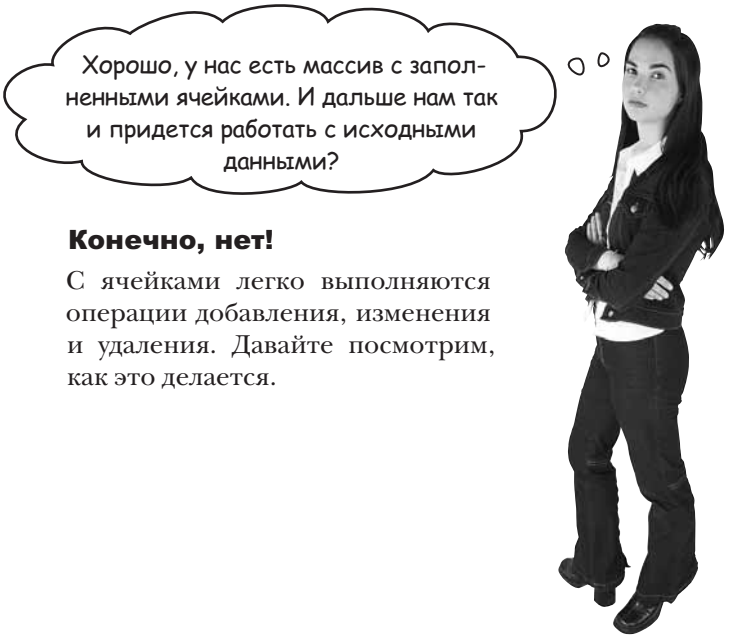
При обращении к ячейке массива по индексу кавычки не нужны.

```
alert( my_arr3[2] );
// В окне сообщения выводится строка Japan
```

При попытке обращения по несуществующему индексу происходит ошибка.

```
alert( my_arr1[1] );
// Ошибка, пустая ячейка
```

Индекс ячейки определяет ее позицию в массиве.



Конечно, нет!

С ячейками легко выполняются операции добавления, изменения и удаления. Давайте посмотрим, как это делается.

Добавление и обновление ячеек

В массив можно включить сколько угодно элементов. В приведенных ранее примерах в массивы `my_arr2` и `my_arr3` были изначально сохранены данные, а массив `my_arr1` остался пустым. Однако в массиве можно добавлять новые и изменять существующие ячейки, причем эти операции также осуществляются по индексу. Рассмотрим несколько разных способов обновления массивов:

Задаем значение первой ячейки массива my_arr1.

```
my_arr1[0] = "France";
alert( my_arr1[0] );
// Выводится строка France
```

Добавляем вторую ячейку в массив my_arr1.

```
my_arr1[1] = "Spain" ;
```

Обновляем значение первой ячейки массива my_arr1.

```
my_arr1[0] = "Italy" ;
alert( my_arr1[0] );
// Выводится строка Italy
```

Обновляем значение третьей ячейки массива my_arr3.

```
my_arr3[2] = "Canada";
alert( my_arr3[2] );
// Выводится строка Canada
```

Упражнение

В файле *my_scripts.js*, после кода определения объекта `card`, создайте массив с именем `deck`, содержащий все 52 карты стандартной колоды.

Вы можете использовать уже созданный объект `card` и многократно вызывать конструктор с нужными параметрами для создания каждой карты — от туза до короля с разными мастями (трефы, черви, бубны, пики) и ценностью (туз — 11 очков, двойка — 2 очка и т. д.).

Упражнение
Решение

Файл *my_scripts.js* содержит массив `deck` с 52 картами стандартной колоды, а также определение объекта `card`. Для создания каждой карты используйте объект `card` и вызывайте конструктор с нужными параметрами.

Задаем имя массива.

При создании каждого объекта card передаются три параметра.

```
var deck = [
  new card('Ace', 'Hearts',11),
  new card('Two', 'Hearts',2),
  new card('Three', 'Hearts',3),
  new card('Four', 'Hearts',4),

  new card('King', 'Hearts',10),
  new card('Ace', 'Diamonds',11),
  new card('Two', 'Diamonds',2),
  new card('Three', 'Diamonds',3),

  new card('Queen', 'Diamonds',10),
  new card('King', 'Diamonds',10),
  new card('Ace', 'Clubs',11),
  new card('Two', 'Clubs',2),

  new card('King', 'Clubs',10),
  new card('Ace', 'Spades',11),
  new card('Two', 'Spades',2),
  new card('Three', 'Spades',3),

  new card('Jack', 'Spades',10),
  new card('Queen', 'Spades',10),
  new card('King', 'Spades',10)
];
```

Не забудьте, что список значений должен быть заключен в квадратные скобки.

my_scripts.js

Но в массиве так много карт! Похоже, для их извлечения нам придется снова написать здоровенный блок кода. Тоже мне развлечение!

Необязательно.

Конечно, к ячейкам можно обращаться по индексу, но мы также можем последовательно перебрать все ячейки при помощи конструкции, напоминающей `each` (см. главу 3), и нам не придется писать длинный код для каждой карты.

Пора познакомиться с циклами...

Повторение операций

В нашем приложении придется часто сдавать и возвращать карты в массив. К счастью, в JavaScript как раз для подобных ситуаций существуют *циклы*. Самое интересное, что вы ими уже пользовались: в главе 3 метод jQuery `each` использовался для перебора элементов на основании селектора jQuery. На этот раз у нас больше свободы выбора, так как в JavaScript существует несколько разновидностей циклов, которые имеют слегка различающийся синтаксис и предназначены для разных целей.

Циклы `for` идеально подходят, когда код выполняется заранее известное количество раз. Это число должно быть известно до начала цикла, иначе выполнение может затянуться до бесконечности. Цикл `for` может не выполняться или много раз выполняться в зависимости от значений переменных.

Цикл `for`:

Циклы for выполняют фрагмент кода заранее определенное количество раз.

Цикл `do...while` выполняет код минимум один раз, а потом продолжает выполнять его до тех пор, пока не будет выполнено некоторое условие (скажем, логическая переменная сменит состояние или счетчик в коде достигнет заданного значения). Цикл `do...while` выполняется один и более раз.

Цикл `do...while`:

Циклы do...while выполняют фрагмент кода и повторяют его, пока не будет выполнено некоторое условие.

Все циклы независимо от их типа состоят из четырех частей.

1. **Инициализация**
   Выполняется один раз в начале цикла.

2. **Проверка условия**
   Принятие решения об остановке цикла или выполнении следующей итерации (обычно в зависимости от значения переменной).

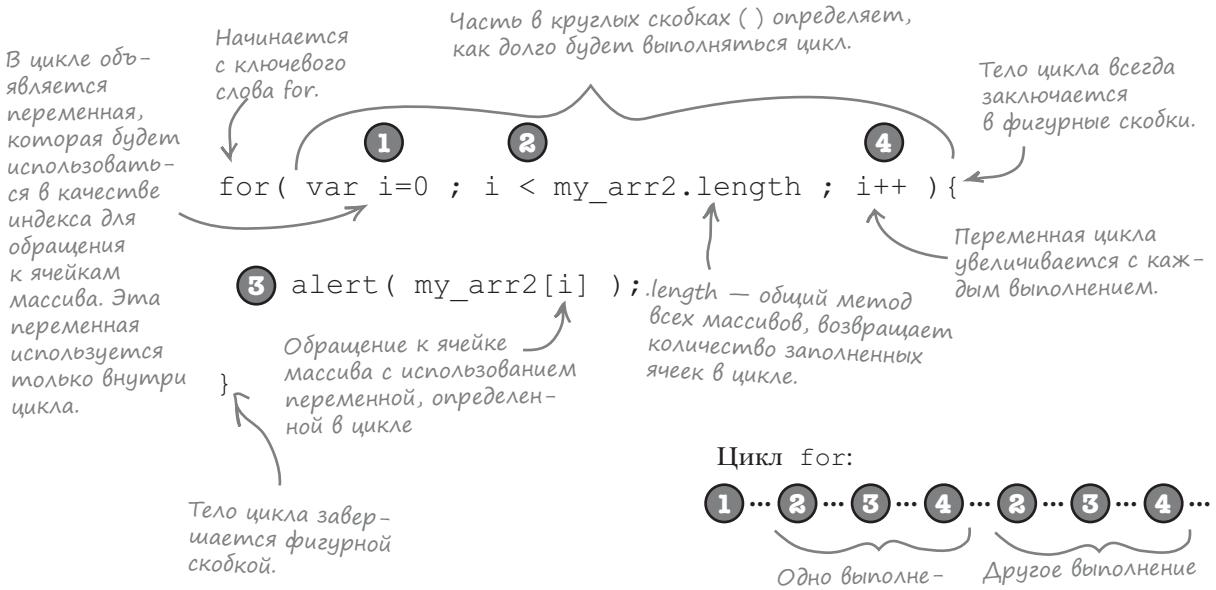
3. **Тело цикла**
   Основной код, повторяемый при каждом выполнении цикла.

4. **Обновление**
   Обновление переменных, используемых при проверке условия.

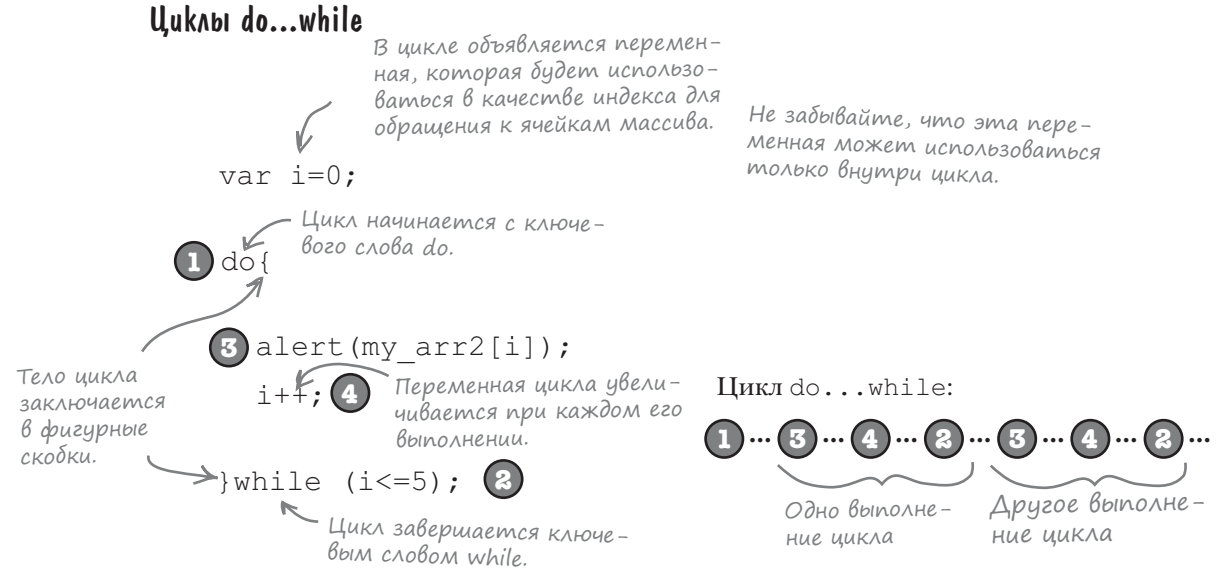
Циклы под увеличительным стеклом

Если присмотреться к упоминавшимся разновидностям циклов, видно, что все они содержат четыре основных компонента, но в несколько ином порядке. Порядок следования этих компонентов отражает основные различия между циклами.

Циклы for



Циклы do...while



Фрэнк: У нас есть массив объектов `card`, но нам ведь нужно извлечь случайную карту при сдаче, верно?

Джо: Да, и к счастью, мы уже написали функцию `getRandom` в главе 3. Она будет возвращать случайное число каждый раз, когда потребуется извлечь карту из массива.

Джим: И что мы будем делать с этой картой?

Фрэнк: Информацию о ней нужно сохранить. Мы должны просуммировать очки и определить, дошел игрок до 21 очка или нет.

Джо: И еще одна причина: одну карту нельзя сдать дважды, поэтому мы должны убедиться в том, что карта не была сдана ранее.

Джим: Карты будут сохраняться в переменных?

Фрэнк: Лучше воспользоваться массивом...

Джо: Точно! Причем хранить сами карты необязательно; достаточно хранить их индексы. По ним можно будет проверить, содержится ли карта в массиве использованных карт `used_cards`.

Джим: И как мы узнаем, содержится ли значение в массиве?

Фрэнк: При помощи вспомогательного метода jQuery с именем `inArray`.

Джо: Звучит разумно. Но мне кажется, что для этого нам стоит воспользоваться несколькими функциями. Нам понадобится случайное число от 0 до 51; потом нужно проверить, не было ли оно использовано ранее. Если карта уже использована, повторяем попытку, а если нет — извлекаем из колоды соответствующую карту и сохраняем ее индекс. Также необходимо вывести изображение карты для игрока.

Джим: И как мы выведем изображение карты?

Фрэнк: Вообще-то изображения у нас уже есть, они упорядочены по мастям и типам, мы воспользуемся этими атрибутами объекта card для отображения графики на странице.

Джо: Точно. Мы создадим элемент DOM и присоединим его к элементу div `my_hand`, уже находящемуся на странице.

Фрэнк: Наши усилия по созданию объекта `card` уже окупаются... За дело!

Поиск иголки в стоге сена

Часто бывает нужно проверить, содержится ли некоторое значение в массиве или нет, например, чтобы избежать дублирования данных или предотвратить повторное включение значения в массив. Например, это особенно важно при использовании массива для хранения покупательской корзины или списка предполагаемых покупок.

Создание массива со всякой всячиной

А если мы захотим узнать, в какой ячейке массива haystack хранится это значение?

```
var haystack = new Array('hay', 'mouse', 'needle', 'pitchfork')
```

jQuery предоставляет целый набор вспомогательных методов, которые позволяют более эффективно решать типичные задачи. В частности, существуют функции для проверки типа браузера, используемого посетителем, для получения текущего времени, для слияния массивов, для удаления дубликатов из массивов и т. д.

Вспомогательный метод, который пригодится в нашей конкретной ситуации, называется `inArray`. Он возвращает позицию (индекс) ячейки, в которой был найден искомый элемент, если он был найден, конечно. Если найти значение в массиве не удалось, метод возвращает `-1`. Как и другие вспомогательные методы, `inArray` не требует передачи селектора, он вызывается прямо для функции jQuery или ее сокращения.

Создаем переменную для хранения возвращаемого значения функции.

Искомое значение

Массив, в котором ищется значение

```
var index = $.inArray( value, array );
```

Сокращенная запись jQuery

Вызов вспомогательного метода inArray

Искомое значение

Массив, в котором ищется значение

```
var needle_index = $.inArray( 'needle', haystack );
```

МОЗГОВОЙ ШТУРМ

Где именно в нашем приложении следует проверять, использовалось ли ранее полученное значение?

Развлечения с магнитами

Расставьте магниты так, чтобы сложить из них код функций для нашего приложения. В готовом коде должны содержаться определения двух функций (`deal` и `hit`), а также слушателя события click для элемента с идентификатором `btnDeal` и определение нового массива `used_cards`, в котором хранится информация о сданных ранее картах.

```
var used_cards = new _____________();
function _____________{
    for(var i=0;i<2;i++){
        hit();
    }
}
function getRandom(num){
    var my_num = Math.floor(_____________________*num);
    return my_num;
}
function ___________{
    var good_card = false;
    do{
        var index = ___________________(52);
        if( !$.inArray(index, _________________ ) > -1 ){
            good_card = true;
            var c = deck[ index ];
            _________________[used_cards.length] = index;
            hand.cards[hand.cards.length] = c;
            var $d = $("<div>");
            $d.addClass("current_hand")
              .appendTo(_________________);
            $("<img>").appendTo($d)
                    .attr( ________ , 'images/cards/' + c.suit + '/' + c.name + '.jpg' )
                    .fadeOut('slow')
                    .fadeIn('slow');
        }
    }___________(!good_card);
    good_card = false;
}
$("#btnDeal").click( _________________(){
    deal();
    $(this).toggle();
});
```

getRandom

deal()

used_cards

'src'

hit()

while

Array

"#my_hand"

function

used_cards

Math.random()

my_scripts.js

Развлечения с магнитами. Решение

Ниже приведен завершенный код функций `deal` и `hit`, а также код слушателя события click для элемента с идентификатором `btnDeal` и определение массива `used_cards`, в котором хранится информация об уже сданных картах.

```
var used_cards = new [Array]();
function [deal()] {
    for(var i=0;i<2;i++){
        hit();
    }
}
function getRandom(num){
    var my_num = Math.floor([Math.random()]*num);
    return my_num;
}
function [hit()] {
    var good_card = false;
    do{
        var index = [getRandom](52);
        if( !$.inArray(index, [used_cards]) > -1 ){
            good_card = true;
            var c = deck[ index ];
            [used_cards][used_cards.length] = index;
            hand.cards[hand.cards.length] = c;
            var $d = $("<div>");
            $d.addClass("current_hand")
              .appendTo([#my_hand]);
            $("<img>").appendTo($d)
                    .attr( ['src'], 'images/cards/' + c.suit + '/' + c.name + '.jpg' )
                    .fadeOut('slow')
                    .fadeIn('slow');
        }
    }[while](!good_card);
    good_card = false;
}
$("#btnDeal").click( [function](){
    deal();
    $(this).toggle();
});
```

Создаем массив для хранения использованных карт.

Используем цикл for для двукратного вызова функции hit.

Снова функция getRandom!

Проверяем, была ли использована только что отобранная карта, при помощи функции inArray.

Управляющая переменная цикла do...while

Получаем карту из массива deck.

Включаем индекс карты в массив used_cards.

Заставляем карту «мигнуть» на экране.

Строим путь к изображению из свойств объекта card.

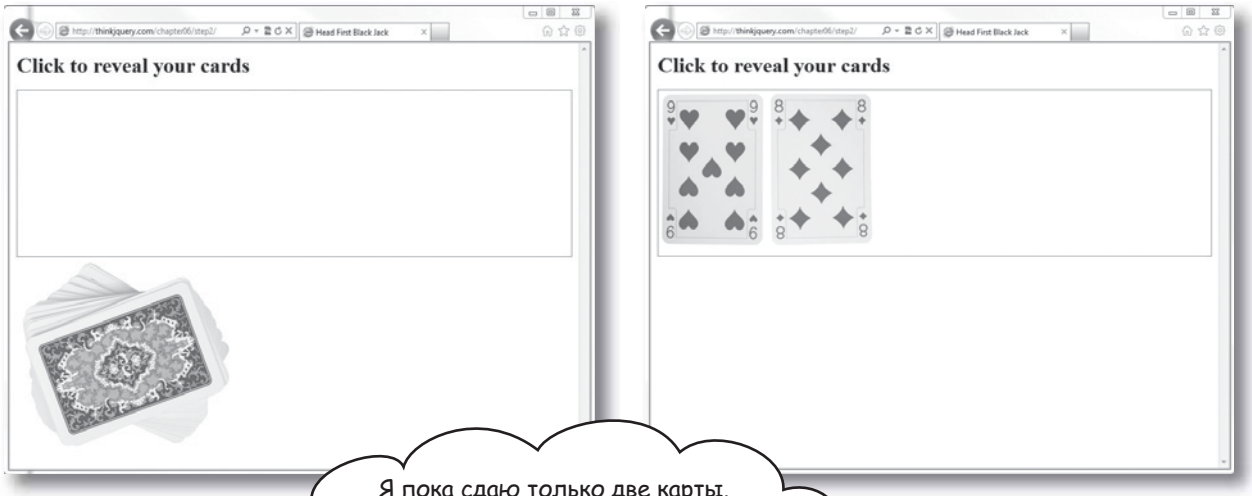
Если карта уже использована, повторяем попытку.

Функция deal вызывается в обработчике click.

my_scripts.js

Тест-драйв

Включите код из упражнения с магнитами в файл *my_scripts.js* (после определения массива `deck`), откройте страницу в браузере. Щелкните на изображении колоды, чтобы получить карты.



Я пока сдаю только две карты. А значит, я почти всегда выигрываю! Но пусть игра будет честной. Сможете ли вы организовать сдачу дополнительных карт?

Конечно, мы можем сдать дополнительные карты из колоды при помощи уже написанной функции `hit`.

Нам только придется придумать, как запускать эту функцию — по щелчку на кнопке или еще как-нибудь. Также появляется дополнительная проблема: необходимо запоминать и подсчитывать сданные карты, чтобы проверить возможный выигрыш или перебор.

Мозговой штурм

Как вы думаете, что следует использовать для хранения всей этой информации?

Готово к употреблению

Вы уже стали настоящим профессионалом по стилю и структуре страниц, так что мы просто приводим обновленный код файлов *index.html* и *my_style.css* для сравнения. После включения нового кода HTML и CSS на странице появляются новые элементы, вскоре мы свяжем их с программным кодом.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Head First Black Jack</title>
        <link href="styles/my_style.css" rel="stylesheet">
    </head>
    <body>
        <div id="main">
            <h1>Click to reveal your cards</h1>
            <h3 id="hdrTotal"></h3>
            <h3 id="hdrResult"></h3>
            <div id="my_hand">
            </div>
            <div id="controls">
                <div id="btnDeal">
                    <img src="images/deck_small.jpg">
                </div>
                <div id="btnHit">
                    <img src="images/deck_small.jpg">
                </div>
                <div id="btnStick">
                    <img src="images/stick_small.jpg">
                </div>
            </div>
        </div>
        <script src="scripts/jquery-1.6.2.min.js"></script>
        <script src="scripts/my_scripts.js"></script>
    </body>
</html>
```

Добавляем новые элементы для сдачи карт и отказа.

index.html

Добавляем код CSS для новых элементов.

```
#controls{
    clear:both;
}

.current_hand{
    float:left;
}

#my_hand{
    clear:both;
    border: 1px solid
gray;
    height: 250px;
    width: 835px;
}

#btnHit, #btnStick,
#btnRestart{
    display:none;
    float:left;
}

h3 {
    display: inline;
    padding-right: 40px;
}
```

my_style.css

Часто Задаваемые Вопросы

В: Существуют ли другие типы циклов, о которых мне следует знать?

О: Да, существуют. Цикл `while` очень похож на `do...while`, но условие в нем проверяется в начале. Также существует цикл `for...in`, который перебирает свойства объекта и последовательно возвращает их значения.

В: Итак, я запустил цикл на выполнение. Могу ли я прервать его в середине?

О: Да, при помощи очень простой команды: `break`. При выполнении этой команды в любой точке цикла его выполнение прерывается, а управление передается команде, следующей за циклом.

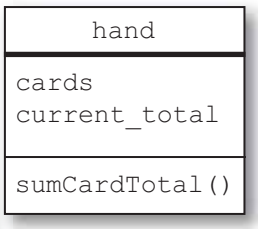
В: А что такое `appendTo`? Раньше мы видели только `append`. Эти методы чем-то различаются?

О: При использовании `append` селектор, вызывающий метод, является контейнером, в который вставляется содержимое. С другой стороны, при использовании `appendTo` содержимое задается до метода (либо в виде селекторного выражения, либо в виде разметки HTML, созданной «на месте») и вставляется в заданный контейнер.

Упражнение

Используя приведенную ниже диаграмму UML, создайте «одноразовый» объект с именем `hand`. Свойство `cards` должно содержать новый пустой массив. Свойству `current_total` присваивается значение 0. Метод `sumCardTotal` должен перебирать все карты в свойстве `cards`, суммировать их значения и задавать полученную сумму как значение свойства `current_total`. Далее свойство `current_total` используется для задания значения элемента с идентификатором `hdrTotal`. Мы уже написали небольшую часть кода объекта за вас.

```
var hand = {
    cards : new Array(),
    current_total : 0,

    sumCardTotal: function(){









    }
};
```



Упражнение
Решение

Итак, теперь у нас имеется объект `hand` со свойством `card` (массив), и функция, которая перебирает массив `card`, получает ценность текущей карты и обновляет сумму очков.

```
var hand = {
    cards : new Array(),
    current_total : 0,

    sumCardTotal: function(){
        this.current_total = 0;
        for(var i=0;i<this.cards.length;i++){
            var c = this.cards[i];
            this.current_total += c.value;
        }
        $("#hdrTotal").html("Total: " + this.current_total );
    }
};
```

Создаем новый массив для свойства card.

Задаем свойству current_total значение 0.

Перебираем ячейки массива card.

Получаем текущую карту из массива.

Прибавляем очки к current_total.

Выводим сумму на экран в элементе hdrTotal.

hand
cards current_total
sumCardTotal()

Но никто не сообщает мне результат игры. В итоге приложение сдаст мне все карты, разве не так?

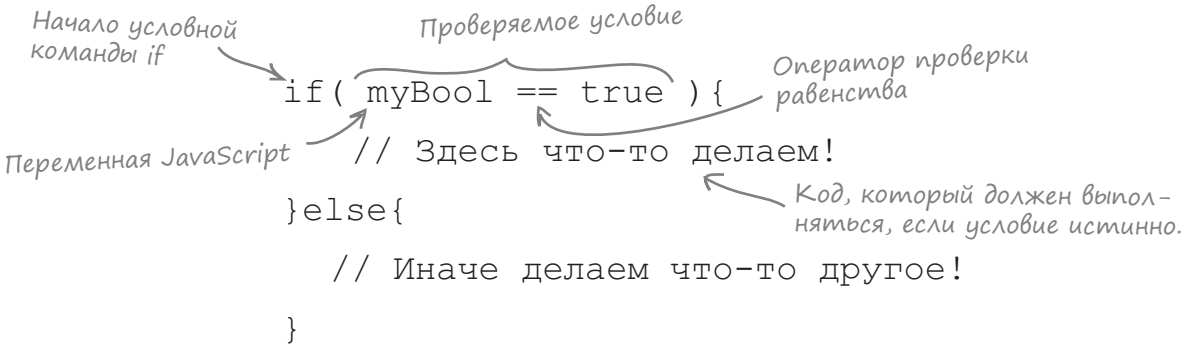
Нет, это определенно нежелательно.

По правилам, описанным в сообщении из клуба «Head First», победитель определяется по нескольким различным критериям. Давайте еще раз вспомним эти критерии.

1. Если сумма очков игрока превышает 21, то происходит перебор. Игрок проиграл, игра закончена.
2. Если сумма очков игрока равна точно 21, то игрок набрал блэкджек, игра закончена.
3. Если сумма очков не превышает 21, но игрок уже получил пять карт, то игра закончена, а игрок побеждает.
4. Если ни одно из этих условий не выполнено, игрок может потребовать сдачи следующей карты (или остановить сдачу).

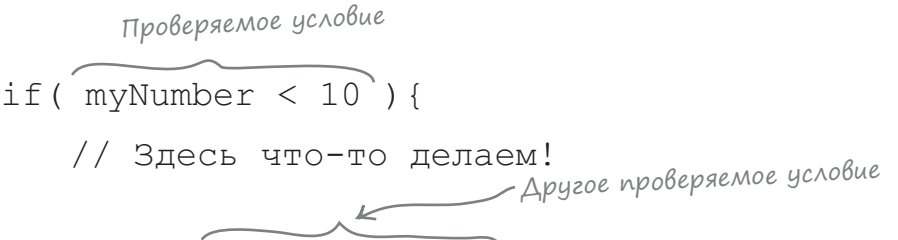
Пора принимать решение... снова!

В главе 3 были рассмотрены условные конструкции для выполнения разного кода в зависимости от решений, принимаемых в коде на основании уже имеющейся информации.




```
if( myBool == true ){
   // Здесь что-то делаем!
}else{
   // Иначе делаем что-то другое!
}
```

Оказывается, существует возможность проверки сразу нескольких условий. Для этого следует объединить `if` и `else` в составную команду `else if`. Давайте посмотрим, как это делается.

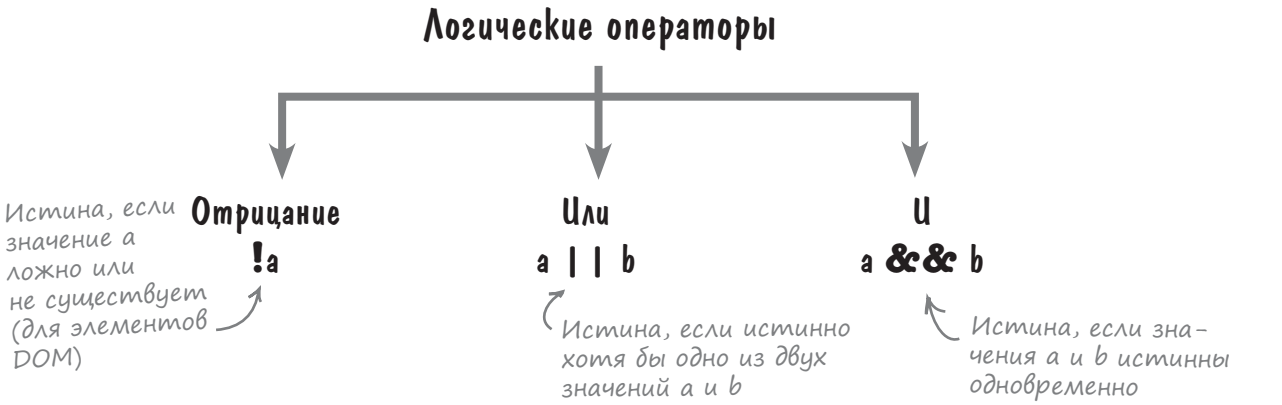



```
if( myNumber < 10 ){
   // Здесь что-то делаем!
}else if( myNumber > 20 ){
   // Иначе делаем что-то другое!
}else{
   // Наконец, делаем что-то третье!
}
```

Где в нашем коде вы бы использовали конструкцию `if` / `else if` / `else`?

Операторы сравнения и логические операторы

Условные конструкции (такие как `if/else` или `do...while`) должны принимать правильное решение на основании проверяемого условия. Для этого в них используются специальные *операторы сравнения* и *логические операторы*. В JavaScript существуют семь разных операторов сравнения и три логических оператора, а также оператор сокращенной записи конструкции `if/else`, называемый *тернарным* оператором. Некоторые из этих операторов уже встречались нам ранее, а ниже приведен полный список.



Возьми в руку карандаш

Измените объект `hand` так, чтобы он проверял значение свойства `current_total` на соответствие критериям игры (если забыли правила, вернитесь и перечитайте исходное сообщение). Ниже приведен существующий объект, а также основа кода, который вам необходимо написать.

```
var hand = {
    cards : new Array(),
    current_total : 0,

    sumCardTotal: function(){
        this.current_total = 0;
        for(var i=0;i<this.cards.length;i++){
            var c = this.cards[i];
            this.current_total += c.value;
        }
        $("#hdrTotal").html("Total: " + this.current_total );
        if(this._____ > 21){
            $("#btnStick").trigger("click");
            $("#hdrResult").html("BUST!");
        } _____(this.current_total _____ ){
            $("#btnStick").trigger("click");
            $("#hdrResult").html("BlackJack!");
        }else if( ____.current_total ___21 ____this.cards.length == 5){
            $("#btnStick").trigger("click");
            $("#hdrResult").html("5 card trick!");
        } _____
            // Продолжаем игру! :)
        }
    }
};
```

my_scripts.js

Возьми в руку карандаш
Решение

В метод `sumCardTotal` включена логика проверки очков сданных карт. Даже в этом простом приложении количество условных и логических операторов получается весьма значительным.

```
var hand = {
    cards : new Array(),
    current_total : 0,

    sumCardTotal: function(){
        this.current_total = 0;
        for(var i=0;i<this.cards.length;i++){
            var c = this.cards[i];
            this.current_total += c.value;
        }
        $("#hdrTotal").html("Total: " + this.current_total );
        if(this.current_total > 21){
            $("#btnStick").trigger("click");
            $("#hdrResult").html("BUST!");
        }else if(this.current_total == 21){
            $("#btnStick").trigger("click");
            $("#hdrResult").html("BlackJack!");
        }else if(this.current_total <= 21 && this.cards.length == 5){
            $("#btnStick").trigger("click");
            $("#hdrResult").html("BlackJack - 5 card trick!");
        }else{
            // Продолжаем игру! :)
        }
    }
};
```

Проверяем условие: значение current_total больше 21.

Проверяем условие: равно ли свойство current_total 21

Проверяем условие: свойство current_total меньше либо равно 21 и игрок уже получил 5 карт.

В противном случае не делаем ничего!

my_scripts.js

Но мы пока не можем вызвать новые функции, потому что они существуют сами по себе, верно?

Да, работа еще не закончена.

В разметке HTML есть все необходимое для сдачи начальных карт, получения очередной карты и завершения игры. Просто эти возможности еще не включены в работу программы. И не забывайте, что при каждой сдаче новой карты необходимо вызывать метод суммирования очков.

Развлечения с магнитами

Расставьте магниты так, чтобы у вас получилась функция, добавляющая несколько слушателей событий в приложение. Слушатели должны подключаться к элементам с идентификаторами `btnHit` и `btnStick`. Кнопка `btnHit` должна сдавать очередную карту, а кнопка `btnStick` — останавливать игру. Также после сдачи каждой карты должен вызываться метод `sumCardTotal`. Мы также включили завершающий фрагмент функции `hit`, в который тоже необходимо внести изменения.

```
    }while(!_____________);
    good_card = false;
    hand._____________();
}
$("#btnDeal").click( _____________(){
    deal();
    $(this).toggle();
    $("#btnHit")._____________();
    $("#btnStick").toggle();
});
$("_____________").click( function(){
    hit();
});
$("#btnStick").click( function(){
    $("#hdrResult").html(_____________);
});
```

function

toggle

good_card

'Stick!'

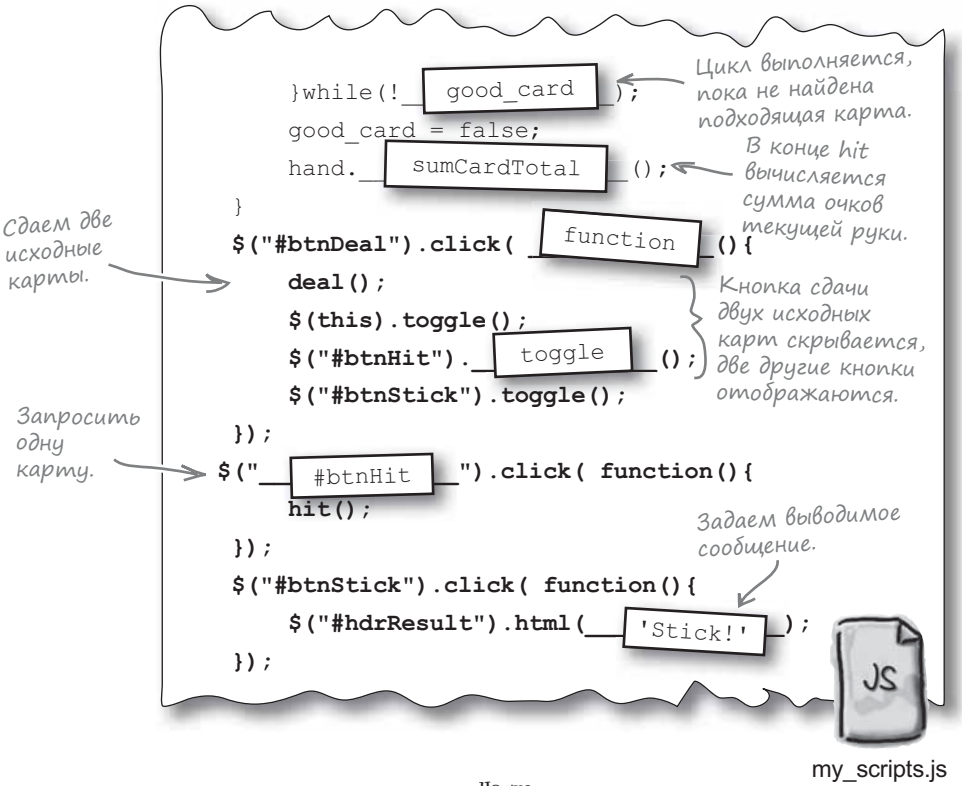
#btnHit

sumCardTotal

my_scripts.js

Развлечения с магнитами. Решение

Следующий код добавляет слушателей событий к кнопкам, а также вызывает метод `sumCardTotal` после каждой сдачи карты.



Часто Задаваемые Вопросы

В: Существуют ли другие способы сравнения значений в JavaScript?

О: Существуют, впрочем, это не столько способ сравнения значений как таковой, а скорее способ принятия решений в зависимости от значений переменных. Мы имеем в виду конструкцию switch, которая может проверять много разных условий. Если вам приходится писать длинные серии команд `if` / `else if` / `else`, их часто удается заменить конструкцией switch.

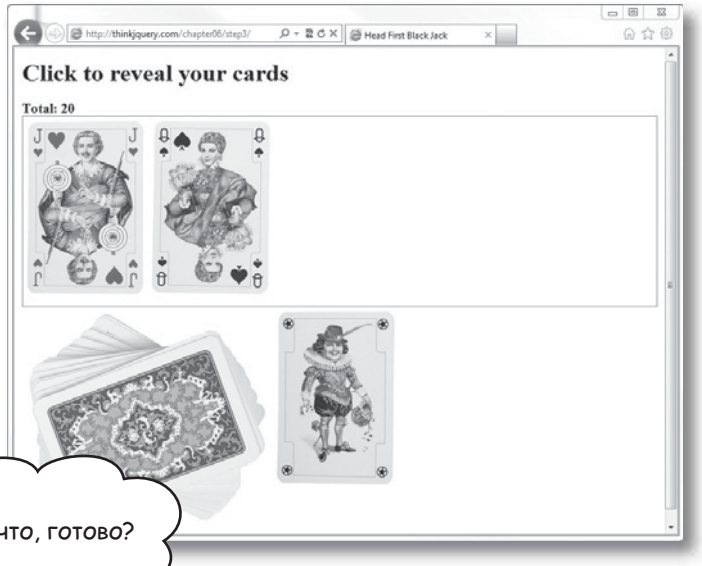
В: Вы упоминали о сокращенной форме команды `if/else`. Как она выглядит?

О: При использовании так называемого тернарного оператора условие отделяется от выполняемого действия вопросительным знаком:

```
a > b ? код_true : код_false
```

Тест-драйв

Включите весь новый код в файл *my_scripts.js* после функции `hit` (не забудьте обновить завершающий фрагмент самой функции `hit`). Откройте страницу в своем любимом браузере.



Фрэнк: Не торопись. Нам еще нужно реализовать возможность сброса, чтобы после окончания партии игрок мог начать игру заново без перезагрузки страницы.

Джо: Также необходимо проследить за тем, чтобы игроки не получали карты из предыдущих раздач. Из массивов нужно все удалить.

Джим: Но как мы это сделаем? Мы динамически добавляли элементы HTML, включали в массивы новые значения. И теперь это все нужно стереть?

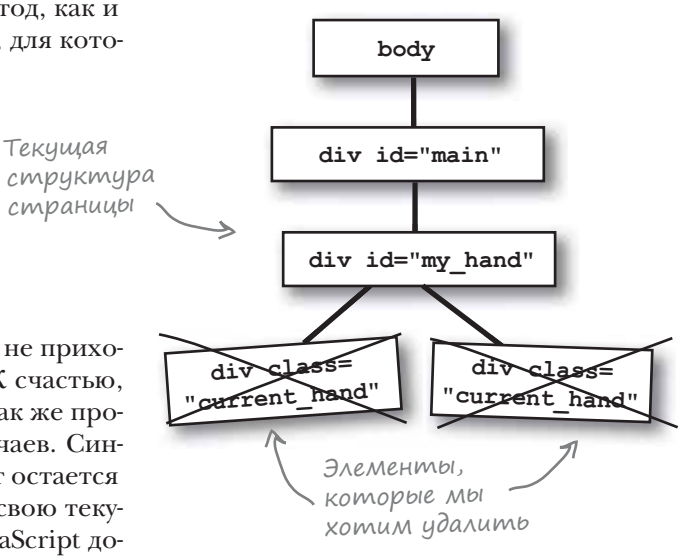
Фрэнк: Да. Способы решения этих задач слегка различаются, но нам действительно придется все стереть.

Джо: Кажется, я знаю! В jQuery для очистки содержимого элементов используется метод `empty`. Для очистки массивов существует несколько решений, но не все они одинаково работают во всех браузерах. Какой же способ лучше подойдет нам?

Стирание информации в jQuery

Помните, как в главе 2 мы использовали метод jQuery `remove` для безвозвратного удаления конкретного элемента и всех его потомков из DOM? Такое решение отлично работает, если вы хотите удалить родительский элемент. Но если основной элемент должен остаться на своем месте, а вы хотите только стереть его содержимое, лучше воспользоваться методом jQuery `empty`. Этот метод, как и `remove`, требует указания селектора, но элемент, для которого он был вызван, остается на своем месте.

```
$("#my_hand").empty();
```

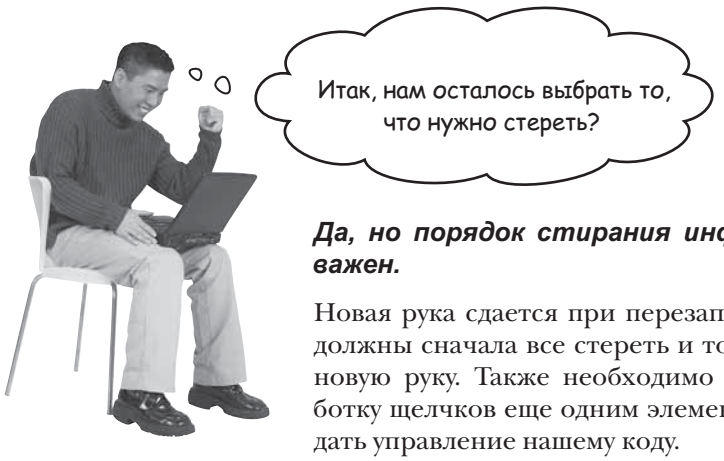


А в JavaScript еще проще

Часто мы используем jQuery для того, чтобы нам не приходилось писать несколько строк кода JavaScript. К счастью, некоторые операции выполняются в JavaScript так же просто, как в jQuery; это как раз один из таких случаев. Синтаксис слегка отличается, но конечный результат остается неизменным, а вам не приходится отслеживать свою текущую позицию в DOM. Для очистки массива в JavaScript достаточно сделать свойство length равным 0:

```
used_cards.length = 0;
```

Проще простого, верно?



Да, но порядок стирания информации тоже важен.

Новая рука сдается при перезапуске, поэтому мы должны сначала все стереть и только *потом* сдать новую руку. Также необходимо обеспечить обработку щелчков еще одним элементом, чтобы передать управление нашему коду.

Возьми в руку карандаш

Включите в файл *index.html* новый элемент `div` (по аналогии с другими управляющими элементами `div`), присвойте ему идентификатор `btnRestart`. Разместите в нем элемент изображения с источником *restart_small.jpg* из папки *images*.

Также включите в файл *my_scripts.js* слушателя события click для элемента `btnRestart`., который очищает элемент `my_hand`, массив `used_cards` и массив `cards` в объекте `hand`. Кроме того, он должен отображать новый элемент `div` с идентификатором `result` и свой элемент `div`, стирать разметку HTML элементов `hdrResult`, а в завершение отображать элемент `btnDeal` и инициировать для него событие click.

index.html

```
        <div id="btnStick">
            <img src="images/stick_small.jpg">
        </div>
        _____________________
            ____________________________________
        ______
        <div id="_______"><img src="" id="imgResult">_______
    </div>
</div>
<script src="scripts/jquery-1.6.2.min.js"></script>
```

my_scripts.js

```
    $("#hdrResult").html('Stick!');
});
$("#btnRestart").click( function(){
    _____________.toggle();
    $(this).______________
    $("#my_hand")._________
    $("#hdrResult").html('');
    used_cards. _________= 0;
    ____________.length = 0;
    hand. _____________  = 0;

    $("#btnDeal").toggle()
                .________('click');
});
```

Возьми в руку карандаш
Решение

Итак, у нас есть кнопка сброса, которая возвращает все элементы в исходное состояние. Теперь немного волшебства JavaScript со свойством `length`, и дело сделано!

```
        <div id="btnStick">
            <img src="images/stick_small.jpg">
        </div>
        <div id="btnRestart">
            <img src="images/restart_small.jpg">
        </div>
        <div id="result"><img src="" id="imgResult"></div>
    </div>
</div>
<script src="scripts/jquery-1.6.2.min.js"></script>
```

Кнопка сброса начинает игру заново.

Результат делается более наглядным.

index.html

```
    $("#hdrResult").html('Stick!');
    $("result").toggle();
});
$("#btnRestart").click( function(){
    $("#result").toggle();
    $(this).toggle();
    $("#my_hand").empty();
    $("#hdrResult").html('');
    used_cards.length = 0;
    hand.cards.length = 0;
    hand.current_total = 0;

    $("#btnDeal").toggle()
                 .trigger('click');
});
```

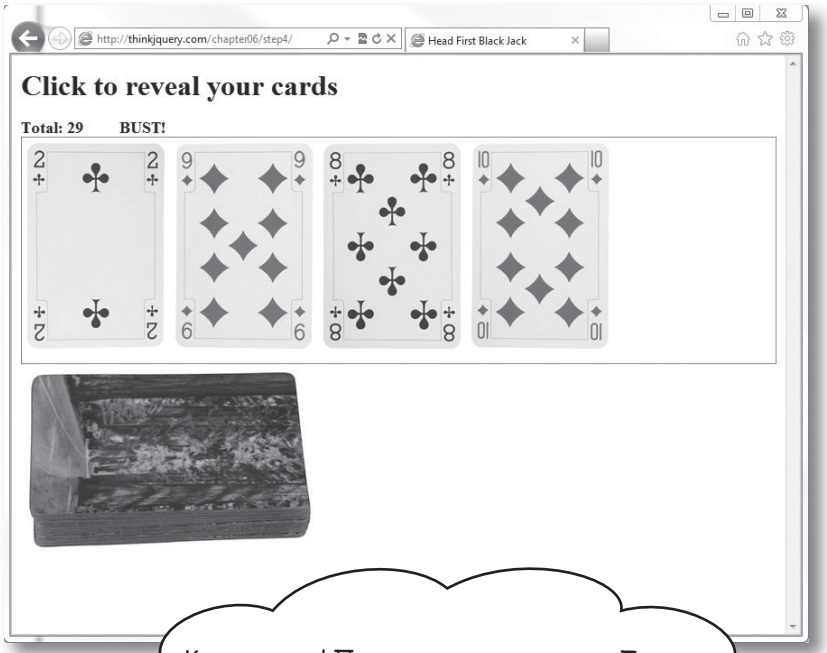
Все элементы возвращаются в исходное состояние.

Имитируем щелчок на элементе btnDeal.

my_scripts.js

ТЕСТ-ДРАЙВ

Добавьте событие click для элемента `btnRestart` в файл *my_scripts.js*. Также не забудьте включить дополнительную разметку HTML в файл *index.html*.



Чтобы было красивее

Включите в файл *my_scripts.js* новую функцию `end`, которая вызывается кнопкой `btnStick`, а также внесите изменения в логике `sumCardTotal`. Обновленную версию файла *my_style.css* можно загрузить по адресу: *http://thinkjquery.com/chapter06/end/styles/my_style.css*.


```
        if(this.current_total> 21){
            $("#btnStick").trigger("click");
            $("#imgResult").attr('src','images/x2.png');
            $("#hdrResult").html("BUST!")
                           .attr('class', 'lose');
        }else if(this.current_total == 21){
            $("#btnStick").trigger("click");
            $("#imgResult").attr('src','images/check.png');
            $("#hdrResult").html("BlackJack!")
                           .attr('class', 'win');
        }else if(this.current_total <= 21 && this.cards.length == 5){
                $("#btnStick").trigger("click");
                $("#imgResult").attr('src','images/check.png');
                $("#hdrResult").html("BlackJack - 5 card trick!")
                               .attr('class', 'win');
        }else{}
        $("#hdrTotal").html("Total: " + this.current_total );
    }
};
function end(){
    $("#btnHit").toggle();
    $("#btnStick").toggle();
    $("#btnRestart").toggle();
}
$("#btnStick").click( function(){
    $("#hdrResult").html('Stick!')
                   .attr('class', 'win');
    $("#result").toggle();
    end();
});
```

Назначаем атрибуту src элемента imgResult изображение в зависимости от результата игры.

Задаем класс заголовка в зависимости от результата игры.

Переключаем состояние всех элементов управления для завершения игры.

Вызываем функцию end для завершения игры (после отказа от сдачи карты).

my_scripts.js

ТЕСТ-ДРАЙВ

Обновите код метода `sumCardTotal` объекта `hand` в файле *my_scripts.js*. Также не забудьте загрузить новую версию файла *my_style.css и установить ее на место старой*.

ГЛАВА 6

Ваш инструментарий jQuery и JavaScript

Глава 6 осталась позади. В ней ваш творческий инструментарий расширился: в нем появились объекты JavaScript, массивы и циклы.

Что будем делать?

От объединения пользовательских эффектов jQuery с функциями JavaScript ваш код (и ваши веб-приложения) становится более эффективным и более мощным. В этой главе мы займемся совершенствованием эффектов jQuery посредством обработки событий браузера, использования временных функций и улучшения общей структуры и возможностей повторного использования пользовательских функций JavaScript.

Надвигается буря

Веб-приложение «Собери монстра» из главы 5 пользовалось большим успехом у детей и родителей. Но похоже, где-то допущена ошибка, из-за которой с эффектом молний возникают проблемы. Джилл из DoodleStuff сообщает о проблемах, а заодно просит доработать приложение.

Джилл, директор по контролю качества

Мы обнаружили, что когда пользователь открывает в браузере новую вкладку, оставляя приложение «Собери монстра» открытым в другой вкладке, при возвращении на старую вкладку молнии бьют непрерывно. Кажется, с приложением что-то не так!

Когда посетитель запускает приложение «Собери монстра»...

...а потом открывает новую вкладку, остается на ней на несколько минут...

...а потом возвращается на вкладку приложения «Собери монстра», молнии непрерывно сверкают, словно эффекты не поспевают друг за другом.

МОЗГОВОЙ ШТУРМ

Попробуйте воспроизвести проблему. Что произошло с молниями? Почему после перехода на другую вкладку исчезли паузы между разрядами?

Мы создали монстра... функцию-монстра

Функция сверкания молнии, написанная нами в главе 5, таит скрытый подвох. Она продолжает работать непрерывно, даже если пользователь ушел со страницы. А когда пользователь возвращается на вкладку, таймер пытается наверстать упущенное время и быстро перерисовывает молнию на экране. Похоже, таймер работает не так, как мы хотели. Что же произошло?

```
function lightning_one(t){
     $("#lightning1").fadeIn(250).fadeOut(250);
     setTimeout("lightning_one()",t);
};
```

Задержка в миллисекундах

Здесь мы указываем интерпретатору JS, что функция должна вызывать сама себя снова и снова.

В JavaScript функция обычно определяется в одном месте, а вызывается в другом. В данном случае мы вызываем функцию из нее самой.

Метод setTimeout приказывает интерпретатору JS выполнить функцию, а затем повторить ее выполнение с некоторой задержкой.

В главе 5 нам был нужен механизм многократного вызова метода с задержкой. Решая эту задачу, мы незаметно для себя создали новую проблему: функция продолжает выполняться и тогда, когда окно теряет фокус (то есть когда посетитель открывает новую вкладку, покидая текущую страницу).

Функция, которая бесконечно и *бесконтрольно* запускает сама себя? Слишком сложно и небезопасно! Как вернуть происходящее под контроль?

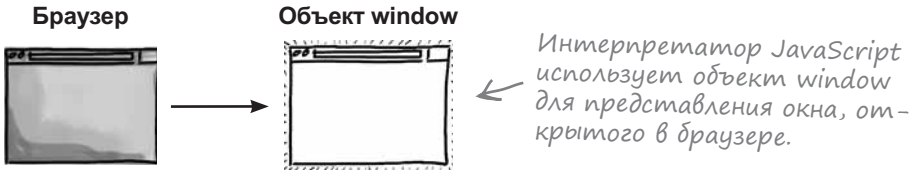
Будьте осторожны!

С функциями, вызывающими сами себя, нужно быть *очень* осторожным.

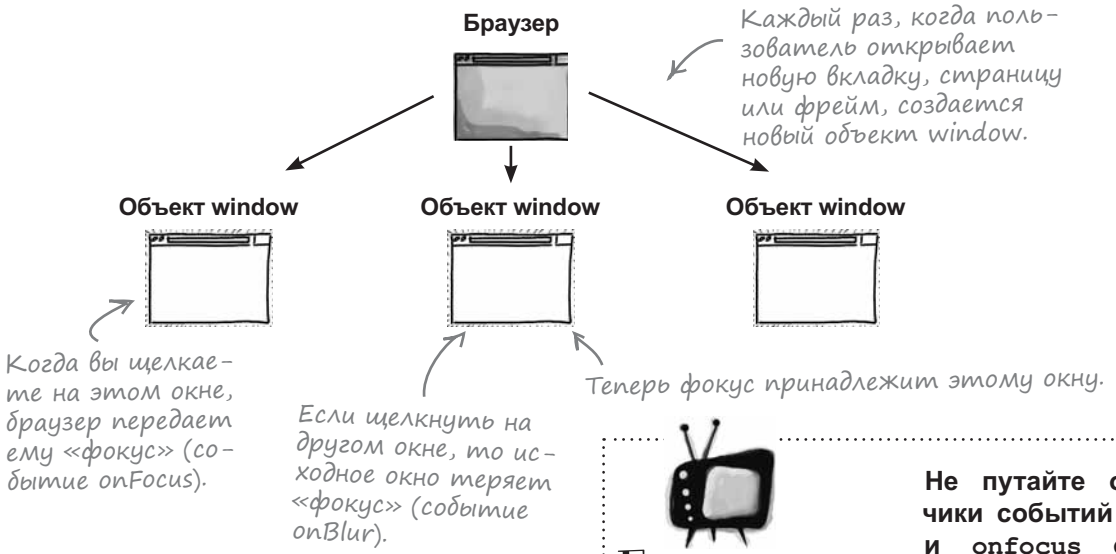
Бесконечные циклы *могут поглощать ресурсы системы, а это вызовет сбой в браузере посетителя.*

Управление временными эффектами

К счастью, анимацией сверкания молний можно управлять при помощи объекта JavaScript's `window`. Объект `window` создается каждый раз, когда посетитель открывает в браузере новое окно; на базе этого объекта реализованы многие полезные функции jQuery и JavaScript. В мире JavaScript объект `window` является *глобальным*. Иначе говоря, он находится на *верхнем уровне* иерархии JavaScript.



Допустим, вы открыли в браузере три вкладки. Браузер создает для каждой вкладки отдельный объект `window`. Это такой же объект, как и те, с которыми мы работали в главе 6; он тоже обладает свойствами, обработчиками событий и методами. И эти объекты чрезвычайно удобны: при помощи обработчиков событий `onblur` и `onfocus` объекта `window` можно узнать, какие операции выполняются пользователем на уровне браузера.



Объект `window` также поддерживает методы работы с таймером, которые мы можем использовать в пользовательских функциях. У объекта `window` много методов, но нас сейчас интересуют только эти методы, необходимые для исправления ошибки с молниями.

Будьте осторожны!

Не путайте обработчики событий `onblur` и `onfocus` объекта JavaScript `window` с методами jQuery `blur` и `focus`.

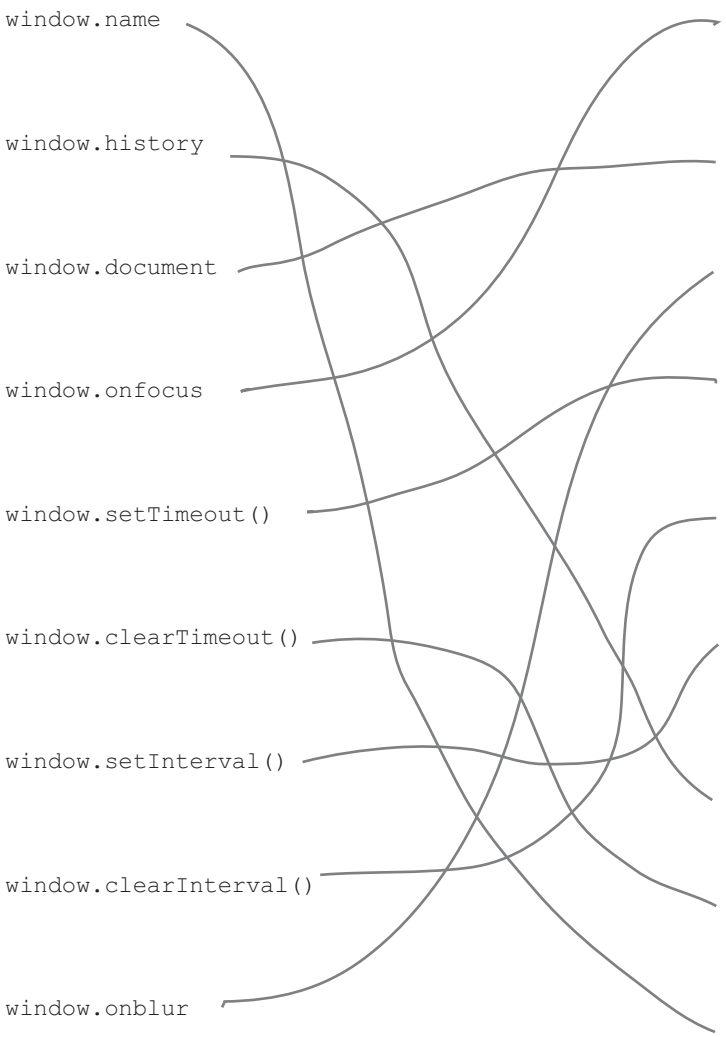
Методы jQuery `blur` *и* `focus` *связываются с полями форм HTML и другими элементами, но не с объектом* `window`.

Кто и что делает?

Соедините каждое свойство, обработчик события или метод окна `window` с его описанием.

`window.name`	Обнаруживает, когда окно получает щелчок, нажатия клавиш или другой ввод.
`window.history`	Свойство объекта `window`, ссылающееся на основное содержимое загруженного документа.
`window.document`	Обнаруживает, когда окно теряет фокус.
`window.onfocus`	Метод объекта `window`, задающий задержку перед выполнением функции или другой команды.
`window.setTimeout()`	Метод объекта `window`, отменяющий задержку между повторными вызовами.
`window.clearTimeout()`	Метод объекта `window`, задающий задержку между повторяющимися выполнениями функции или другой команды.
`window.setInterval()`	Свойство объекта `window` для обращения к URL-адресам, которые ранее загружались в окне.
`window.clearInterval()`	Метод объекта `window`, используемый для отмены задержки перед вызовом.
`window.onblur`	Свойство объекта `window` для чтения или задания имени окна.

Кто и что делает? Решение



Обнаруживает, когда окно получает щелчок, нажатия клавиш или другой ввод.

Свойство объекта `window`, ссылающееся на основное содержимое загруженного документа.

Обнаруживает, когда окно теряет фокус.

Метод объекта `window`, задающий задержку перед выполнением функции или другой команды.

Метод объекта `window`, отменяющий задержку между повторными вызовами.

Метод объекта `window`, задающий задержку между повторяющимися выполнениями функции или другой команды.

Свойство объекта `window` для обращения к URL-адресам, которые ранее загружались в окне.

Метод объекта `window`, используемый для отмены задержки перед вызовом.

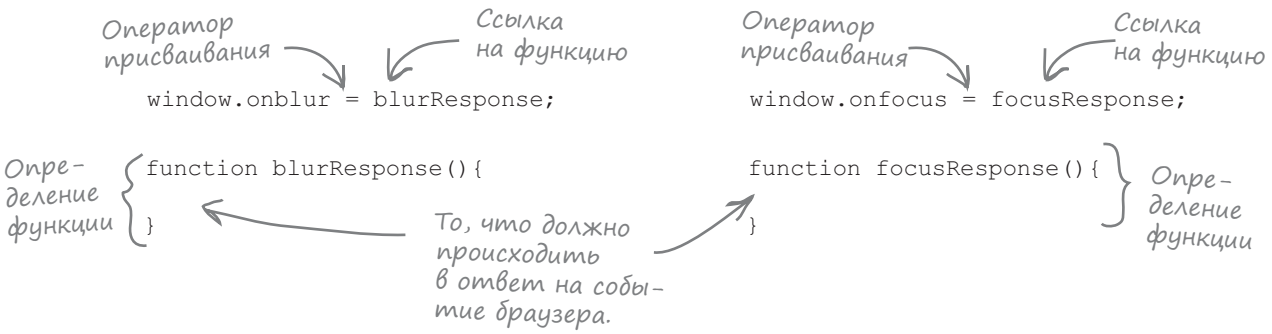
Свойство объекта `window` для чтения или задания имени окна.

Мозговой штурм

Обработчики событий `onfocus` и `onblur` объекта `window` обнаруживают получение/потерю фокуса окном, но как что-то сделать в ответ на эти события?

Обработка событий браузера в onblur и onfocus

Итак, мы знаем, что в `window.onfocus` **можно определить, когда окно получает фокус (то есть посетитель активизирует страницу или передает окну ввод с клавиатуры или мыши), а в** `window.onblur` можно обработать потерю фокуса активным окном. Но что нужно сделать для обработки этих событий? Методам `onfocus` и `onblur` следует присвоить *ссылку на функцию.*



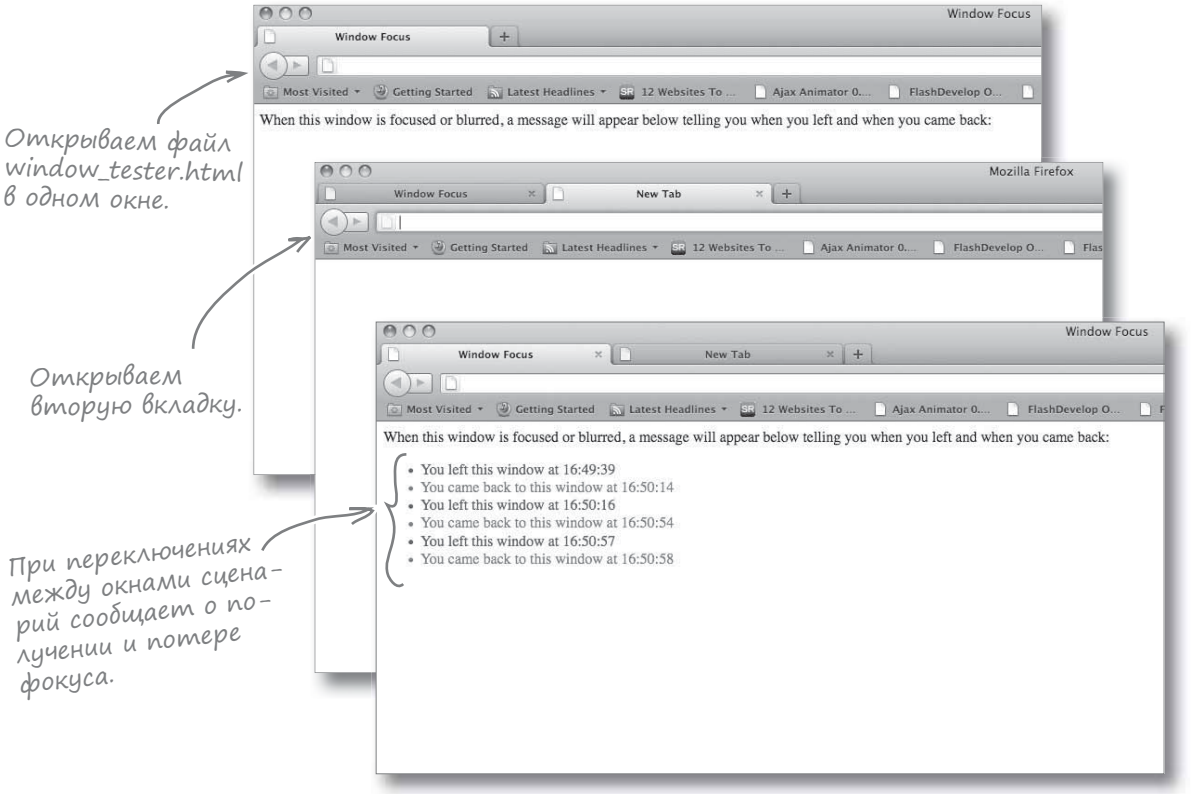
Перед нами наглядный пример того, какую пользу приносит написание пользовательских функций. Имеется объект `window`, содержащий гору информации о том, что пользователь делает в браузере; вы пишете пользовательскую функцию, которая работает с данными, полученными от объекта. Получается, что при обработке события можно делать абсолютно все, *что угодно*, главное написать для этого подходящую пользовательскую функцию...

ТЕСТ-ДРАЙВ

Давайте опробуем обработчики событий `onfocus` и `onblur` объекта `window` в деле. Найдите в файлах главы 7 папку с именем *window_tester*. Загрузите файл *window_tester.html* из этой папки в своем любимом браузере. Откройте вторую вкладку и поэкспериментируйте с переключением между двумя вкладками.

ТЕСТ-ДРАЙВ

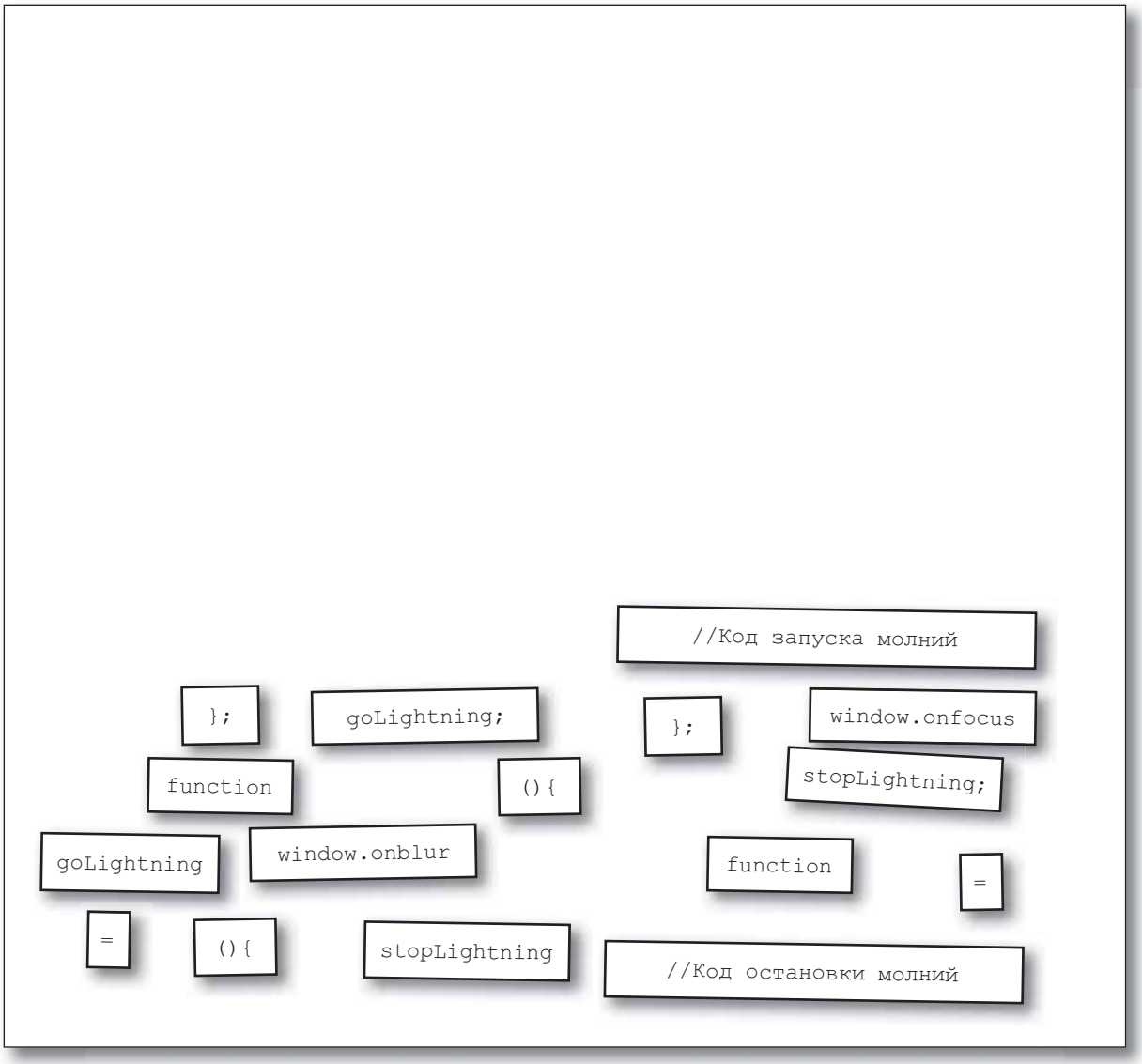
Загрузите файл *window_tester.html*, откройте вторую вкладку и попробуйте переключаться между окнами, попеременно щелкая на них.



При помощи информации, полученной от объекта `window`, можно остановить молнии, когда посетитель покидает окно приложения «Собери монстра», и продолжить отображение эффекта после его возвращения.

Развлечения с магнитами

Сложите из магнитов определения функций для обработчиков `onblur` и `onfocus`. Первая функция останавливает сверкание молнии, когда браузер теряет фокус (она называется `stopLightning`). Другая функция снова запускает эффект при получении фокуса браузером (ей присваивается имя `goLightning`). Код функций пока писать не нужно, просто разложите магниты с комментариями (начинающимися с `//`) внутри каждой функции.



Развлечения с магнитами. Решение

Ниже приведены объявления функций для обоих обработчиков событий объекта `window`.

```
window.onblur = stopLightning;
window.onfocus = goLightning;

function stopLightning (){
    //код остановки молний
};

function goLightning (){
    //Код запуска молний
};
```

Наши функции присваиваются событиям window.onblur и window.onfocus.

При вызове этой функции молнии останавливаются.

А при вызове этой функции они начинают сверкать снова.

Но пока в этих функциях нет ничего, кроме комментариев. Функции должны что-то **делать**! Может, просто скопировать сюда код старых функций?

Верно. Функции ничего не делают... пока.

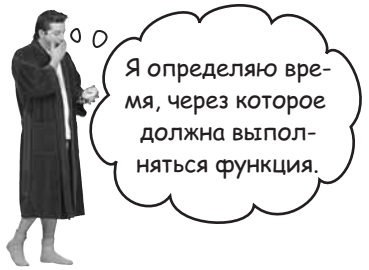
Не торопитесь копировать старый код. Эффект сверкания молнии удобнее реализовать с использованием методов работы с таймером объекта `window`.

Методы работы с таймером определяют время выполнения функций

И в JavaScript, и в jQuery имеются методы для выполнения функций по таймеру. У объекта JavaScript `window` есть четыре метода для работы с таймером: `setTimeout`, `clearTimeout`, `setInterval` и `clearInterval`. jQuery поддерживает метод `delay`. Давайте поближе познакомимся с этими методами и их возможностями.

Методы JavaScript

setTimeout




```
setTimeout(myFunction, 4000);
```

Функция, вызываемая по истечении тайм-аута

Задержка таймера (в миллисекундах)

setInterval


```
setInterval(repeatMe, 1000);
```

Функция, вызываемая после истечения каждого интервала

Интервал между вызовами функции (в миллисекундах)

Метод delay jQuery

delay


```
slideDown().delay(5000).slideUp();
```

Выполнение таких цепочек в jQuery называется «очередью эффектов».

В данном примере метод delay вставляет 5-секундную задержку между эффектами slideUp и slideDown..

Возьми в руку карандаш

Какой из этих методов подойдет для исправления функции `goLightning`? Напишите для каждого метода, пригодится ли он в нашей ситуации, и объясните, почему вы его выбрали (или не выбрали).

Метод	Будем использовать?	Почему?
`setTimeout`		
`setInterval`		
`delay`		

Возьми в руку карандаш
Решение

Какой из этих методов подойдет для исправления функции `goLightning`? Вот наши ответы.

Метод	Будем использовать	Почему?
`setTimeout`	*Нет*	*Метод setTimeout предназначен для создания задержки перед однократным выполнением функции.*
`setInterval`	*Да*	*Метод setInterval предназначен для многократного выполнения функции с заданными интервалами. Именно это нам и нужно в эффекте с молниями.*
`delay`	*Нет*	*Метод delay хорошо подходит для создания серий эффектов, но у него нет средств планирования отложенного запуска.*

Итак, **setInterval** лучше всего подойдет для функции **goLightning**, но функция **stopLightning** должна остановить таймер. Как бы это сделать? Может методом **clearInterval**?

Хороший вопрос!

Метод `clearInterval` останавливает вызов функции по таймеру, запущенный методом `setInterval`. При вызове `clearInterval` необходимо передать параметр. Вот как это делается:

Присваивается переменной, идентифицирующей метод setInterval.

```
myInterval = setInterval(repeatMe, 1000);

clearInterval(myInterval);
```

Переменная передается в параметре clearInterval.

Метод clearInterval останавливает таймер setInterval и прекращает периодическое выполнение.

Часто Задаваемые Вопросы

В: Метод `setTimeout` во всех браузерах работает одинаково?

О: Нет. Mozilla Firefox и Google Chrome работают так, как было описано выше («складывая» вызовы функций). Internet Explorer 9 продолжает вызывать функцию так, как предполагалось в главе 5. Это показывает, что проблемы с межбраузерной совместимостью возникают не только у веб-дизайнеров.

В: Можно ли использовать функции `setInterval` и `setTimeout` для чего-то другого, кроме объекта `window`?

О: К сожалению, нельзя. Это конкретные методы объекта `window`, и вызываться они могут только по ссылке на объект `window`. Однако возможен вызов методов без префикса «window»; браузер определит, что вызов нужно связать с текущим объектом `window`. Впрочем, префикс все же рекомендуется указывать.

Кто и что делает?

Соедините каждый вызов метода с его описанием.

```
window.clearInterval(int1);
```

Обнаруживает, когда текущее окно получает фокус, и вызывает метод `goLightning`.

```
window.onfocus = goLightning;
```

Выполняет функцию `lightning_one` каждые 4 секунды, с присваиванием результата переменной `int1`.

```
setTimeout(wakeUp(),4000);
```

Обнаруживает, когда текущее окно теряет фокус, и вызывает функцию `stopLightning`.

```
$("#container #lightning1").
fadeIn(250).delay(5000).fadeOut(250).;
```

Останавливает таймер и прекращает повторение для `int1`.

```
int1 = setInterval( function() {
          lightning_one();
        },
        4000
      );
```

Устанавливает задержку 4 секунды перед вызовом функции `wakeUp`.

```
window.onblur = stopLightning;
```

Создает пятисекундную паузу между эффектами `fadeIn` и `fadeOut`.

РЕШЕНИЕ

Соедините каждый вызов метода с его описанием.

```
window.clearInterval(int1);

window.onfocus = goLightning;

setTimeout(wakeUp(),4000);

$("#container #lightning1").
fadeIn(250).delay(5000).fadeOut(250).;

int1 = setInterval( function() {
          lightning_one();
        },
        4000
      );

window.onblur = stopLightning;
```

- Обнаруживает, когда текущее окно получает фокус, и вызывает метод `goLightning`.
- Выполняет функцию `lightning_one` каждые 4 секунды, с присваиванием результата переменной `int1`.
- Обнаруживает, когда текущее окно теряет фокус, и вызывает функцию `stopLightning`.
- Останавливает таймер и прекращает повторение для `int1`.
- Устанавливает задержку 4 секунды перед вызовом функции `wakeUp`.
- Создает пятисекундную паузу между эффектами `fadeIn` и `fadeOut`.

Пишем функции stopLightning и goLightning

Теперь вы больше знаете о таймере, давайте посмотрим, как использовать эти методы в нашем коде.

```
goLightning();
```
Молнии запускаются при загрузке страницы.

```
window.onblur = stopLightning;
```
Функция stopLightning вызывается при потере фокуса браузером.

```
window.onfocus = goLightning;
```
Функция goLightning вызывается при получении фокуса браузером.

```
function stopLightning (){
  //Код остановки молний
};
```
Сброс таймеров для трех интервалов. Нам понадобятся три вызова clearInterval. Знаете, почему?

```
function goLightning (){
  //Код запуска молний
};
```
Установка трех таймеров для трех интервалов. Да, здесь нам понадобятся три вызова setInterval.

Упражнение

Пора исправить недостатки приложения «Собери монстра». Заполните пропуски именами переменной, функции или метода. Если вы сомневаетесь, снова просмотрите код на двух предыдущих страницах. Мы заполнили некоторые пропуски за вас.

```
goLightning();
window.onblur = stopLightning;
window.onfocus = goLightning;
var int1, int2, int3 ;
function goLightning(){
   int1= ..............( function() {

       }, ...................
       4000
     );

   ........= ..............( function() {

       }, ...................
       5000
     );
   ........= ..............( function() {
         lightning_three();
       },
       7000
     );
}

function stopLightning()
{
    window. ..................(int1 );
    window. ..................(........);
    window. ..................(........);
}
function lightning_one(){
  $("#container #lightning1").fadeIn(250).fadeOut(250);
};

function ..................  {
  $("#container #lightning2").fadeIn(250).fadeOut(250);
};

function ..................  {
  $("#container #lightning3").fadeIn(250).fadeOut(250);
};
```

my_scripts.js

Упражнение
Решение

У нас появились две пользовательские функции, реагирующие на события `onfocus` и `onblur` объекта `window` (в них используются функции, написанные нами в главе 5).

```
goLightning();
window.onblur = stopLightning;
window.onfocus = goLightning;
var int1, int2, int3 ;
function goLightning(){
  int1 = setInterval( function() {
        lightning_one();
      },
      4000
    );
  int2 = setInterval ( function() {
        lightning_two();
      },
      5000
    );
  int3 = setInterval ( function() {
        lightning_three();
      },
      7000
    );
}

function stopLightning()
{
   window.clearInterval( int1 );
   window.clearInterval( int2 );
   window.clearInterval( int3 );
}
function lightning_one(){
  $("#container #lightning1").fadeIn(250).fadeOut(250);
};

function lightning_two(){
  $("#container #lightning2").fadeIn(250).fadeOut(250);
};

function lightning_three() {
  $("#container #lightning3").fadeIn(250).fadeOut(250);
};
```

Объявляем три переменные для хранения таймеров, чтобы при необходимости их можно было снова сбросить в браузере.

Здесь вызывается функция lightning_one.

Устанавливаем три разных таймера с тремя интервалами.

Затем вызываем функцию lightning_two.

И последней вызывается функция lightning_three.

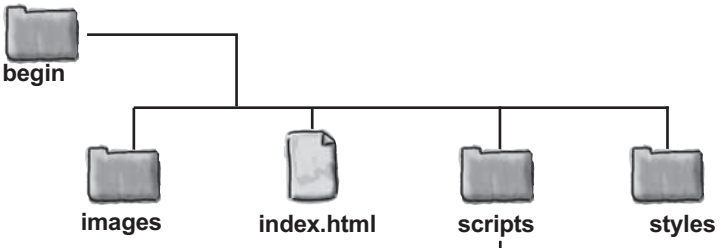
Сброс таймеров для трех интервалов

Определения трех функций.

my_scripts.js

Задание!

Вносить все изменения и исправления в код, написанный для главы 5, было бы слишком сложно; проще начать заново, с пустого файла сценария. Найдите в архиве кода, загруженного для книги, папку главы 7. В ней находится вложенная папка *begin*, которая имеет следующую структуру:



Включите код с предыдущей страницы...

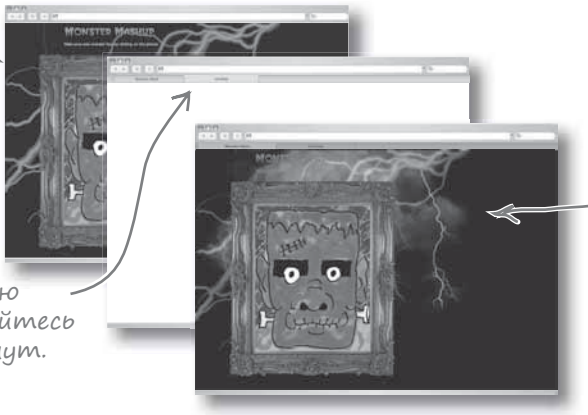
...в блок $(document).ready своего файла сценария.

```
$(document).ready(function(){

});//end doc.onready function
```

ТЕСТ-ДРАЙВ

Включите код с предыдущей страницы в файл сценария, откройте страницу в браузере и посмотрите, удалось ли нам решить проблему с эффектами молний.

Откройте приложение «Собери монстра» в окне браузера.



Откройте новую вкладку, оставайтесь на ней пару минут.

Затем вернитесь на исходную вкладку с приложением «Собери монстра». При возвращении первый эффект молнии должен запуститься только через четыре секунды.

Раз уж мы занялись исправлениями, почему бы не избавиться от однообразных функций, написанных в главе 5?

Отличная идея. У нас несколько почти одинаковых функций для обработки щелчков, которые можно заменить ***одной*** универсальной функцией.

Здесь лучше подойдет другая структура данных, которая содержит несколько переменных.

Нельзя ли написать одну функцию, которую можно будет использовать в каждом из этих случаев?

```
var headclix = 0, eyeclix = 0, noseclix = 0, mouthclix = 0;

        $("#head").click(function(){
            if (headclix < 9){
                  $("#head").animate({left:"-=367px"},500);
                  headclix+=1;
            }
            else{
                  $("#head").animate({left:"0px'},500);
                  headclix = 0;
            }
        });

      $("#eyes").click(function(){
            if (eyeclix < 9){
                  $("#eyes").animate({left:"-=367px"},500);
                  eyeclix+=1;
            }
            else{
                  $("#eyes").animate({left:"0px"},500);
                  eyeclix = 0;
            }
        });

      $("#nose").click(function(){
            if (noseclix < 9){
                  $("#nose").animate({left:"-=367px"},500);
                  noseclix+=1;
            }
            else{
                  $("#nose").animate({left:"0px"},500);
                  noseclix = 0;
            }
        });//end click

       $("#mouth").click(function(){
            if (mouthclix < 9){
                  $("#mouth").animate({left:"-=367px"},500);
                  mouthclix+=1;
            }
            else{
                  $("#mouth").animate({left:"0px"},500);
                  mouthclix = 0;
            }
        });//end click
```

my_scripts.js

Развлечения с магнитами

В коде на предыдущей странице найдите фрагменты, общие для разных аспектов приложения. Сложите из магнитов код универсальной функции `moveMe`, которая вызывается при щелчке на любой из подвижных частей изображения. В первом параметре `moveMe` передается индекс массива `clix`, а во втором — ссылка на часть изображения, на которой был сделан щелчок.

```
var clix = __________; // head,eyes,nose,mouth

      $("#head").click( function(){

            _________________
      });//end click function

      $("#eyes").click( function(){

            _________________
      } );//end click function

      $("#nose").click( function(){

            _________________
      });//end click function

      $("#mouth").click( function(){

            _________________
      });//end click function

      function moveMe(________){

            if (__________ < 9){
                  $(obj).animate({left:"-=367px"},500);
                  clix[i] = clix[i]+1;
            }else{
                  clix[i] = 0;
                  $(______).animate({left:"0px"},500);
            }
      }
```

moveMe(2, this);

obj

moveMe(0, this);

i, obj

moveMe(3, this);

clix[i]

[0,0,0,0]

moveMe(1, this);

my_scripts.js

Развлечения с магнитами. Решение

Итак, теперь у нас имеется одна универсальная функция, работающая с массивом. Одна функция создает меньше проблем с сопровождением кода, в ней проще находить и исправлять ошибки.

```
var clix = [0,0,0,0] ; // head,eyes,nose,mouth

      $("#head").click( function(){

            moveMe(0, this);
      });//end click function

      $("#eyes").click( function(){

            moveMe(1, this);
      } );//end click function

      $("#nose").click( function(){

            moveMe(2, this);
      });//end click function

      $("#mouth").click( function(){

            moveMe(3, this);
      });//end click function

      function moveMe( i, obj ){

            if ( clix[i] < 9){
                  $(obj).animate({left:"-=367px"},500);
                  clix[i] = clix[i]+1;
            }else{
                  clix[i] = 0;
                  $( obj ).animate({left:"0px"},500);
            }
      }
```

Преобразование набора переменных в массив делает код более компактным.

Функция moveMe получает ссылку на ячейку массива clix. Эта информация может использоваться для отслеживания количества щелчков на каждом элементе.

Функции moveMe также передается ссылка на текущий объект для выполнения анимации.

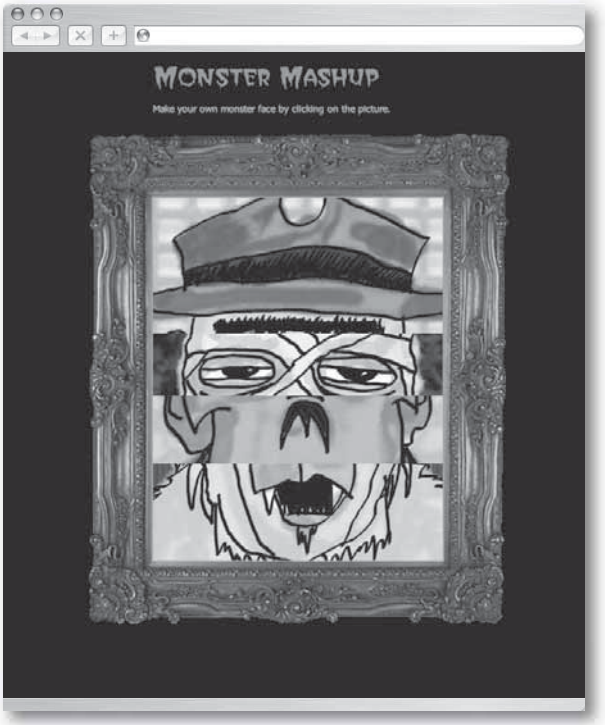
Универсальная функция moveMe снижает риск ошибок программирования и сокращает количество функций, которые вам придется сопровождать.

Дублирующаяся логика теперь сосредоточена в одном месте. Если в программе обнаружится ошибка, исправить ее будет проще и быстрее, чем в нескольких дубликатах.

my_scripts.js

Тест-драйв

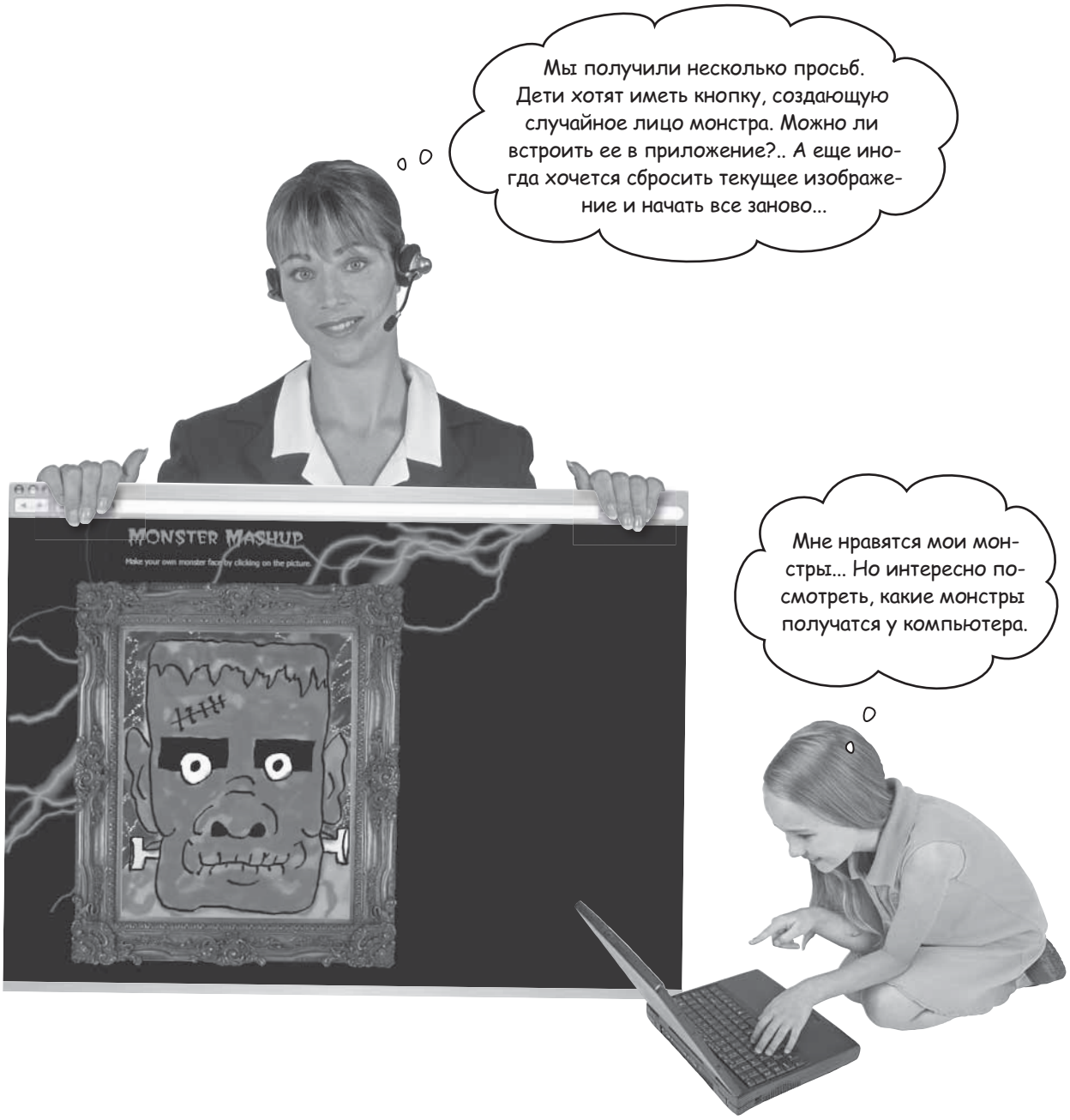
Включите код из упражнения с магнитами на предыдущей странице в файл *my_scripts.js*. Сохраните файл, откройте страницу *index.html* в браузере и убедитесь в том, что новая версия функции правильно обрабатывает щелчки на разных частях лица монстра.



Внешне страница приложения нисколько не изменилась, но мы-то знаем, что код стал более эффективным, содержит меньше дубликатов и создает меньше проблем с сопровождением.

Новая просьба

Джилл и вся группа контроля качества довольны вашими исправлениями. А раз уж им нравится ваша работа, они передают вам просьбу на расширение функциональности приложения от потенциальных клиентов.



Случайные монстры

Мы уже не раз создавали функции, использующие генератор случайных чисел, так что сейчас вы уже стали настоящим специалистом по ним. Сейчас нам предстоит создать функцию для выполнения случайной анимации лица монстра. Давайте разделим общую задачу на серию меньших подзадач. Начнем с определения текущей позиции в каждой полосе с изображениями.

Нам потребуется отслеживать текущую позицию для каждой полосы изображений. Допустим, посетитель сейчас находится здесь.

*Текущая позиция вычисляется как количество щелчков, умноженное на расстояние между частями лица на полосе (367 пикселов). В нашем примере текущая позиция равна 2 * 367, то есть 734.*

От текущей позиции нужно перейти к целевой позиции, которая по сути является случайной позицией в полосе изображений. Операцию перемещения удобнее разбить на две части:

1 **Получение случайного числа.**

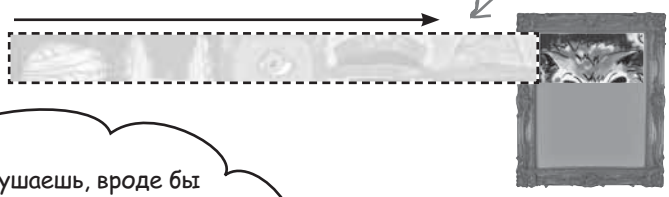
Так мы получали случайные числа в главах 2 и 3.

```
var my_num = Math.floor((Math.random()*5) + 5);
```

Но на этот раз потребуется число от 1 до 10 (потому что каждая полоса состоит из 10 фрагментов).

2 **Для каждой части лица выбирается случайная позиция.**

*Каждая часть лица монстра оказывается в случайной позиции, умноженной на ширину каждого фрагмента. Для случайного числа 7 целевая позиция равна 7 * 367, то есть 2569.*



Хм, вас послушаешь, вроде бы все просто... Но что мы будем делать с определением текущей позиции? Как узнать, какая часть лица отображается на полосе, особенно если ее кто-то уже прокручивал?

Все проще, чем кажется на первый взгляд.

Чтобы узнать, как это делается, просто переверните страницу.

Мы уже знаем текущую позицию...

К счастью, нам не потребуются ни новые переменные, ни новые функции. Текущая позиция определяется значением в соответствующей ячейке массива `clix`, в которой хранится информация, сколько раз пользователь щелкнул на каждой части лица монстра. Нам понадобится одна строка кода:

Переменной текущей позиции присваивается значение clix[index].

```
var current_position = clix[index] ;
```

...и функция getRandom уже готова

Мы уже создавали функции для получения случайных чисел в главах 2, 3 и 6. Сейчас мы сможем использовать готовую функцию с минимальными изменениями.

```
function getRandom(num){
      var my_random_num = Math.floor(Math.random()*num);
      return my_random_num;
}
```

Функция getRandom получает в аргументе число...

...генерирует и возвращает целое число. В данном случае генерируется число от 0 до 10.

Умножая Math.random на число, переданное в аргументе, мы генерируем случайное число в интервале от 0 до num.

1. **Присваиваем значение переменной и передаем ее функции:**

Передаем значение функции

```
num = 10;
getRandom(num);
```

2. **Выполняем основную операцию функции:**

Функции, специализирующиеся на решении одной задачи, иногда называют **вспомогательными**.

```
var my_random_num = Math.floor(Math.random()*num);
```

3. **Возвращаем результат:**

```
return my_random_num;
```

Вызов дает значение целевой позиции (т. е. случайный фрагмент лица), к которой мы хотим перейти.

Готово к употреблению

Включите код, выделенный жирным шрифтом, в файлы *index.html* и *my_scripts.js*. В нем определяется функция, генерирующая случайное изображение монстра, а также выводятся сообщения с информацией о целевой позиции (выбранной на основании случайного числа) и текущей позиции (определяемой количеством щелчков, сделанных посетителем).

```
<header id="top">
    <img src="images/Monster_Mashup.png" />
    <button id="btnRandom">Randomize</button>
    <button id="btnReset">Reset</button>
    <p>Make your own monster face by clicking on the picture.</p>
</header>
```

В интерфейс нужно включить кнопки для генерирования случайного монстра и сброса.

index.html

```
var w = 367; // Ширина одного фрагмента
var m = 10; // Количество фрагментов

$("#btnRandom").click( randomize );
$("#btnReset").click(  );

function getRandom(num){
      var my_random_num = Math.floor(Math.random()*num);
      return my_random_num;
}
function randomize(){
      $(".face").each(function(index){
            var target_position = getRandom(m);
            var current_position = clix[index] ;
            clix[index] = target_position;
            var move_to = target_position * w;
            $(this).animate({left:"-="+move_to+"px"},500);
      });
};
```

Определение случайной позиции для каждой части лица

Переменной target_position присваивается результат вызова getRandom.

Обновляем clix[index], чтобы пользователь мог продолжать щелкать на частях лица монстра.

Переменной move_to присваивается случайная позиция, умноженная на ширину одного фрагмента на полосе.

Выполняем пользовательский код анимации перемещения полосы влево.

my_scripts.js

ТЕСТ-ДРАЙВ

Включите код с предыдущей страницы в ваши файлы, откройте страницу *index.html* в своем любимом браузере для тестирования функции `randomize`. Щелкните на кнопке Randomize 10–20 раз, чтобы тщательно протестировать новую функцию.

Генератор монстров работает...

На нескольких первых щелчках генератор монстров делает именно то, что требуется.

...но только на нескольких первых щелчках

Но через несколько щелчков начинаются какие-то проблемы.

Части лица пропадают? Такого мы не программировали. Они должны были попадать в случайную позицию! Может, мы оказались слишком далеко?

Вы правы.

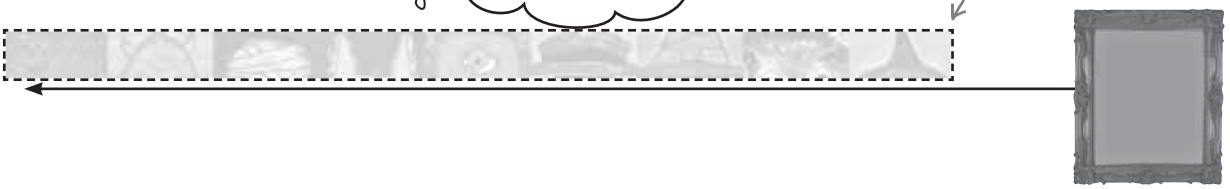
У наших функций имеются непредвиденные побочные эффекты. Но скорее всего, эти функции делают в точности то, что мы написали в своем коде. Давайте посмотрим, что мы могли упустить.

Повторные вызовы animate продолжают смещать полосу изображений влево и вскоре выводят ее за пределы видимости. Изображение, что называется, «уходит за край».

```
$(this).animate({left:-="+move_to+"px"},500);
```

Если пользователь продолжает щелкать на кнопке Randomize, полоса уйдет налево так далеко, что изображения перестанут отображаться на экране.

Помогите! Мы хотим обратно в рамку!

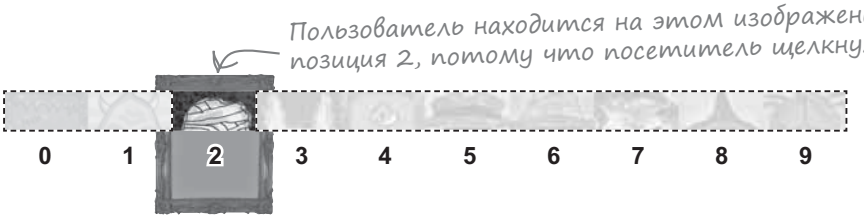


НАПРЯГИ МОЗГИ

Как вы думаете, что нужно сделать, чтобы полоса изображений не уходила за край, а оставалась на случайных фрагментах монстра?

Перемещение относительно текущей позиции

Чтобы полоса изображений не выходила за край, а останавливалась на случайной части лица монстра (как было задумано), необходимо перемещать ее *относительно текущей позиции*. Для этого нужно включить в сценарий проверку текущей позиции и условную логику. Давайте разберемся подробнее.



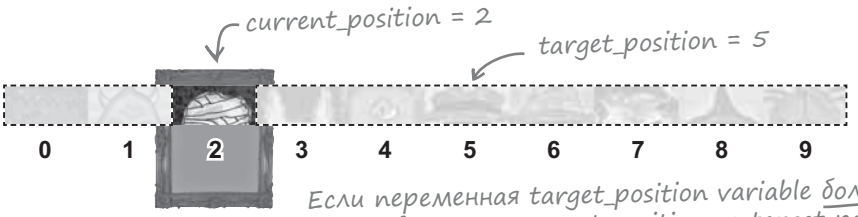
Далее пользователь щелкает на кнопке Randomize, которая генерирует случайное число в интервале от 0 до 9. Рассмотрим две принципиально возможные ситуации.

Ситуация 1: target_position > current_position

Функция `getRandom` вернула значение 5. Таким образом, переменной `target_position` присваивается значение 5, то есть она *больше* переменной `current_position`. Мы должны написать логику для обработки этой ситуации.

На сколько позиций нужно сместить полосу?

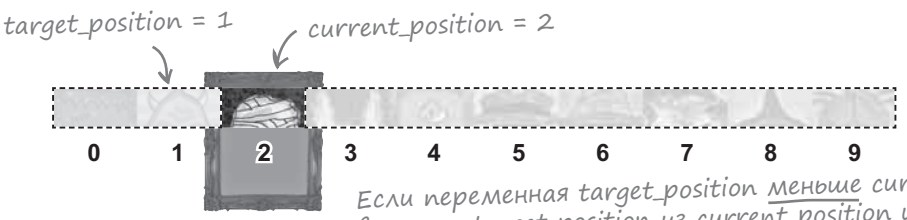
Вычитая current_position из target_position, получаем 3. Нужно переместиться на три позиции влево.



Если переменная target_position variable *больше* current_position, нужно *вычесть* current_position из target_position и сместить полосу изображений влево вызовом animate({left:"-=".

Ситуация 2: target_position < current_position

Функция `getRandom` вернула значение 1. Переменная `target_position` равна 1, то есть она меньше переменной `current_position`. Как вы думаете, какую логику следует применить в этом случае (по образцу условной логики из ситуации 1)?



Вычитая target_position из current_position, получаем 1. Необходимо сместиться на одну позицию вправо.

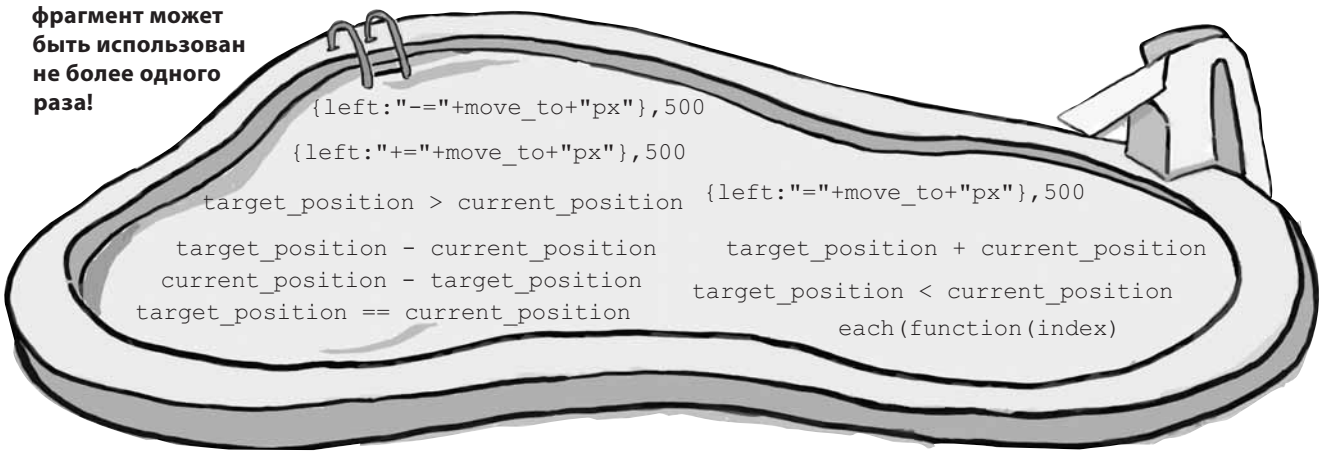
Если переменная target_position *меньше* current_position, нужно *вычесть* target_position из current_position и сместить полосу изображений вправо вызовом animate({left:"+=".

Ребус в бассейне

Выловите из бассейна фрагменты кода и расставьте их на пустых местах в коде. Каждый фрагмент может использоваться **только один** раз; некоторые фрагменты могут остаться неиспользованными. Ваша **задача** — исправить ошибку в функции, чтобы части лица монстра не оставались пустыми.

```
var w = 367;
var m = 10;

function getRandom(num){
       var my_random_num = Math.floor(Math.random()*num);
       return my_random_num;
}
function randomize(){
       $(".face").....{
              var target_position = getRandom(m);
                     var current_position = clix[index] ;
                     clix[index] = target_position;

                     if(.....) {
                            var move_to = (.....) * w;
                            $(this).animate(.....);
                     }else if(.....){
                            var move_to = (.....) * w;
                            $(this).animate(.....);
                     }else{
                            // Позиция не изменилась - ничего не делать.
                     }
              });
};
```

Внимание: каждый фрагмент может быть использован не более одного раза!



Решение ребуса в бассейне

Выловите из бассейна фрагменты кода и расставьте их на пустых местах в коде. Каждый фрагмент может использоваться **только один** раз; некоторые фрагменты могут остаться неиспользованными. Ваша **задача** — исправить ошибку в функции, чтобы части лица монстра не оставались пустыми.

```
var w = 367;
var m = 10;

function getRandom(num){
      var my_random_num = Math.floor(Math.random()*num);
      return my_random_num;
}
function randomize(){
      $(".face").each(function(index) {
            var target_position = getRandom(m);
            var current_position = clix[index] ;
            clix[index] = target_position;

            if( target_position > current_position ) {
                  var move_to = ( target_position - current_position ) * w;
                  $(this).animate({left:"-="+move_to+"px"},500);
            }else if( target_position < current_position ){
                  var move_to = ( current_position - target_position ) * w;
                  $(this).animate( {left:"+="+move_to+"px"},500 );
            }else{
                  // Позиция не изменилась - ничего не делать.
            }
      });
};
```

Выполнить следующий код для каждого элемента класса face.

Если переменная target_position больше current_position...

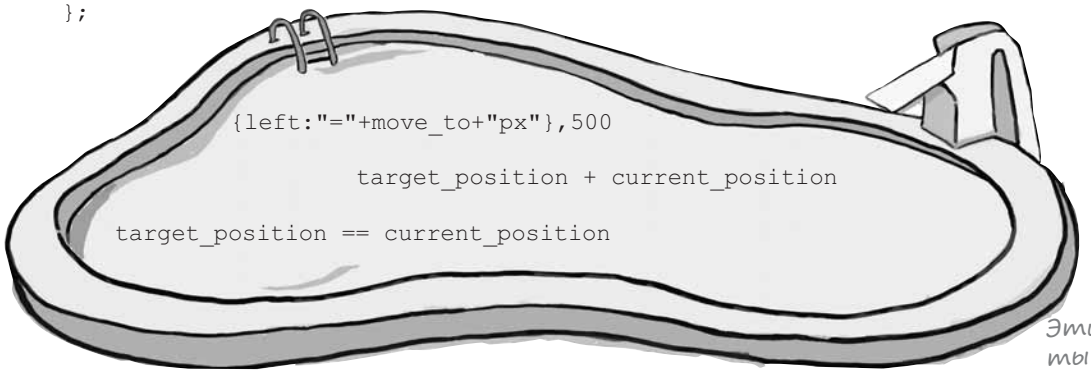
...вычесть target_position из current_position.

Полоса изображений перемещается влево. Для этого используется вызов animate({left:"-=".

Если переменная target_position меньше current_position...

...вычесть target_position из current_position.

Полоса изображений перемещается вправо. Для этого используется вызов animate({left:"+="...



Эти фрагменты оказались лишними.

Функция генерирования случайных монстров работает отлично. Пора переходить к функции сброса изображения в исходное состояние?

Совершенно верно.

Помните кнопку Reset в файле *index.html* несколько страниц назад? Осталось связать ее с пользовательской функцией сброса.

Развлечения с магнитами

Разложите магниты по местам так, чтобы у вас получился код кнопки и пользовательской функции сброса. Мы уже расставили часть магнитов за вас.

```
$("#btnReset").click( reset );

function reset(){

}
```

```
$(".face")
});
clix[index] = 0;
$(this)
.each(function(index){
.animate({left:"0px"},500);
```

Развлечения с магнитами. Решение

Вуаля! Всего несколько строк — кнопка сброса работает, и монстра можно собирать заново.

```
$("#btnReset").click( reset );
```
Связываем пользовательскую функцию сброса с кнопкой.

```
function reset(){
```
Определяем пользовательскую функцию сброса.

```
$(".face")
    .each(function(index){
```
Используем each, чтобы функция сбрасывала текущую позицию каждой части до 0.

```
clix[index] = 0;
```
Обнуляем массив clix.

```
$(this)
    .animate({left:"0px"},500);
```
Каждая полоса возвращается к началу, для чего ее свойству CSS left присваивается абсолютная позиция 0px.

```
});
}
```

Часто Задаваемые Вопросы

В: Объект `window` существует во всех браузерах?

О: Да, все современные браузеры поддерживают объект `window`, с которым вы можете работать. Каждый объект `window` (по одному на каждую вкладку браузера) также имеет отдельный объект `document`, в который загружается веб-страница.

В: Почему перемещения выполняются относительно текущей позиции? Почему я не могу перейти туда, куда указывает случайное число?

О: Такое решение будет работать, но вам придется возвращать изображение в начальную позицию, а потом перемещать его в позицию, определяемую генератором случайных чисел. Это удваивает объем кода, который вам придется написать, и замедляет работу приложения.

В: Как работает функция `reset`?

О: Функция `reset` просто перебирает все элементы с классом `face` и присваивает их свойству CSS `left` значение 0. Затем она обнуляет все ячейки массива `clix`, как на момент загрузки страницы.

Ниже приведен весь код, написанный на предыдущих страницах. Если вы еще не сделали этого ранее, добавьте код, выделенный жирным шрифтом, в файл *my_scripts.js* и приготовьтесь к тестированию всех новых возможностей приложения.

```
var w = 367; // Ширина одного фрагмента
var m = 10;  // Количество фрагментов
$("#btnRandom").click( randomize );
$("#btnReset").click( reset );

function getRandom(num){
      var my_random_num = Math.floor(Math.random()*num);
      return my_random_num;
}
function randomize(){
      $(".face").each(function(index){
            var target_position = getRandom(m);
            var current_position = clix[index] ;
            clix[index] = target_position;
                  if( target_position > current_position ) {
                  var move_to = (target_position - current_position) * w;
                        $(this).animate({left:"-="+move_to+"px"},500);
                  }else if( target_position < current_position ){
                        var move_to = (current_position - target_position) * w;
                        $(this).animate({left:"+="+move_to+"px"},500);
                  }else{
                        // Позиция не изменилась - ничего не делать.
                  }
      });
}

function reset(){
      $(".face").each(function(index){
            clix[index] = 0;
            $(this).animate({left:"0px"},500);
      });
}
```

my_scripts.js

ТЕСТ-ДРАЙВ

Введите код с предыдущей страницы, откройте файл *index.html* в своем любимом браузере для тестирования функций `randomize` и `reset`. Щелкните на кнопке Randomize 10–20 раз, чтобы тщательно протестировать новую функцию. Время от времени щелкайте на кнопке Reset, она тоже должна работать так, как задумано.

Все работает!

Части лица монстра теперь перемещаются влево-вправо, отчего приложение смотрится еще лучше.

А кнопка Reset возвращает все в исходное состояние.

«Собери монстра-2» — настоящий хит!

ГЛАВА 7

Ваш инструментарий jQuery

Глава 7 осталась позади. Ваш творческий инструментарий расширился: появился объект `window`, функции работы с таймером и пользовательские функции.

Объект window

Объект верхнего уровня в иерархии JavaScript.

Обладает свойствами, обработчиками событий и методами, которые позволяют обнаруживать события браузера и реагировать на них.

Событие onFocus сообщает об активизации окна браузера.

Событие onBlur сообщает о потере фокуса окном.

Функции работы с таймером

Методы объекта window.

setTimeout обеспечивает вызов функции с заданной задержкой.

setInterval многократно выполняет функцию с заданными промежутками.

clearInterval отменяет запланированные периодические вызовы функции.

Оптимизированные пользовательские функции

Пользовательские функции позволяют создавать интерактивные веб-страницы, соответствующие современным стандартам.

Старайтесь по возможности объединять и оптимизировать ваши функции, чтобы уменьшить объем программного кода. Чем компактнее код, тем проще его сопровождать и отлаживать.

Пожалуйста, передайте данные

Использовать jQuery для всяких фокусов с CSS и DOM довольно весело, но при программировании веб-приложений необходимо получать данные с сервера и отображать их на странице. Возможно, вам даже захочется обновлять небольшие фрагменты страницы без полной перезагрузки страницы. Технология Ajax в сочетании с jQuery и JavaScript позволяет решить эту задачу. В этой главе вы узнаете, как в jQuery реализуются обращения Ajax к серверу и что можно сделать с полученной информацией.

Бегом к современным технологиям

От: **Отдел маркетинга MegaCorps**
Тема: **42-й Ежегодный марафон Интернетвиля — страница результатов**

Привет группе веб-разработчиков,

Наша фирма публикует веб-страницу с результатами марафона на кубок Интернетвиля. Но наша страница *сильно* отстает от реальности, мы обновляем ее только после получения всех результатов. Пользователи желают моментального обновления информации, как в Twitter и Facebook, и хотят сразу видеть результаты.
Предлагаем вам работу за хорошее вознаграждение. А если сможете обновить нашу страницу результатов марафона к следующей неделе, можете рассчитывать на места в VIP-ложе в финале. (Мы упоминали, что в этом году соревнования проходят на Мауи?)

Вот что нам нужно:

1) должна существовать возможность вывода результатов соревнований для мужчин, для женщин или для всех участников сразу;

2) информация на странице должна обновляться по мере того, как участники пересекают финишную прямую;

3) обновление результатов должно происходить без перезагрузки страницы;

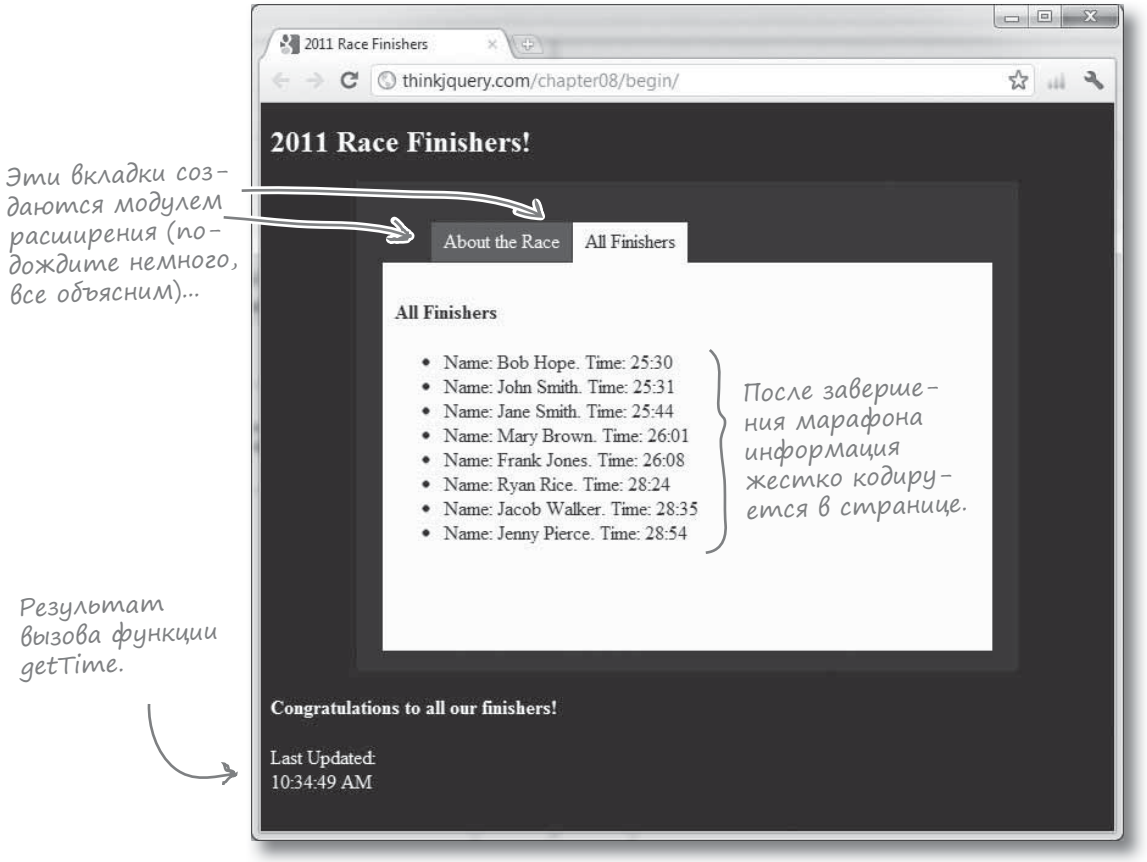
4) на странице должно быть указано время последнего обновления и частота обновлений. Пользователи имеют право запускать и останавливать обновления по своему усмотрению.

Внешне страница изменится не сильно, поэтому работу можно начать с прошлогодней версии. Гонка для нас очень важна, и мы с нетерпением ждем, что у вас получится!

--
Диона Хаузни, руководитель отдела маркетинга
Фирма MegaCorp

Прошлогодняя страница

Давайте поближе познакомимся с прошлогодней страницей. Посмотрим, как она выглядела и как была устроена ее разметка. Все это поможет нам лучше понять, чего же от нас хочет заказчик.



Настройка модуля расширения

Модули расширения совершенствуют функциональность базовой библиотеки jQuery или упрощают выполнение некоторых операций. В приведенном примере модуль `idTabs` в сочетании с CSS преобразует элемент `ul` во вкладки, активизируемые щелчками, и сообщает ссылкам `a` в элементах `li`, какие элементы `div` должны отображаться. Этот модуль расширения предоставляет очень удобную навигационную структуру для страниц, обеспечивая визуальное разделение разных видов информации при использовании общей области вывода.

Пока не обращайте внимания на модули расширения.

Они предоставляют дополнительную функциональность для стандартной библиотеки jQuery. Мы остановимся на этой теме в главе 10, а пока просто посмотрим, что может сделать этот модуль для ускорения работы над проектом...

Готово к употреблению

Прежде чем двигаться дальше, мы также просмотрим прошлогодние файлы. Код и разметка находятся в файле *last_year.zip* (вместе с другими файлами этой главы, которые можно загрузить по адресу *http://thinkjquery.com/chapter08*). Ниже приведены фрагменты трех основных файлов, которые нам понадобятся: *my_style.css*, *index.html* и *my_scripts.js*.

```
body{
     background-color: #000;
     color: white;
}

/* Стиль вкладок */
#main {
  color:#111;
  width:500px;
  margin:8px auto;
}
#main > li, #main > ul > li
{ list-style:none; float:left; }
#main ul a {
  display:block;
  padding:6px 10px;
  text-decoration:none!important;
  margin:1px 1px 1px 0;
  color:#FFF;
  background:#444;
}
```

Комментарий CSS

Весь дальнейший код CSS предназначен для создания вкладок.

```
#main ul a:hover {
  color:#FFF;
  background:#111;
}
#main ul a.selected {
  margin-bottom:0;
  color:#000;
  background:snow;
  border-bottom:1px solid snow;
  cursor:default;
}
#main div {
  padding:10px 10px 8px 10px;
  *padding-top:3px;
  *margin-top:-15px;
  clear:left;
  background:snow;
  height: 300px ;
}
#main div a {
  color:#000; font-weight:bold;
}
```

my_style.css

Создание ссылок, которые будут преобразованы во вкладки модулем расширения.

Элементы div для хранения содержимого вкладок

```
<div id="main">
    <ul class="idTabs">
        <li><a href="#about">About the Race</a></li>
        <li><a href="#finishers">All Finishers</a></li>
    </ul>
    <div id="about">
        <h4>About the race</h4>This race Bit to Byte Campaign!
    </div>
    <div id="finishers">
        <h4>All Finishers</h4>
            <ul id="finishers_all">
                <li>Name: Bob Hope. Time: 25:30</li>
                <li>Name: John Smith. Time: 25:31</li>
                <li>Name: Jane Smith. Time:  25:44</li>
                ...
            </ul>
        </div>
...
<script src="scripts/jquery-1.6.2.min.js"></script>
<script src="scripts/my_scripts.js"></script>
<script src="scripts/jquery.idTabs.min.js"></script>
```

Часть информации о прошлогодних участниках закодирована в странице. Тем, кто занимался обновлением, не позавидуешь...

Как обычно, включаем файлы JavaScript. Этот же механизм используется для подключения модулей расширения.

index.html

```
$(document).ready(function(){
    getTime();
    function getTime(){
      var a_p = "";
      var d = new Date();
      var curr_hour = d.getHours();
      (curr_hour < 12) ? a_p = "AM" : a_p = "PM";
      (curr_hour == 0) ? curr_hour = 12 : curr_hour = curr_hour;
      (curr_hour > 12) ? curr_hour = curr_hour - 12 : curr_hour = curr_hour;
      var curr_min = d.getMinutes().toString();
      var curr_sec = d.getSeconds().toString();
      if (curr_min.length == 1) { curr_min = "0" + curr_min; }
      if (curr_sec.length == 1) { curr_sec = "0' + curr_sec; }
      $('#updatedTime').html(curr_hour + ":" + curr_min + ":" + curr_sec + " " + a_p );
    }
});
```

Вызов пользовательской функции getTime

Новый экземпляр объекта JavaScript Date

Методы объекта Date

Тернарный оператор JavaScript (подробности чуть позже)

my_scripts.js

Даешь динамику!

Заказчик хочет, чтобы страница обновлялась практически в реальном времени, так что от жесткого кодирования результатов в файле HTML придется отказаться. А код JavaScript прежде использовался только для обновления времени на странице! А нам представилась идеальная возможность поднять свои навыки jQuery на следующий уровень. jQuery, JavaScript, немного Ajax и XML... И веб-приложения следующего поколения начинают выглядеть как *динамические* (в отличие от *статических*) *настольные приложения, оперативно реагирующие на происходящие события.*

Ajax (сокращение от Asynchronous JavaScript and XML, т. е. «Асинхронный JavaScript и XML») — механизм передачи структурированных данных между веб-сервером и браузером, без участия посетителя сайта. При использовании Ajax ваши страницы и приложения запрашивают у веб-сервера только то, что им действительно необходимо — только те части страницы, которые должны измениться, и веб-сервер предоставляет им только эти данные. Это приводит к сокращению трафика, более компактным обновлениям и ускорению перерисовки.

А самое лучшее, что страницы AJax строятся на базе стандартных интернет-технологий, которые вы уже встречали в этой книге или уже знали прежде:

- HTML
- CSS
- JavaScript
- The DOM

Для использования Ajax необходимо поближе познакомиться с весьма популярным форматом данных XML и с методом jQuery для обработки Ajax-запросов `ajax`.

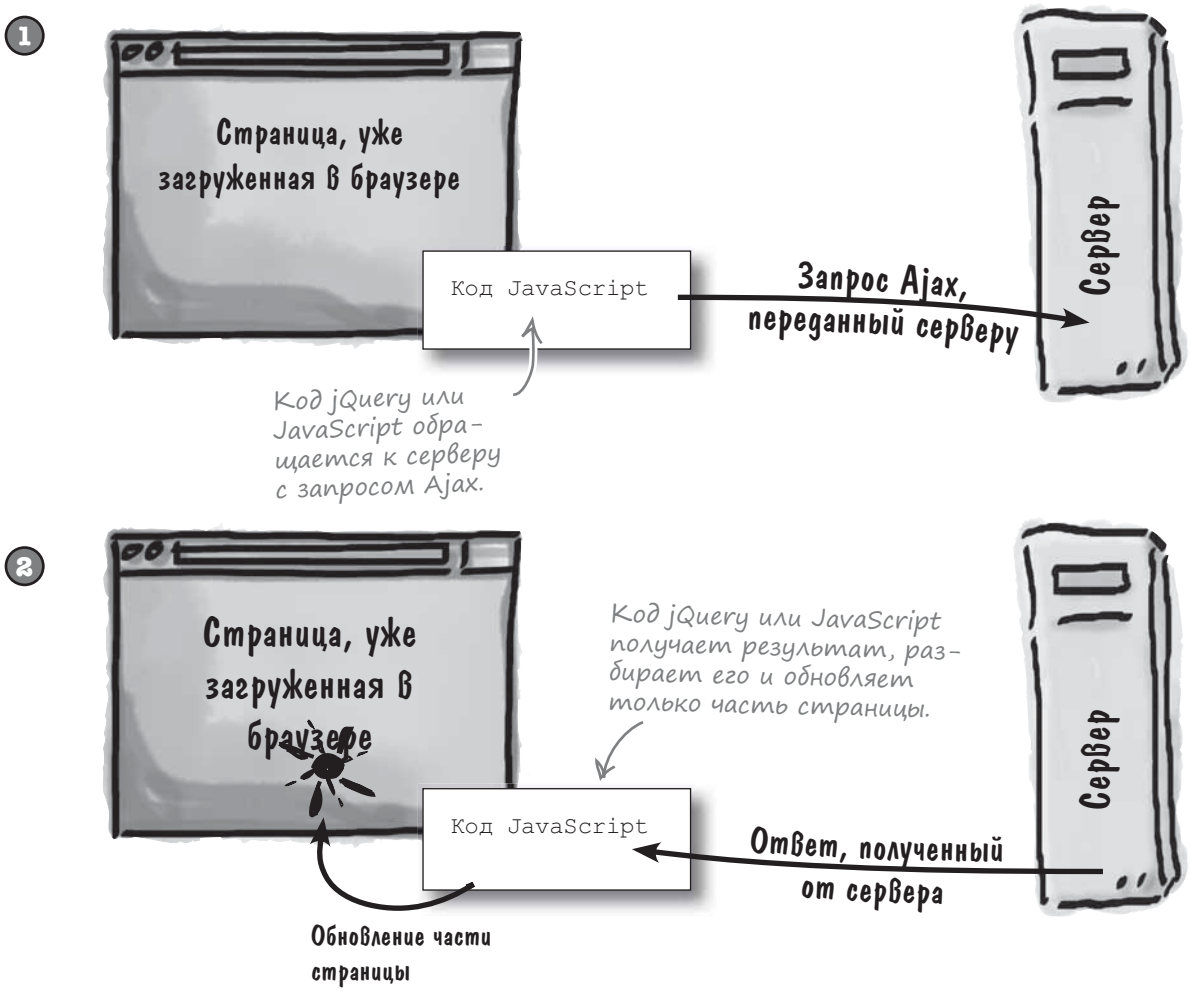
При использовании Ajax ваши веб-страницы запрашивают у сервера необходимую информацию в тот момент (и там), когда это нужно.

СТАРЫЕ и НОВЫЕ Веб-технологии

Несмотря на то что мы уже кое-что знаем о jQuery, работа с данными угрожает затянуть нас в эпоху старых веб-технологий, когда для частичного или полного обновления данных нам приходилось обновлять целую страницу или переходить к другой странице. А вместе со старыми веб-технологиями возвращается и эпоха медлительных сайтов, когда с сервера каждый раз приходилось заново запрашивать всю страницу. Какой прок в изучении классных возможностей JavaScript, если работа с данными снова все замедлит?

Знакомьтесь: Ajax

Ajax позволяет организовать динамический обмен данными с сервером. Используя Ajax и манипуляции с DOM в коде jQuery и JavaScript, можно загружать или обновлять только часть страницы.



Структура Ajax

Как упоминалось ранее, Ajax — механизм передачи структурированных данных между веб-сервером и браузером без участия посетителя сайта. Но в действительности Ajax — не монолит, а комбинация нескольких разных технологий для построения впечатляющих, интерактивных веб-приложений. JavaScript позволяет взаимодействовать со структурой DOM страницы. Асинхронность означает, что передача данных может осуществляться в фоновом режиме, без нарушения работы страницы или участия пользователя, взаимодействующего со страницей. А буква «X» в названии относится исключительно к формату данных.

Что такое Ajax?

Асинхронный

JavaScript обращается с запросом к серверу. В это время вы можете взаимодействовать со страницей: вводить данные в формах, даже нажимать кнопки, пока веб-сервер работает где-то на заднем плане. Затем, когда сервер вернет результаты своей работы, ваш код может обновить только измененную часть страницы. Вам вообще не приходится ждать! Такова сила асинхронных запросов!

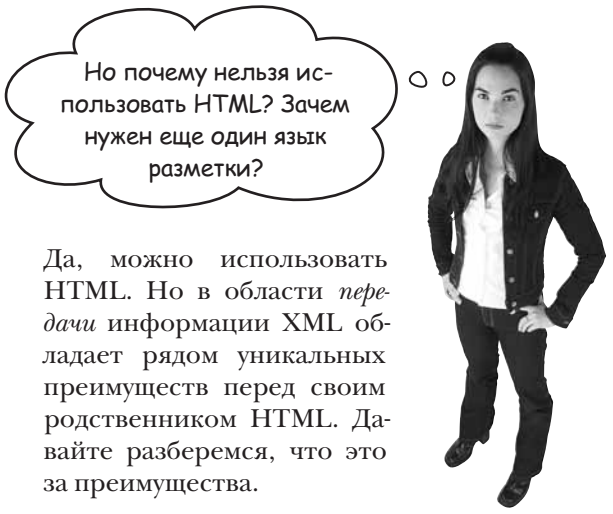
XML

XML (eXtensible Markup Language) — спецификация хранения информации, а также описания структуры этой информации. И хотя XML является языком разметки (как и HTML), он не имеет собственных тегов. XML позволяет автору разметки создать любые теги, которые ему потребуются.

A J A X

JavaScript

JavaScript, как вы уже знаете, — язык сценариев, используемый при разработке веб-приложений, прежде всего для создания функций, встраиваемых или включаемых в документы HTML и взаимодействующих с DOM.



Да, можно использовать HTML. Но в области *передачи* информации XML обладает рядом уникальных преимуществ перед своим родственником HTML. Давайте разберемся, что это за преимущества.

Фактор «X»

XML — сокращение от слов «e**X**tensible **M**arkup **L**anguage» (т. е. «расширяемый язык разметки»). Это широко распространенный, стандартизированный способ представления данных и текста в формате, который может обрабатываться без существенного вмешательства человека. Информация, отформатированная в XML, может передаваться между разными платформами, приложениями и даже разными языками (как языками программирования, так и естественными). Она также может использоваться в широком спектре средств разработчика и служебных программ. Данные в формате XML легко создаются и редактируются; все, что для этого необходимо — это простой текстовый редактор и объявление XML в начале файла. А остальное зависит только от вас!

XML ничего не ДЕЛАЕТ!

Это может прозвучать немного странно, но XML сам по себе практически ничего не делает. Задача XML — структурирование и хранение информации для передачи. По сути, XML является метаязыком для описания языков разметки. Иначе говоря, XML предоставляет механизм описания тегов и структурных отношений между ними. Важно понимать, что XML не заменяет HTML, а лишь дополняет его. Во многих веб-приложениях XML используется для форматирования данных для передачи, тогда как HTML используется для форматирования и отображения данных. Посмотрим, как выглядит файл XML с информацией о книгах.

Объявление XML должно присутствовать в каждом документе. В нем определяется версия XML, используемая в документе.

Потомки корневого узла (открывающие и закрывающие теги). В данном случае описываются книги.

```
<?xml version="1.0" encoding="utf-8"?>
<books>                                   Корневой узел (или тег)
    <book>
      <title>The Hitchhikers Guide to the Galaxy</title>
      <author>Douglas Adams</author>
      <year>1980</year>
    </book>
    <book>
      <title>The Color of Magic</title>
      <author>Terry Pratchett</author>
      <year>1983</year>
    </book>
    <book>
      <title>Mort</title>
      <author>Terry Pratchett</author>
      <year>1987</year>
    </book>
    <book>
      <title>And Another thing...</title>
      <author>Eoin Colfer</author>
      <year>2009</year>
    </book>
</books>                                  Закрытие корневого узла (или тега)
```

Другие теги используются для хранения данных.

books.xml

XML используется для форматирования данных с целью передачи, тогда как HTML используется для форматирования и отображения данных.

Часто Задаваемые Вопросы

В: Главное достоинство XML — возможность создания своих тегов?

О: Точно! Очень удобно определять элементы и структуры, соответствующие потребностям вашей задачи. Что еще лучше, формат XML стандартизирован, поэтому очень многие люди умеют работать с ним.

В: А не проще ли определить собственный формат данных?

О: На первый взгляд кажется, что проще, но закрытые форматы данных (которые вы создаете для себя) создают проблемы. Если их не документировать, пользователи забудут, как они работают. А если что-нибудь изменится, вам придется проследить за обновлением всего: клиента, сервера, базы данных, документации… Это весьма хлопотно.

В: И что, XML действительно так широко используется?

О: Благодаря гибкости при создании любых необходимых структур данных, язык XML стал основой многих языков разметки в Web. Сейчас существует более 150 разных типов языков, основанных на XML: RSS (RDF Site Summary или Real Simple Syndication) для создания каналов новостей или аудио/видеоматериалов; KML (Keyhole Markup Language) для географической информации, используемой в Google Earth; OOXML (Office Open XML) для файлов текстовых процессоров, предложенный Microsoft; SVG (Scalable Vector Graphics) для описаний двумерных векторных изображений; SOAP (Simple Object Access Protocol) для определений методов обмена информацией с веб-службами... Количество применений XML огромно!

В: Ладно, я понимаю, почему XML стоит использовать, но разве он не превращается в «закрытый формат данных», когда мы начинаем объявлять имена элементов?

О: Нет, не превращается. Именно в этом и заключается изящество XML: он гибок. Сервер и клиент должны работать с одинаковыми именами элементов, но это соответствие часто может быть согласовано на стадии выполнения.

В: А разве не все веб-страницы асинхронны? Скажем, браузер загружает графику, когда я уже просматриваю страницу...

О: *Браузеры* асинхронны, а стандартные веб-страницы — нет. Когда веб-странице требуется информация от программы на стороне сервера, обработка приостанавливается, пока сервер не ответит... если только страница не выдаст асинхронный запрос. А для решения этой задачи как раз и предназначается Ajax.

В: Как определить, когда стоит использовать Ajax и асинхронные запросы?

О: Руководствуйтесь простым правилом: если вы хотите, чтобы какие-то действия выполнялись, когда пользователь продолжает работать, вам нужен асинхронный запрос. Но если для продолжения работы пользователю необходима информация или ответ от вашего приложения, пускай подождет. В таких ситуациях используют синхронные запросы.

В: Разве для взаимодействия с XML не нужно использовать XHTML?

О: XHTML — *это и есть* XML. Язык XHTML не так близок к HTML, как считается. У XHTML более жесткие требования к парсингу кода, он происходит из того же семейства, что и XML. Но это не означает, что он может взаимодействовать с XML лучше, чем HTML. В разметке, приведенной в книге, используется HTML5, который будет включать в себя XHTML5 после публикации спецификаций.

Уговорили, я уже хочу использовать Ajax. Но ведь начинать нужно с проработки структуры, верно? Так было всегда...

Вы правы.

Давайте разберемся со структурой, чтобы мы могли перейти к включению поддержки Ajax в наши страницы...

Развлечения с магнитами

Сложите из магнитов код создания двух новых вкладок, на которых отображается информация о результатах участников: мужчин (идентификатор `male`) и женщин (идентификатор `female`). Вкладку **About** можно убрать, но вкладка **All Finishers** должна остаться. В каждом разделе создайте пустой элемент `ul` для информации об участниках. Также удалите все существующее содержимое из элемента `ul finishers_all`.

index.html

```
<body>
    <header>
        <h2>_____</h2>
    </header>
    <div id="main">
        <ul class="idTabs">
            <li><a href="_____">Male Finishers</a></li>
            <li><a href="#female">_____</a></li>
            <li><a href="#all">All Finishers</a></li>
        </ul>
        <div id="male">
            <h4>Male Finishers</h4>
            <ul id="_____"></ul>
        </div>
        <div _____>
            <h4>Female Finishers</h4>
            <ul id="finishers_f"></ul>
        </div>
        <div _____>
            <h4>All Finishers</h4>
            <ul id=_____></ul>
        </div>
    </div>
    <footer>
        <h4>Congratulations to all our finishers!</h4>
        <br>Last Updated: <div id=_____></div>
    </footer>
    <script src="scripts/jquery-1.6.2.min.js"></script>
    <script src=_____></script>
    <script src="scripts/jquery.idTabs.min.js"></script>
</body>
```

"scripts/my_scripts.js"

#male

"finishers_all"

2011 Race Finishers!

"updatedTime"

Female Finishers

finishers_m

id="female"

id="all"

Развлечения с магнитами. Решение

Теперь на странице должны появиться две новых вкладки, для мужчин и для женщин.

index.html

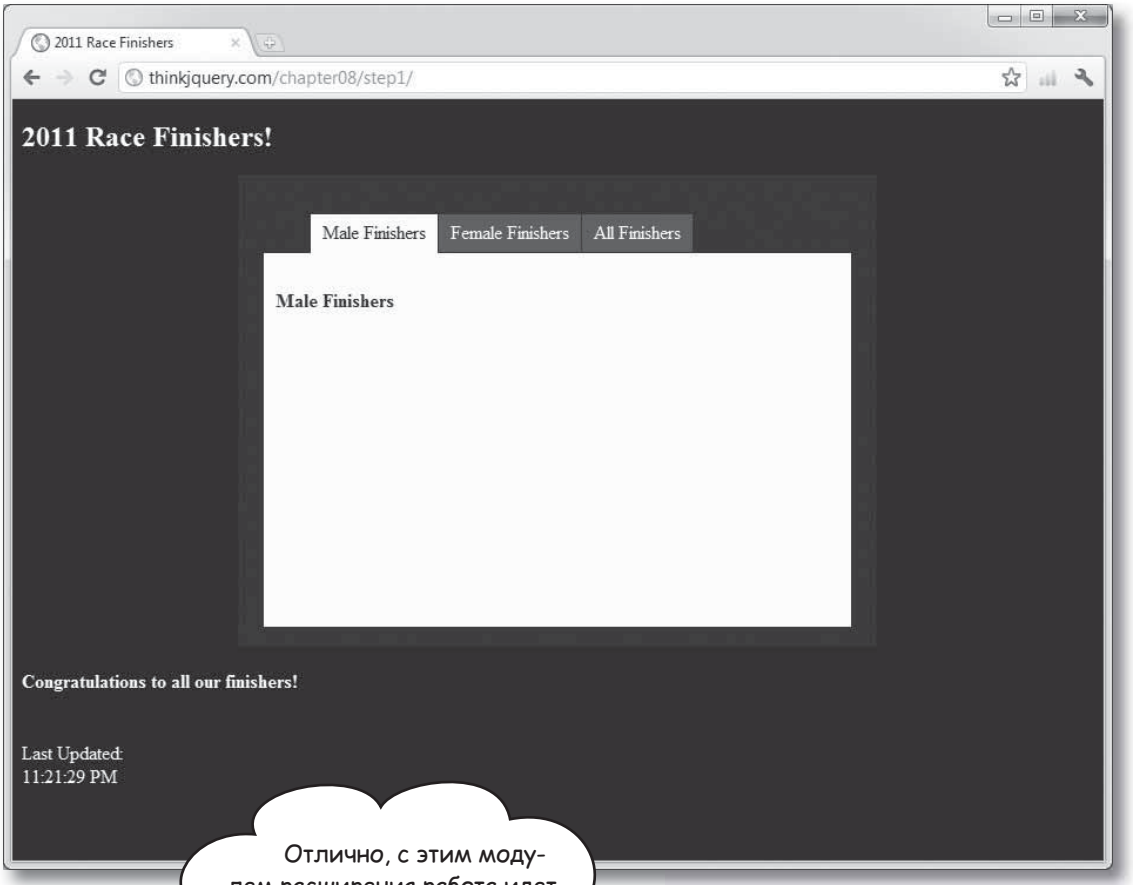
```
<body>
    <header>
        <h2>2011 Race Finishers!</h2>
    </header>
    <div id="main">
        <ul class="idTabs">
            <li><a href="#male">Male Finishers</a></li>
            <li><a href="#female">Female Finishers</a></li>
            <li><a href="#all">All Finishers</a></li>
        </ul>
        <div id="male">
            <h4>Male Finishers</h4>
            <ul id="finishers_m"></ul>
        </div>
        <div id="female">
            <h4>Female Finishers</h4>
            <ul id="finishers_f"></ul>
        </div>
        <div id="all">
            <h4>All Finishers</h4>
            <ul id="finishers_all"></ul>
        </div>
    </div>
    <footer>
        <h4>Congratulations to all our finishers!</h4>
        <br>Last Updated: <div id="updatedTime"></div>
    </footer>
    <script src="scripts/jquery-1.6.2.min.js"></script>
    <script src="scripts/my_scripts.js"></script>
    <script src="scripts/jquery.idTabs.min.js"></script>
</body>
```

Наш список победителей

Включение модуля расширения jQuery необходимо для создания вкладок.

ТЕСТ-ДРАЙВ

Включите в файл *index.html* код, созданный нами в упражнении с магнитами, и откройте страницу в своем любимом браузере.



Хорошая работа!

Страница постепенно принимает нужный вид. Теперь давайте посмотрим, как получить данные с сервера, чтобы заполнить каждую вкладку актуальной информацией.

Получение данных методом ajax

Нужны данные? jQuery и Ajax их вам с удовольствием предоставят. Метод jQuery ajax возвращает объект (еще не забыли главу 6?) с данными о конкретной операции, которую вы пытаетесь выполнить. Метод `ajax` **может получать много разных параметров; в частности, с его помощью можно** *отправить* **данные серверу (POST) или же** *запросить* **их с сервера (GET).**

Сокращение jQuery

Метод jQuery ajax

```
$.ajax({
    url: "my_page.html"
    success: function(data){
    }
});
```

URL-адрес данных, запрашиваемых через Ajax

Эта функция выполняется в том случае, если метод ajax был выполнен успешно. Вскоре мы напишем код этой функции.

Данные, возвращаемые в результате вызова ajax

Полный список всех параметров метода приведен на сайте документации jQuery по адресу *http://api.jquery.com/jQuery.ajax/*. Также в jQuery существуют вспомогательные методы для работы с вызовами Ajax. Вскоре мы вернемся к ним, честное слово.

А пока включите в файл *my_scripts.js* следующий код (только строки, выделенные жирным шрифтом).

```
$(document).ready(function(){
    $.ajax({
        url: "finishers.xml",
        cache: false,
        dataType: "xml",
        success: function(xml){
        }
    });

    getTime();

    function getTime(){
        var a_p = "";
        var d = new Date();
```

Загружаем файл finishers.xml средствами Ajax.

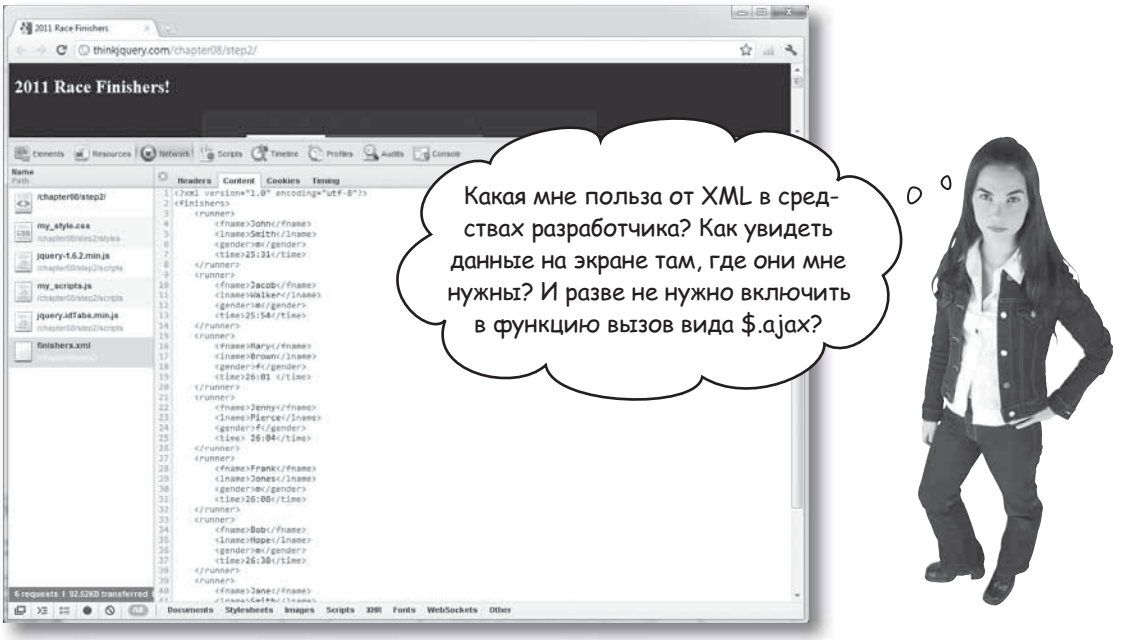
Параметр обеспечивает локальное кэширование результатов, снижающее количество обращений к серверу.

Тип данных, которые мы рассчитываем получить от сервера.

my_scripts.js

ТЕСТ-ДРАЙВ

Включите в файл *my_scripts.js* код с предыдущей страницы. Затем загрузите файл с данными XML для этой главы по адресу *http://thinkjquery.com/chapter08/step2/finishers.xml* и сохраните его в одном каталоге с файлом *index.html*. Когда это будет сделано, откройте файл *index.html* в браузере и перейдите на вкладку «Network» (в средствах разработчика в Google Chrome) или на вкладку «Net» (в Firebug для Firefox). Ваш файл XML должен быть здесь вместе с остальными файлами страницы.



Все верно.

Теперь, мы знаем, что файл XML загружается, нужно выбрать нужный текст и отобразить его на экране. Нам следует перебрать данные *всех* участников марафона, чтобы включить их в нужный список на странице. И действительно, вызовы ajax полезно включить в функции, чтобы их было удобно вызывать в случае надобности.

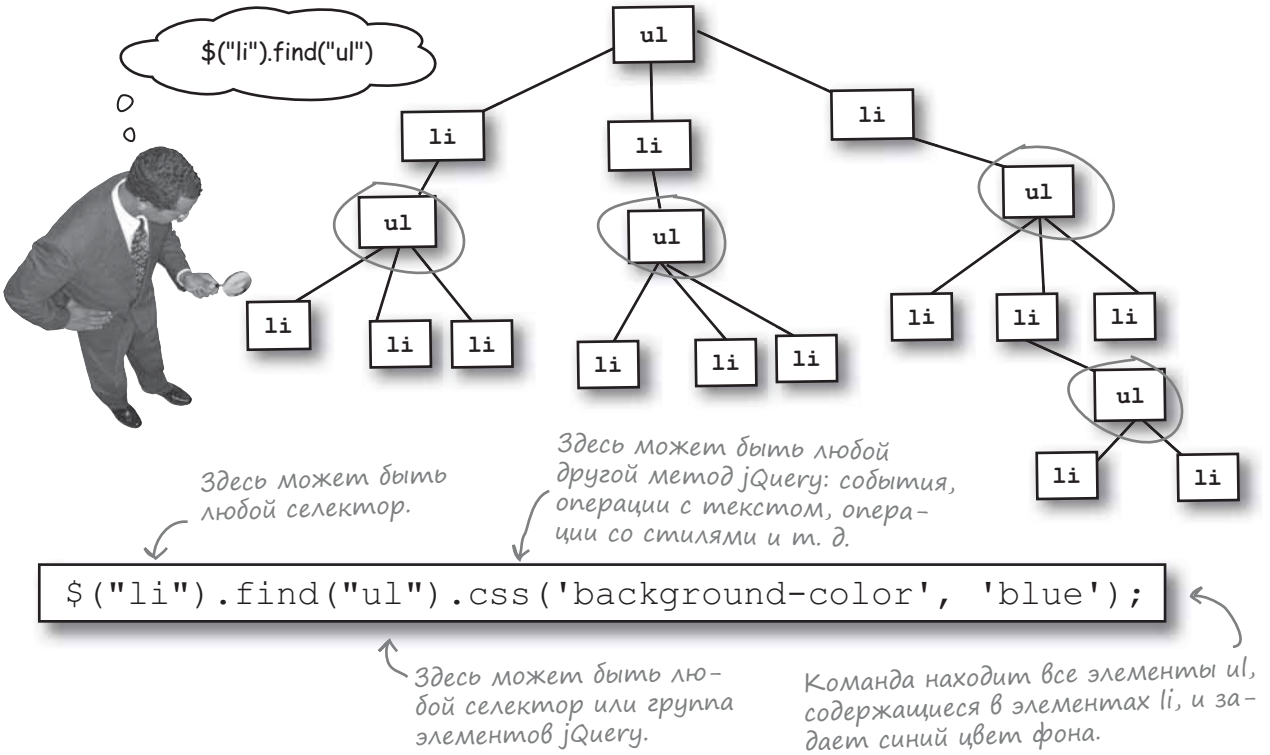
Будьте осторожны!

Вызовы ajax подчиняются политике единого домена!

Политика единого домена — концепция безопасности для JavaScript и других сценарных языков на стороне клиента. Она позволяет сценариям, выполняемым на странице, обращаться к ресурсам (таким как свойства и методы элементов) ***на том же сервере***. Обращения к элементам страниц других серверов блокируются. Проверка не распространяется на директивы JavaScript include, но наш файл XML ей подвержен. Это означает, что он должен находиться **на одном сервере с загружающей его страницей**.

Разбор данных XML

Мы должны как-то извлечь данные каждого участника из файла XML и отобразить их на экране. К счастью, в jQuery имеется метод `find`, который ищет элементы по заданному критерию. Метод `find` перебирает потомков элементов в структурированном, иерархическом наборе данных (таком как дерево DOM или документ XML) и строит новый массив с подходящими элементами. Методы `find` и `children` похожи (метод `children` рассматривался в главе 4, когда мы строили вегетарианское меню), однако `children` переходит только на один уровень вниз по дереву DOM, а нам, возможно, потребуется спуститься глубже.



Сочетание методов find и each позволяет найти группу элементов по некоторому условию и в цикле обработать каждый найденный элемент.

МОЗГОВОЙ ШТУРМ

Как вы думаете, с какими компонентами документа XML необходимо взаимодействовать для отображения результатов участников на экране?

Развлечения с магнитами

Расставьте магниты так, чтобы у вас получилась функция `getXMLRacers`, которая вызывает метод `ajax` и загружает файл *finishers.xml*. После успешной загрузки функция стирает все текущее содержимое списков, после чего находит всех участников в файде XML, определяет пол каждого участника, присоединяет его к соответствующему списку и в любом случае включает участника в список `finishers_all`. Затем вызов функции `getTime` обновляет текущее время на странице.

```
function _____________
    $.ajax({
      url: ___________________,
      cache: false,
      dataType: "xml",
      _________ function(xml){
        $(_________________).empty();
        $('#finishers_f')___________;
        $('#finishers_all').empty();
        $(xml).find_________________(function() {
          var info = '<li>Name: ' + $(this).find_________________ + ' ' + $(this).
find("lname").text() + '. Time: ' + _____________________.text() + '</li>';
          if( $(this).find("gender").text() == "m" ){
            $('#finishers_m').append_______
          }else if ( $(this).find("gender").text() == "f" ){
            ____________________.append(info);
          }else{  }
            ____________________.append(info);
        });
        ___________
      }
    });
}
```

"finishers.xml"

getXMLRacers(){

getTime();

success:

(info)

$('#finishers_all')

'#finishers_m'

("runner").each

$('#finishers_f')

.empty()

("fname").text()

$(this).find("time")

my_scripts.js

Развлечения с магнитами. Решение

Используя `find` и `each`, мы перебираем файл *finishers.xml*, проверяем пол каждого участника, а затем размещаем его данные на соответствующей вкладке веб-приложения.

```
function getXMLRacers(){
    $.ajax({
       url: "finishers.xml",
       cache: false,
       dataType: "xml",
       success: function(xml){
          $('#finishers_m').empty();
          $('#finishers_f').empty();
          $('#finishers_all').empty();
          $(xml).find("runner").each(function() {
             var info = '<li>Name: ' + $(this).find("fname").text() + ' ' + $(this).
find("lname").text() + '. Time: ' + $(this).find("time").text() + '</li>';
             if( $(this).find("gender").text() == "m" ){
                $('#finishers_m').append(info)
             }else if ( $(this).find("gender").text() == "f" ){
                $('#finishers_f').append(info);
             }else{  }
                $('#finishers_all').append(info);
          });
          getTime();
       }
    });
}
```

Очистка всех элементов ul для заполнения обновленными данными

Перебор всех элементов runner в файле XML

Проверка пола каждого участника для включения в соответствующий список

Кроме того, все участники включаются в список finishers_all.

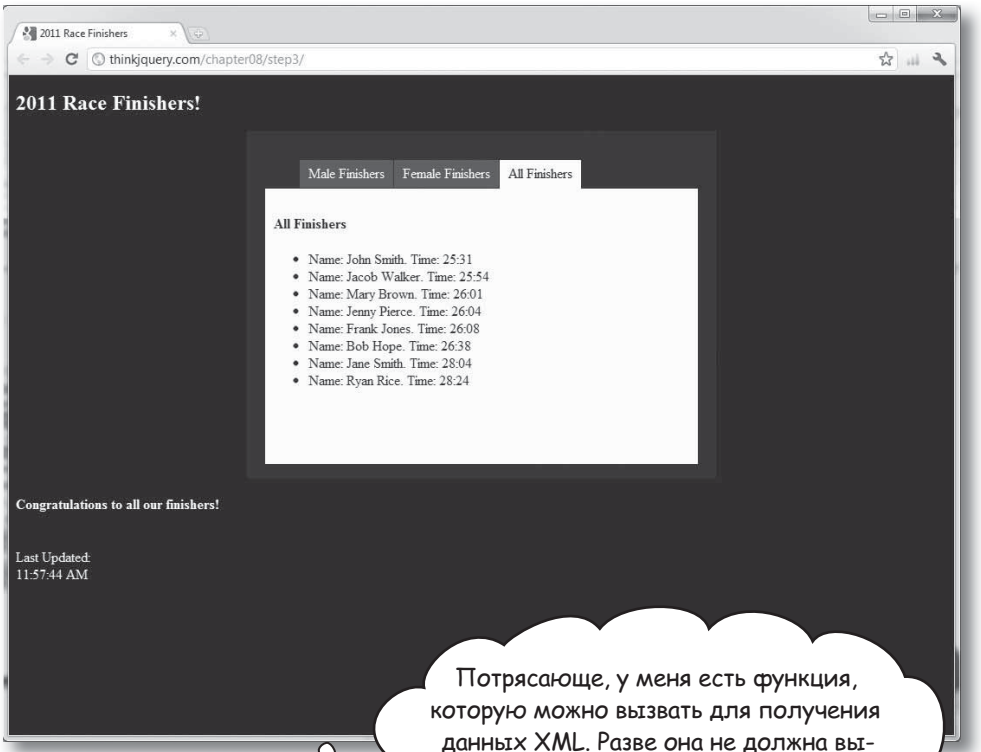
Вызов функции getTime для включения в страницу обновленного времени последнего вызова функции getXMLRacers

my_scripts.js

В этом примере строка, начинающаяся с «var info = ...», оказалась слишком длинной, поэтому мы разбили ее надвое. Вам в своем коде делать это необязательно.

ТЕСТ-ДРАЙВ

Включите в файл *my_scripts.js* функцию `getXMLRacers`. Также замените вызов функции `getTime` (в секции `document.ready`) вызовом функции `getXMLRacers`. Функция `getTime` теперь вызывается в нашей новой функции. Проследите за тем, чтобы код выполнялся через веб-сервер. Так что URL-адрес, например, должен начинаться с префикса *http://*, а не *file://*. Также убедитесь в том, что файл XML находится на одном сервере с файлом HTML; в противном случае у вас возникнут проблемы с нарушением политики единого домена.



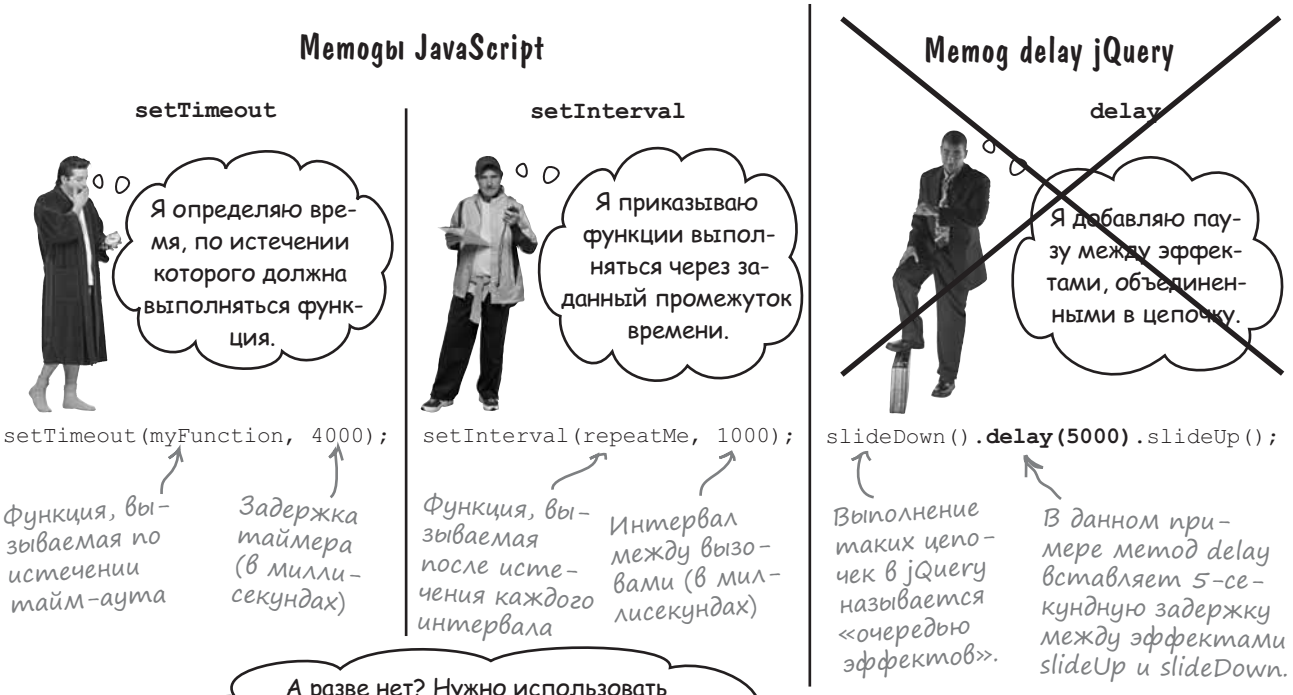
Потрясающе, у меня есть функция, которую можно вызвать для получения данных XML. Разве она не должна вызываться многократно, чтобы обновлять страницу автоматически?

Да, должна.

К счастью, в предыдущих главах вы уже видели, как запланировать регулярное выполнение какой-либо операции на странице. Давайте еще посмотрим, как это делается и какие варианты возможны на этот раз...

Планирование событий

Как было показано в предыдущей главе, JavaScript и jQuery поддерживают методы для вызова функций с течением времени. У объекта JavaScript `window` есть четыре метода для работы с таймером: `setTimeout`, `clearTimeout`, `setInterval` и `clearInterval`. jQuery поддерживает метод `delay`, но этот метод предназначен для создания эффектов и не поддерживает возможности планирования или повторения действий. Значит, этот метод в нашем случае бесполезен...



Не торопитесь!

Это далеко не очевидно. Метод `setInterval` обычно используется для планирования периодических событий на странице, а при зависимости от внешних ресурсов (таких как наш файл данных) он может создать проблемы.

Будьте осторожны!

С `setInterval` следующий вызов произойдет даже, если вызываемая функция еще не завершила работу.

Если вы ожидаете информации с другого сервера или реакции пользователя, `setInterval` *может повторно вызвать функцию до получения результатов первого вызова. Порядок возвращения результатов необязательно соответствует порядку вызова функций.*

Самоактивизируемые функции

Самоактивизируемая функция вызывает саму себя в ходе своего нормального выполнения. Такие функции бывают особенно полезны, когда вам нужно дождаться завершения текущей операции, выполняемой функцией, прежде чем выполнять ее снова. Объединяя самоотносимость с вызовом `setTimeout`, можно запланировать выполнение функции, но перейти к нему только, если предыдущий вызов функции был успешным. В противном случае управление в точку вызова передано не будет, а следовательно, функция не будет вызвана повторно.

Возьми в руку карандаш

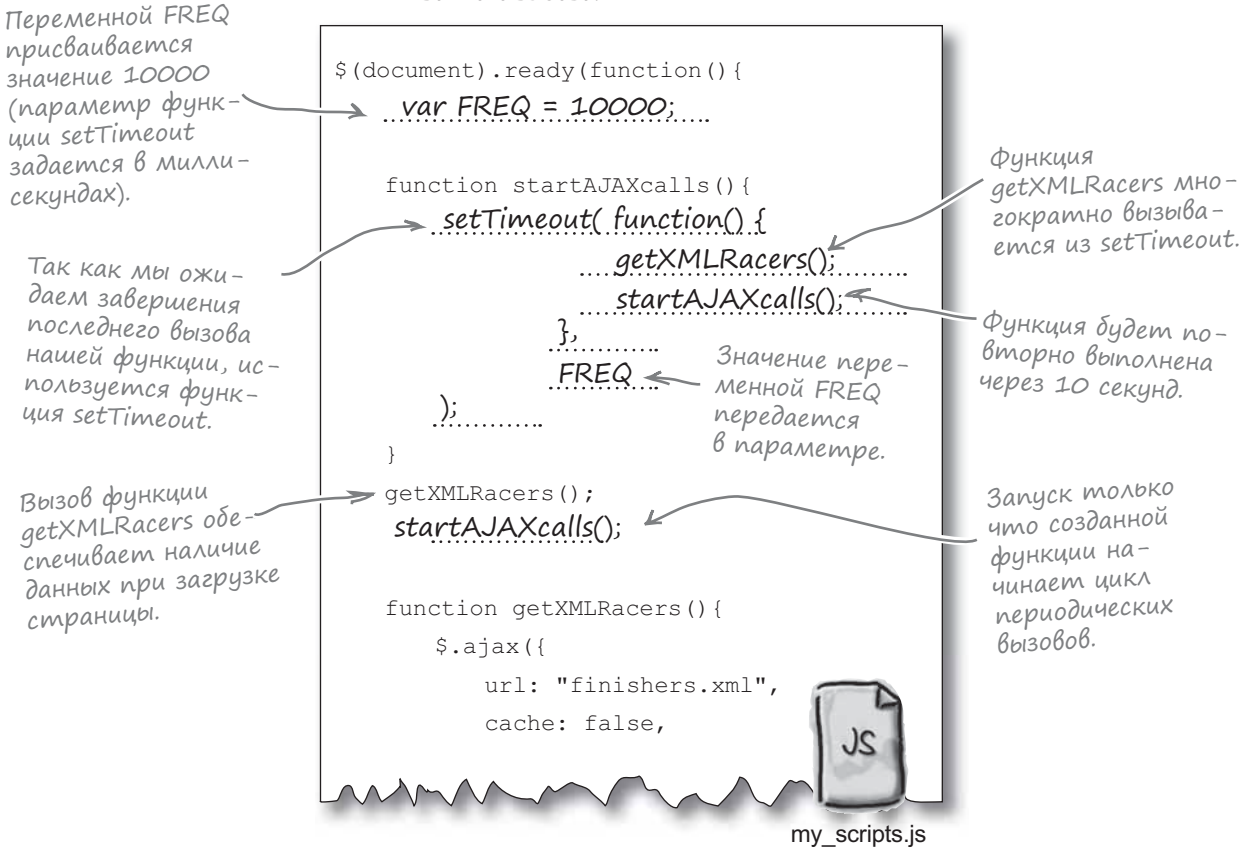
Создайте функцию с именем `startAJAXcalls`, которая вызывается при загрузке страницы и которая вызывает функцию `getXMLRacers` каждые 10 секунд. Определите в начале сценарного файла (в секции `$(document).ready`) переменную с именем `FREQ`, которой присваивается периодичность повторных вызовов `getXMLRacers` в миллисекундах. Используйте `setTimeout` для вызова функции `startAJAXcalls` с целью ее самоактивизации после завершения `getXMLRacers`. Также включите в код вызов функции `startAJAXcalls` для запуска таймера.

```
$(document).ready(function(){
    ……..

    function startAJAXcalls(){
        ……..
                    ……..
                    ……..
                ……..
                ……..
        ……..
    }
    getXMLRacers();
        ……..

    function getXMLRacers(){
        $.ajax({
            url: "finishers.xml",
            cache: false,
```

my_scripts.js

Возьми в руку карандаш
Решение

Функция `startAJAXcalls` вызывает `setTimeout` для вызова `getXMLRacers`, чтобы получить данные XML и вызвать саму себя. Такой самовызов гарантирует, что в следующий раз функция будет вызвана после завершения предыдущего вызова. Мы предотвращаем накопление запросов к серверу при низкой пропускной способности сети или если ответ от сервера не успел вернуться до момента следующего запланированного вызова.



Часто Задаваемые Вопросы

В: Во всех книгах об Ajax говорится, что я должен использовать объект `XMLHttpRequest`. Это так?

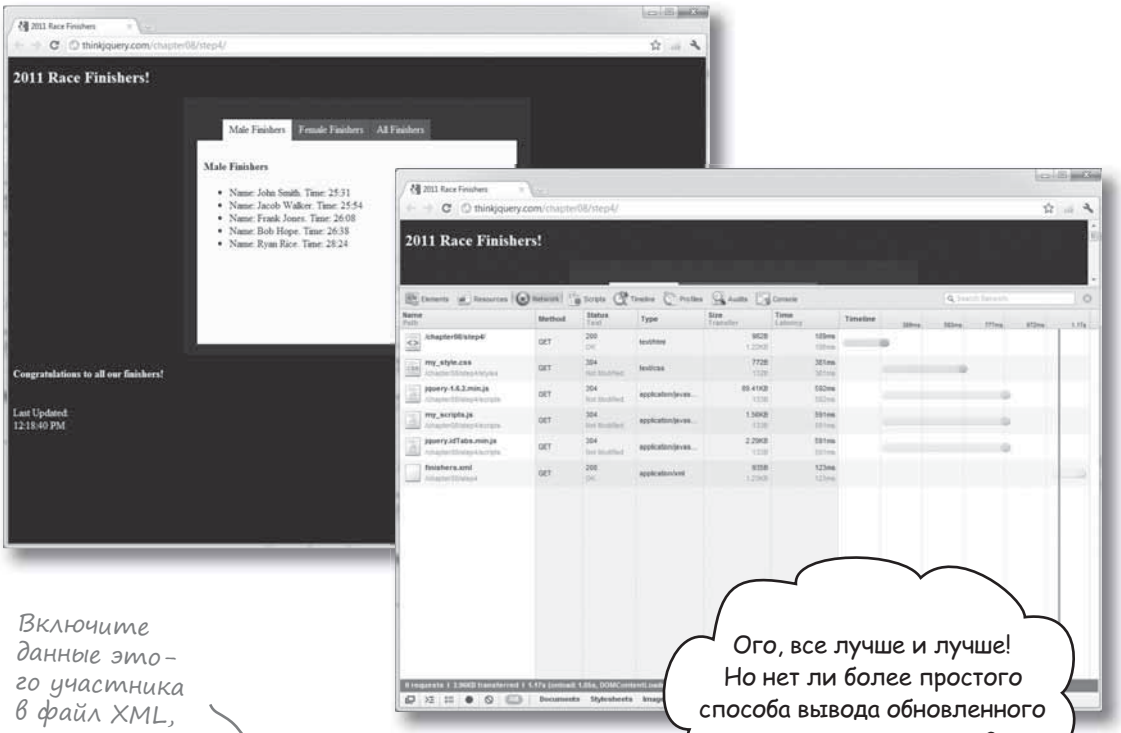
О: Да, но не в jQuery. Веб-программисту не нужно работать с этим объектом напрямую. jQuery делает это за вас при использовании метода `ajax`. Кроме того, поскольку вызовы Ajax зависят от браузера, jQuery вычисляет оптимальный способ выполнения запросов Ajax для каждого посетителя сайта.

В: А что произойдет, если сервер вернет ошибку или не ответит? Приложение так и будет ждать?

О: Нет, бесконечное ожидание вам не грозит. Продолжительность тайм-аута задается в одном из параметров вызова `ajax`. Кроме того, по аналогии с параметром `success`, позволяющим выполнить функцию при успешном завершении, также существуют другие параметры для обработки событий `error`, `complete` и т. д.

Тест-драйв

Обновите файл *my_scripts.js* только что написанным кодом. Не забудьте включить вызов новой функции сразу же после вызова функции `getXMLRacers` в конце сценария. Откройте страницу в браузере, воспользуйтесь средством «Network» Google Chrome или «Net» в Firebug для Firefox и убедитесь в том, что файл загружается каждые 10 секунд. После проверки измените файл XML в своем любимом текстовом редакторе (используйте приведенные ниже данные) и убедитесь в том, что данные нового участника появились на странице... (Не забудьте сохранить файл XML после обновления!)



Включите данные этого участника в файл XML, сохраните его. Убедитесь в том, что данные будут автоматически загружены на страницу.

```
<runner>
    <fname>Justin</fname>
    <lname>Jones</lname>
    <gender>m</gender>
    <time>29:14</time>
</runner>
```

Ого, все лучше и лучше! Но нет ли более простого способа вывода обновленного времени на странице?

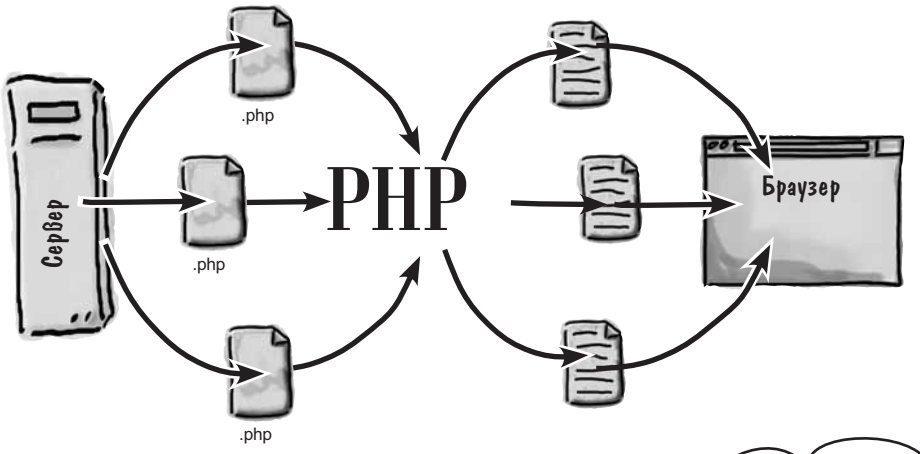
Сервер нам поможет

Как вы уже видели, HTML хорошо подходит для отображения информации на странице, а XML — для форматирования передаваемых данных. А если вы хотите, чтобы страница что-то делала, например определяла текущее время или *читала информацию из базы данных?* Конечно, можно выйти из положения при помощи jQuery и JavaScript, но почему бы не воспользоваться технологией, предназначенной специально для таких задач?

Серверные языки приходят на помощь!

Существует много разных серверных языков (например, JSP, ASP и Cold Fusion), но наше внимание будет сосредоточено на одном из них: PHP.

PHP (сокращение от PHP: Hypertext Processor; да, вот такое рекурсивное сокращение — не спрашивайте нас, почему!) — бесплатный серверный язык сценариев общего назначения, используемый для построения динамических веб-страниц. Файлы, содержащие код PHP, выполняются на сервере; сгенерированный код HTML передается браузеру для отображения. Технология PHP более подробно описана в следующей главе, а пока давайте посмотрим, как она поможет нам с реализацией «обновляемого времени».

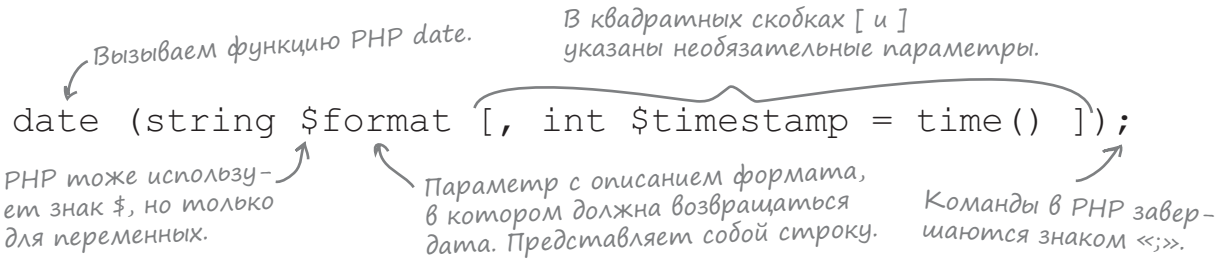


PHP используется для динамического генерирования разметки HTML, которая затем отображается в браузере.

Стоп! Если вы не выполнили инструкции из Приложения II по установке PHP и MySQL, то следующая страница работать не будет. Для нее необходима рабочая поддержка PHP. Сделайте это, прежде чем читать дальше.

Который час?

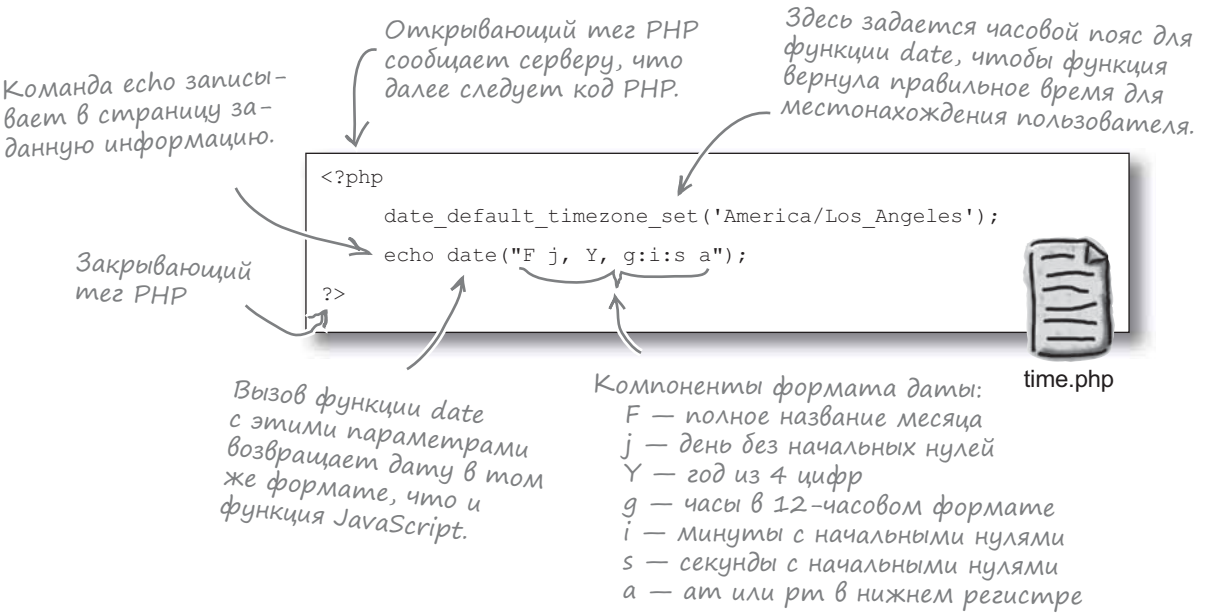
Ладно, признаемся: в JavaScript есть готовая функция для получения времени. Но использовать такую большую, сложную функцию для такой простой задачи... К счастью, PHP предоставляет очень простой способ получения времени с использованием функции `date`. Как и функции, которые мы создавали ранее, эта функция получает несколько параметров и возвращает дату в формате, описанном переданными параметрами. Главный параметр определяет, как должна отображаться дата. Давайте присмотримся повнимательнее:



За полным списком параметров функции `date` обращайтесь по адресу *http://php.net/manual/en/function.date.php*.

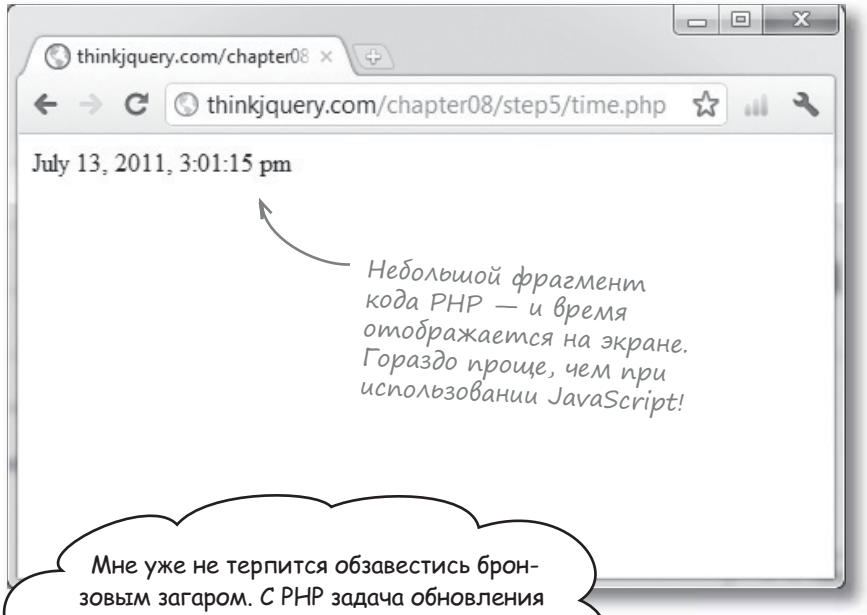
Задание!

Создайте новый файл в той же папке, в которой находится файл *index.html*, присвойте ему имя *time.php*. Включите в файл *time.php* следующий фрагмент:



ТЕСТ-ДРАЙВ

Сохраните файл *time.php*, откройте его в браузере и убедитесь в правильности формата даты. Код PHP должен выполняться через веб-сервер, поэтому URL-адрес должен начинаться с префикса *http://*, а не *file://*. Также убедитесь в том, что URL-адрес указывает на сервер, на котором вы занимаетесь разработкой кода.



Да, отправляться на Гавайи еще рано. Еще нужно кое-что сделать.

1. **Вывести на странице информацию о том, когда произошло последнее обновление.**
2. **Вывести частоту обновлений.**
3. **Предоставить пользователям возможность прекращать и снова запускать обновление на их усмотрение.**

Начнем с первого и второго пунктов списка. Мы будем рассматривать их вместе, потому что они связаны друг с другом.

Возьми в руку карандаш

Добавьте тег `<span>` с идентификатором `freq` в нижний блок страницы *index.html*. Он нужен для вывода результата новой функции `showFrequency`, сообщающей частоту обновления страницы. Также создайте другую функцию с именем `getTimeAjax`, которая загружает файл *time.php* с использованием метода `load` — вспомогательного метода jQuery для работы с Ajax. Этот метод получает в параметре URL-адрес и автоматически выводит результат в области `div` с идентификатором `updatedTime`. Наконец, замените вызов функции `getTime` в `getXMLRacers` вызовом новой функции `getTimeAjax`.

```
<footer>
  <h4>Congratulations to all our finishers!</h4>
  ..................................
  <br><br>
  Last Updated: <div id="updatedTime"></div>
</footer>
<script src="scripts/jquery-1.6.2.min.js"></script>
<script src="scripts/my_scripts.js"></script>
```

index.html

```
function ..........................
  ..................."Page refreshes every " + FREQ/1000 + " second(s).");
}

.
.
.
.

function ..................
  $(..................).load(..................);
}
```

my_scripts.js

Возьми в руку карандаш
Решение

Итак, мы добавили тег `<span>` с идентификатором `freq` в нижний блок страницы *index.html*; в нем будет отображаться частота обновления страницы. Также была создана новая функция `getTimeAjax`, которая загружает файл *time.php* с использованием вспомогательного метода Ajax `load`, а потом записывает результат в область `updatedTime`. Функция `getXMLRacers` также была изменена, теперь вместо функции JavaScript `getTime` в ней используется функция `getTimeAjax`.

```
<footer>
  <h4>Congratulations to all our finishers!</h4>
  <span id="freq"></span>
  <br><br>
  Last Updated: <div id="updatedTime"></div>
</footer>
<script src="scripts/jquery-1.6.2.min.js"></script>
<script src="scripts/my_scripts.js"></script>
```

Добавляем элемент для вывода частоты обновлений.

index.html

```
function showFrequency(){
    $("#freq").html("Page refreshes every " + FREQ/1000 + " second(s).");
}
. . .
function getTimeAjax(){
  $('#updatedTime').load("time.php");
}
```

Создаем две новые функции: для вывода частоты и для получения времени с сервера средствами Ajax.

Делим на 1000, чтобы преобразовать миллисекунды в секунды.

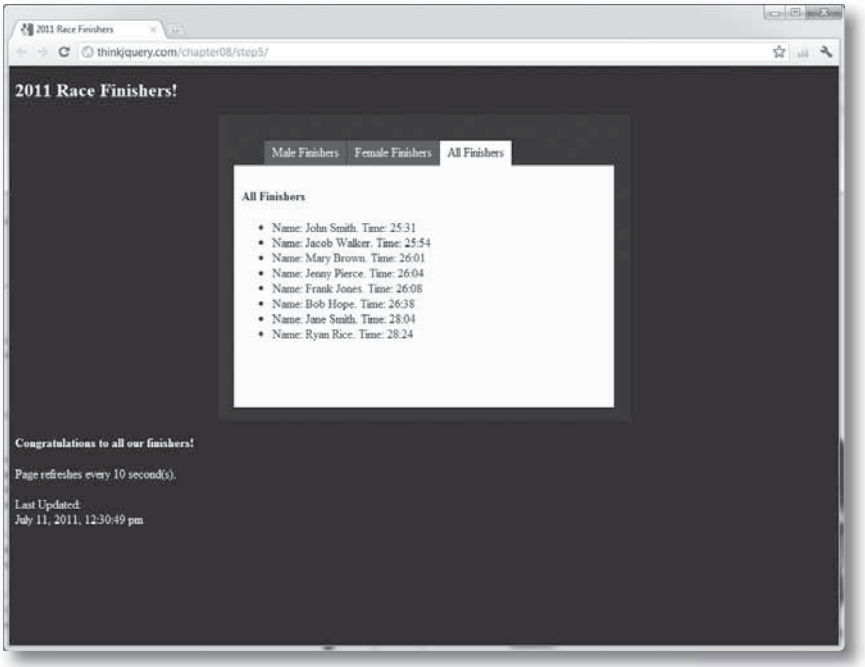
Загружаем файл time.php средствами Ajax.

Результат выводится на экран в элементе updatedTime.

my_scripts.js

ТЕСТ-ДРАЙВ

Включите только что написанный код в файл *my_scripts.js*. Также не забудьте включить новый элемент `span` в файл *index.html* и заменить вызов функции `getTime` на `getTimeAjax`.



1. ~~Вывести на странице информацию о том, когда произошло последнее обновление.~~

2. ~~Вывести частоту обновлений.~~

3. **Предоставить пользователям возможность прекращать и снова запускать обновление на их усмотрение.**

Но как остановить работу функции, которая вызывает сама себя?

Это не так просто.

Нужно изменить код функции так, чтобы она выполнялась только при соблюдении некоторых условий.

МОЗГОВОЙ ШТУРМ

Какие конструкции из рассмотренных нами ранее проверяли выполнение условий?

Отключение планирования событий на странице

В главах 5 и 7 мы создали функцию, которая использовала `setTimeout` для постоянного вызова функций, создававших эффект молнии. Это привело к неожиданным последствиям — при потере фокуса страницей и последующем возвращении визуальные эффекты начинали громоздиться друг на друга.

А так как мы уже определили, что в нашей ситуации нужно дождаться завершения предыдущего вызова, мы не можем переключиться на использование `setInterval` для этих вызовов.

Нужно найти другое решение. Может, что-нибудь из того, что мы уже видели? Мы не можем использовать события браузера `window.onblur` и `window.onfocus` — не заставлять же пользователя покидать страницу для обновления. Но уже в нескольких главах нам встречались примеры выполнения кода в *условных конструкциях*; давайте воспользуемся этой возможностью и на этот раз.

МОЗГОВОЙ ШТУРМ

Как вы думаете, какую условную конструкцию следует применить в данной ситуации? (Подсказка: мы уже использовали ее для проверки пола участников в файле XML.)

Часто Задаваемые Вопросы

В: А что еще, кроме XML, можно загружать в странице средствами Ajax?

О: При использовании jQuery на странице можно загружать много разных видов информации. Как вы уже видели, метод `load` позволяет загрузить результаты файла PHP прямо в элемент HTML. Также возможна загрузка других файлов HTML, файлов JavaScript, простого текста и объектов JSON (JavaScript Object Notation). Объекты JSON рассматриваются в следующей главе.

В: Какие еще вспомогательные методы для работы с Ajax существуют в jQuery?

О: В jQuery существует пять вспомогательных методов для работы с Ajax: `get`, `getJSON`, `getScript`, `post` и `load`. Первые четыре вызываются через объект jQuery, но метод `load` может вызываться для любого элемента, который используется в качестве приемника для возвращаемых данных.

В: Когда следует использовать метод `load`, а когда `ajax`?

О: Метод `load` предназначен для загрузки конкретного блока данных в конкретное место, как в нашей функции `getTimeAjax`. Метод `ajax` намного сложнее, он имеет больше параметров и находит более широкое применение. Он может использоваться для загрузки другой информации, а также для передачи данных на сервер для обработки. Эти возможности будут представлены в следующей главе.

Возьми в руку карандаш

Создайте глобальную переменную `repeat` со значением по умолчанию `true`. Создайте функцию, которая изменяет переменную `repeat` по щелчку новой кнопки с идентификатором `btnStop`. Запишите в элемент `span` с идентификатором `freq` код HTML для вывода сообщения «Updates paused». Создайте кнопку `btnStart`, которая присваивает глобальной переменной `repeat` значение `true` и вызывает функции `startAJAXcalls` и `setTimeout`, когда переменная `repeat` принимает значение `true`. Разместите новые кнопки в нижнем блоке страницы.

```
$(document).ready(function(){

    ..........................
    var FREQ = 10000;
    function startAJAXcalls(){

        ..............
            setTimeout( function() {
                getXMLRacers();
                startAJAXcalls();
            },
            FREQ

        ...........
    );
.
.
    $("#btnStop").click(function(){

        ..................
        $("#freq").html(                    );
                                ..................
    });

    ....................function(){

        ..................
        startAJAXcalls();

        ..................
    });
```

my_scripts.js

```
<footer>
   <h4>Congratulations to all our finishers!</h4>
   ..........................................................
   ..........................................................
   <br>
   <span id="freq"></span> <br><br>
```

index.html

Возьми в руку карандаш Решение

Когда все сделано, в приложении появляется переменная `repeat`. От ее состояния зависит, будет ли функция вызывать сама себя для получения файла XML с обновленной информацией. Значение переменной меняется кнопками `btnStop` и `btnStart`, находящимися в нижнем блоке страницы. Эти кнопки задают текст элемента `span` с идентификатором `freq`, чтобы при включенном или отключенном обновлении выводились разные сообщения.

По умолчанию переменная равна true, поэтому при загрузке страницы обновление включено.

Проверяем переменную repeat.

При щелчке на кнопке btnStop переменной присваивается значение false.

Кнопка btnStart возвращает переменной значение true. Также необходимо вызвать функцию startAJAXcalls для того, чтобы файл снова загружался.

Новые кнопки размещаются в нижнем блоке страницы.

```
$(document).ready(function(){
    var repeat = true;
    var FREQ = 10000;
    function startAJAXcalls(){
        if(repeat) {
            setTimeout( function() {
                getXMLRacers();
                startAJAXcalls();
            },
            FREQ
        )
    });
.
.
    $("#btnStop").click(function(){
        repeat = false;
        $("#freq").html( "Updates paused." );
    });
    $("#btnStart").click( function(){
        repeat = true;
        startAJAXcalls();
        showFrequency();
    });
```

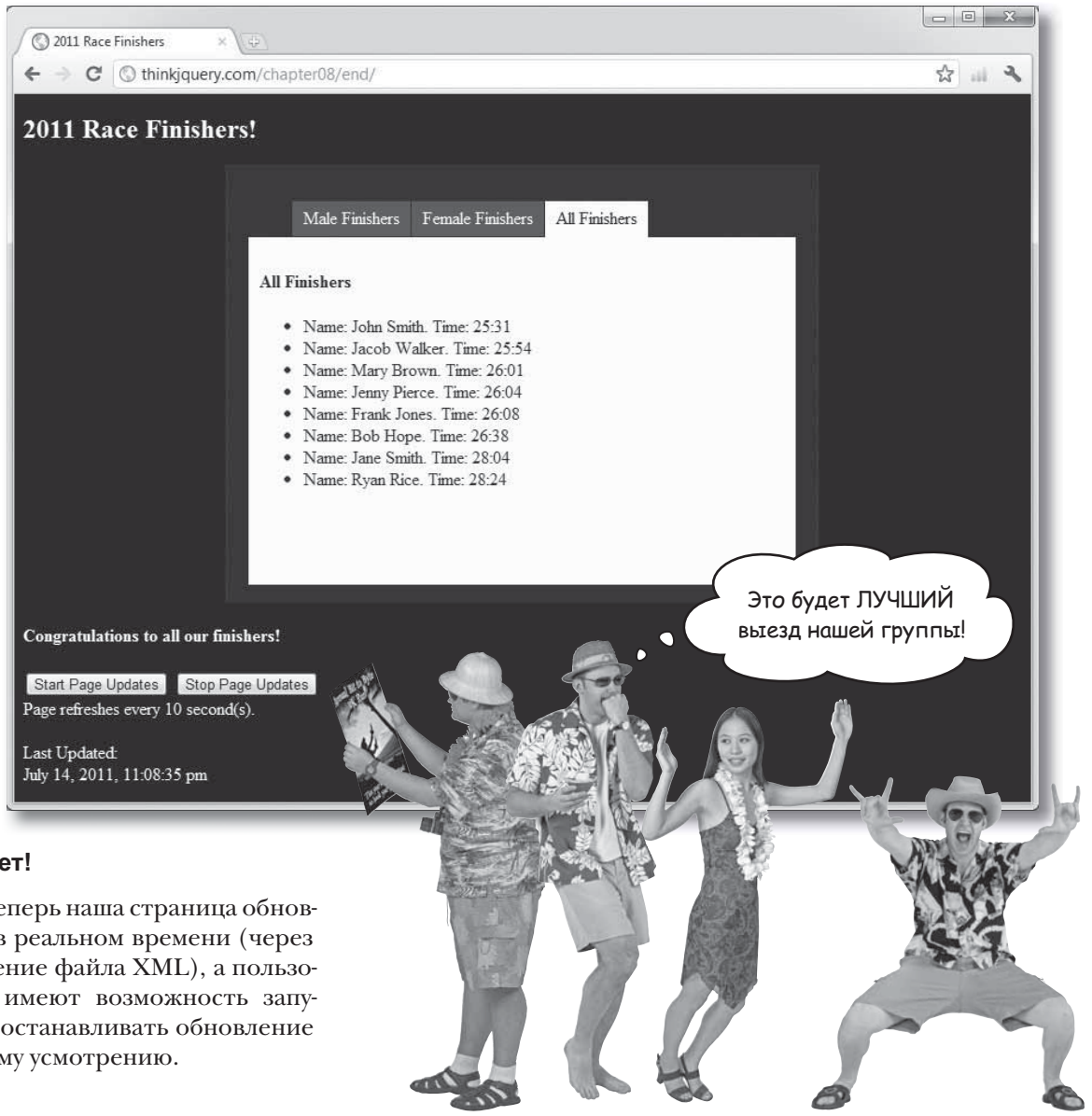
my_scripts.js

```
<footer>
    <h4>Congratulations to all our finishers!</h4>
    <button id="btnStart">Start Page Updates</button>
    <button id="btnStop">Stop Page Updates</button>
    <br>
    <span id="freq"></span> <br><br>
```

index.html

ТЕСТ-ДРАЙВ

Включите новый код в файлы *my_scripts.js* и *index.html*. Загрузите страницу в своем любимом браузере и убедитесь в том, что она по-прежнему работает. Проверьте, что новые кнопки действительно останавливают выдачу запросов Ajax (воспользуйтесь вкладкой «Network» в Google Chrome или вкладкой «Net» в Firebug для Firefox).



Работает!

Итак, теперь наша страница обновляется в реальном времени (через обновление файла XML), а пользователи имеют возможность запускать и останавливать обновление по своему усмотрению.

ГЛАВА 8

Ваш инструментарий jQuery/Ajax

Глава 8 осталась позади. Ваш творческий инструментарий расширился: в нем появились навыки работы с PHP, XML и Ajax.

Ajax

Комбинация технологий, позволяющая обновить часть веб-страницы без повторной загрузки всей страницы.

Использует обращения к служебному серверу, который обрабатывает данные до того, как вернуть их приложению.

В jQuery функциональность Ajax реализуется методом ajax.

XML

Формальный, но гибкий язык разметки для описания данных и их структур.

Часто используется для хранения информации или для форматирования данных с целью передачи.

Используется во многих распространенных веб-технологиях: RSS, SOAP/веб-службы, SVG и др.

Вспомогательные методы ajax()

В jQuery существует пять вспомогательных методов, упрощающих вызов ajax. Все они по умолчанию получают разные наборы параметров, но в конечном итоге вызывают метод ajax:

```
$.get
$.getJSON
$.getScript
$.post
$.load
```

PHP

Серверный язык сценариев; позволяет обработать содержимое веб-страницы на сервере до того, как передавать ее клиентскому браузеру.

9 Данные JSON

Клиент встречается с сервером

Возможность чтения данных из файлов XML безусловно полезна, но иногда этого оказывается недостаточно. Другой, более эффективный формат передачи данных JSON (JavaScript Object Notation) упрощает получение данных со стороны сервера. Кроме того, данные JSON проще генерируются и читаются, чем данные XML. При помощи jQuery, PHP и SQL можно создать базу данных для хранения информации, которая позднее читается с использованием JSON и отображается на экране средствами jQuery. Вот она, истинная мощь веб-приложений!

В отделе маркетинга MegaCorp никто не знает XML

От: **Отдел маркетинга MegaCorp**
Тема: **Re: страница результатов 42-го ежегодного марафона**

Привет веб-программистам!

Нам очень понравились изменения, внесенные вами на сайте.

Однако мы столкнулись с проблемой: в нашем офисе никто не знает XML! И мы не понимаем, как включить данные новых участников в файл для сайта.

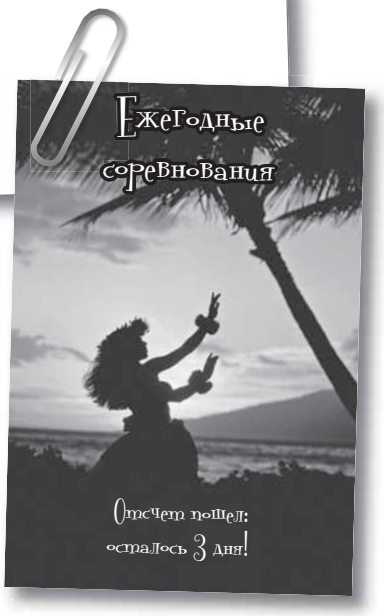
Мы пытались, но при малейшей неточности на сайте начинает твориться что-то странное... Участники не отображаются; поля исчезают со страницы, хотя они и присутствуют в файле XML. Все это очень странно.

Нам хотелось бы, чтобы для ввода нового участника было достаточно заполнить несколько полей и нажать кнопку. Вы сможете это сделать?

И так, чтобы возможные ошибки не нарушали нормальной работы сайта...

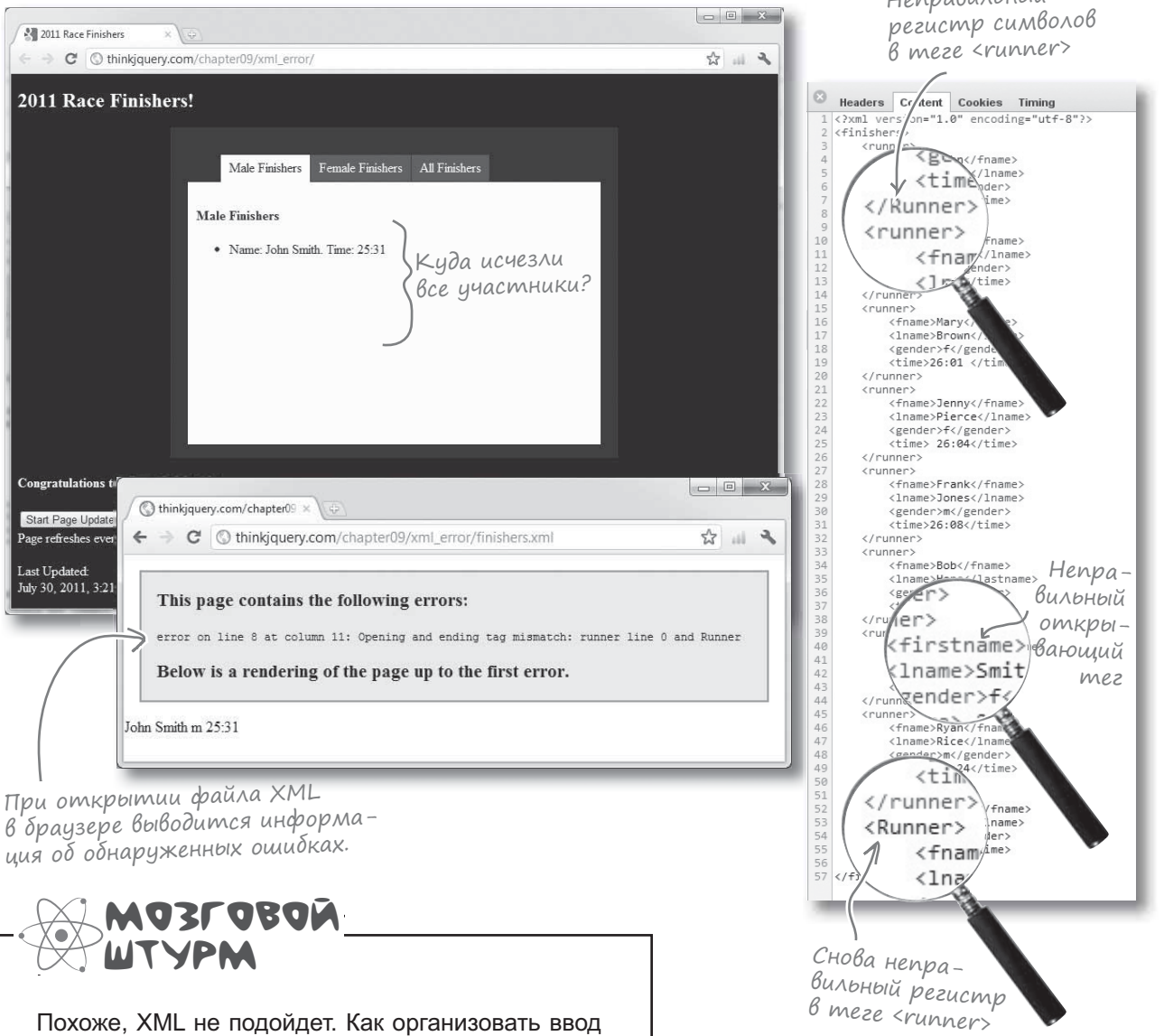
Знаю, что до вылета на Гавайи осталось всего три дня, но нам хотелось бы, чтобы сайт заработал до нашего отъезда. Как вы думаете, мы успеем?

--
Диона Хаузни
Руководитель отдела маркетинга
Фирма MegaCorp



Ошибки в XML

Если файл XML содержит некорректные данные, то чтение и разбор XML перестают работать. Такие ошибки возникают из-за проблем с тегами (если вы забыли закрыть тег или использовали неправильный регистр символа в имени тега). Данные тоже могут создать проблемы, если они не были должным образом закодированы для использования в XML.



МОЗГОВОЙ ШТУРМ

Похоже, XML не подойдет. Как организовать ввод информации об участниках в отделе маркетинга?

Ввод данных на веб-странице

Скорее всего, вы уже подумали об использовании формы HTML. На форме можно ввести любые данные и передать их серверу для обработки, а используемые на них разнообразные элементы позволяют вводить самые разные виды данных. Формы будут более подробно описаны в главе 10, а пока мы воспользуемся всего двумя простейшими элементами форм: текстовым полем и раскрывающимся списком. Возможно, вы уже профессионально разбираетесь в формах, и все же давайте в общих чертах посмотрим, с чем мы имеем дело.

Тег <input> сообщает форме, что этот элемент будет поставлять информацию.

Атрибут type указывает браузеру, как следует отображать этот элемент.

Закрывающий тег элемента

```
<input type="text" name="txtEmail" />
```

Имя элемента передается серверу для обработки.

Тег <select> сообщает браузеру, что нужно вывести раскрывающийся список.

```
<select name="truthiness">
    <option value="1">True</option>
    <option value="0">False</option>
</select>
```

Значение value выбранного пункта списка отправляется серверу.

Элемент option определяет пункт раскрывающегося списка.

Наверное, проще всего создать новую вкладку с формой для ввода данных?

Да, такое решение будет работать и легко реализуется, так как мы уже умеем добавлять новые вкладки.

А после этого можно будет заняться сохранением/загрузкой этих данных для отображения в списках участников.

Готово к употреблению

Включите в файл *index.html* **новую вкладку для ввода информации об участниках на форме. Также внесите изменения в файл** *my_style.css*, чтобы увеличить ширину элемента с идентификатором `main`.

my_style.css

```
#main {
  background:#181818;
  color:#111;
  padding:15px 20px;
  width:600px;
  border:1px solid #222;
  margin:8px auto;
}
```

```
<ul class="idTabs">
    <li><a href="#male">Male Finishers</a></li>
    <li><a href="#female">Female Finishers</a></li>
    <li><a href="#all">All Finishers</a></li>
    <li><a href="#new">Add New Finisher</a></li>
</ul>
<div id="male">
    <h4>Male Finishers</h4><ul id="finishers_m"></ul>
</div>
<div id="female">
    <h4>Female Finishers</h4><ul id="finishers_f"></ul>
</div>
<div id="all">
    <h4>All Finishers</h4>  <ul id="finishers_all"></ul>
</div>
<div id="new">
    <h4>Add New Finisher</h4>
    <form id="addRunner" name="addRunner" action="service.php" method="POST">
        First Name: <input type="text" name="txtFirstName" id="txtFirstName" /> <br>
        Last Name: <input type="text" name="txtLastName" id="txtLastName" /> <br>
        Gender: <select id="ddlGender" name="ddlGender">
            <option value="">--Please Select--</option>
            <option value="f">Female</option>
            <option value="m">Male</option>
        </select><br>
        Finish Time:
        <input type="text" name="txtMinutes" id="txtMinutes" size="10" maxlength="2" />(Minutes)
        <input type="text" name="txtSeconds" id="txtSeconds" size="10" maxlength="2" />(Seconds)
        <br><br>
        <button type="submit" name="btnSave" id="btnSave">Add Runner</button>
        <input type="hidden" name="action" value="addRunner" id="action">
    </form>
</div>
```

Добавляем новую вкладку.

Добавляем новую форму HTML для сбора данных и их передачи серверу.

Атрибут action сообщает форме, куда следует передать данные для обработки.

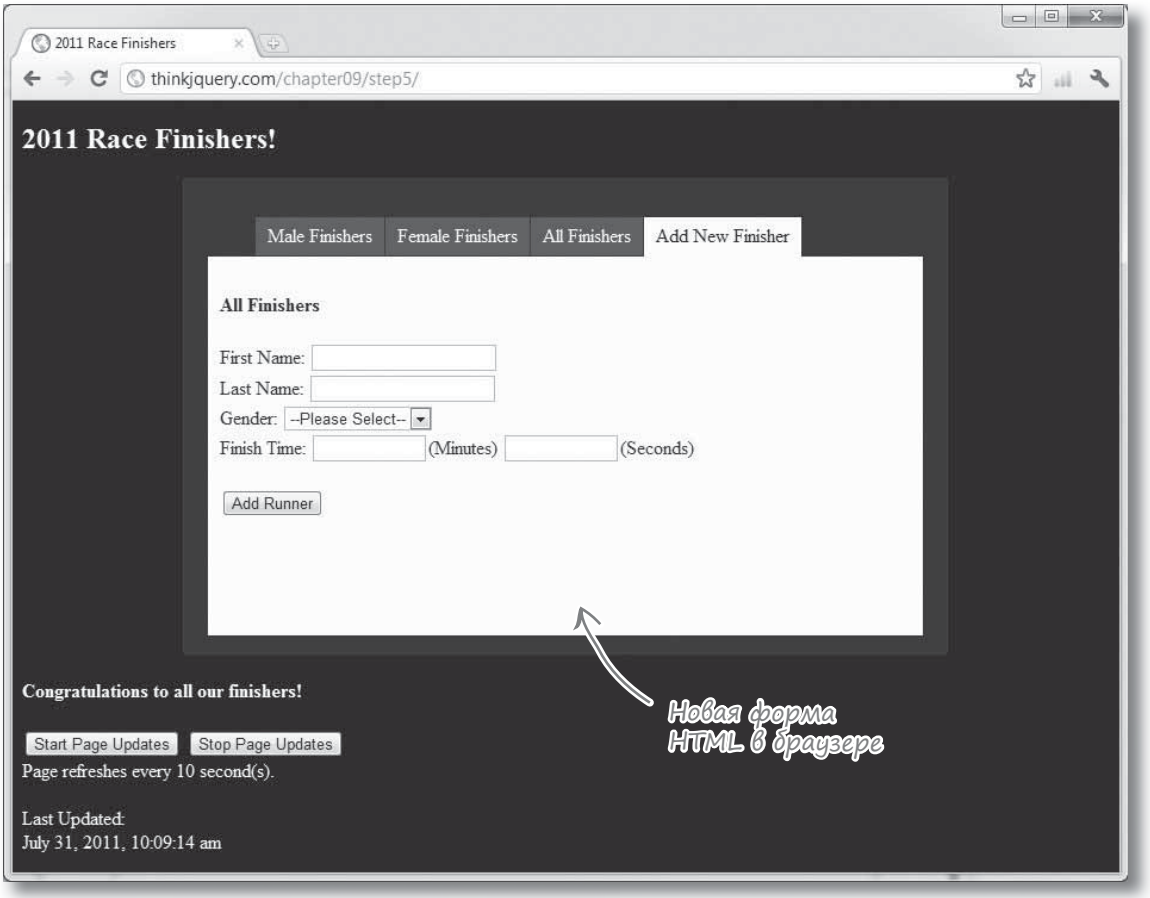
Атрибут method определяет способ передачи данных серверу.

Скрытое поле HTML. Вскоре мы расскажем о том, как используется это поле.

index.html

ТЕСТ-ДРАЙВ

Откройте файл *index.html* в браузере. Перейдите на вкладку **Add New Finisher**, присмотритесь к новой форме и полям, добавленным на страницу.



МОЗГОВОЙ ШТУРМ

Итак, теперь у нас есть форма для ввода данных. Как, по вашему мнению, следует организовать хранение и выборку данных?

Что делать с данными

Теперь необходимо отправить серверу данные, введенные на форме, и как-то их сохранить. Для сохранения информации в базе данных мы воспользуемся другим языком, PHP. Не волнуйтесь, скоро мы поговорим о PHP и базах данных, а пока сосредоточимся на том, как передать данные от формы серверу.

Существуют два способа отправки данных серверу с использованием HTTP: GET и POST (они также называются «методами»). Главное различие между GET и POST — конкретный механизм передачи данных. GET присоединяет имена полей формы и их значения в конец URL-адреса в формате пар «ключ/значение». PHP читает эту информацию из ассоциативного массива `$_GET[]`, который передается серверу при отправке данных формы. Передаваемые данные следуют после знака ? в URL-адресе.

Метод POST тоже передает данные в ассоциативном массиве, но они кодируются другим способом, и эти данные не видны конечному пользователю в URL-адресе. В ассоциативном массиве `$_POST[]` **хранится вся информация из элементов формы**. Таким образом, как и массив `$_GET[]`, он состоит из пар «ключ/значение» элементов формы.

Метод HTTP GET

```
<form id="my_form" method="get" action="x.php">
  <input type="text" name="a" value="1" />
  <input type="text" name="b" value="2" />
</form>
```

Отправка данных серверу.

x.php?a=1&b=2

```
<?php
    echo $_GET["a"] ; // Writes out "1"
    echo $_GET["b"] ; // Writes out "2"
?>
```

Метод HTTP POST

```
<form id="my_form" method="post" action="x.php">
  <input type="text" name="a" value="1" />
  <input type="hidden" name="c" value="3" />
</form>
```

Отправка данных серверу.

x.php

```
<?php
    echo $_POST["a"] ; // Writes out "1"
    echo $_POST["c"] ; // Writes out "3"
?>
```

В этой главе мы будем использовать метод POST.

Но если форма отправляет информацию сама, для чего jQuery и все остальное?

Да, форма может отправить информацию...

Но как упоминалось в предыдущей главе при описании преимуществ jQuery и Ajax, отправка или получение данных не требует повторной загрузки всей страницы. Но прежде чем отправлять данные серверу средствами jQuery и Ajax, необходимо подготовить их к отправке.

Форматирование данных перед отправкой

Прежде чем отправлять информацию серверу (средствами Ajax), необходимо сначала подготовить ее — преобразовать в формат, который может передаваться вызовами Ajax и который будет понятен серверу. Для этого данные сериализуются в один объект, чтобы вызов Ajax мог передать их как единое целое. В jQuery существуют два вспомогательных метода сериализации данных: `serialize` и `serializeArray`. Первый объединяет весь ввод на форме в одну строку пар «ключ/значение», разделенных амперсандами (**&**). Второй создает ассоциативный массив пар «ключ/значение»; массив также является одним объектом, но он намного лучше структурирован, чем результат `serialize`. Мы рассмотрим обе возможности, но при работе с данными марафона будет использоваться метод `serializeArray`.

serialize

```
<form id="my_form">
  <input type="text" name="a" value="1" />
  <input type="text" name="b" value="2" />
  <input type="hidden" name="c" value="3" />
</form>
```

```
$("#my_form").serialize();
```

Селектор идентификатора формы

Метод serialize

Результат

```
a=1&b=2&c=3
```

serializeArray

```
<form id="my_form">
  <input type="text" name="a" value="1" />
  <input type="hidden" name="c" value="3" />
</form>
```

```
$("#my_form:input").serializeArray();
```

Селектор идентификатора формы, за которым следует фильтр элемента HTML input. Селектор должен найти только элементы HTML с типом «input».

Вызов метода serializeArray

Результат

```
[
  {
    name: "a",
    value: "1"
  },
  {
    name: "c",
    value: "3"
  }
]
```

Отправка данных серверу

jQuery предоставляет вспомогательный метод `post`, предназначенный для отправки данных серверу. Метод `post` получает несколько параметров (в числе которых URL-адрес, по которому отправляется информация), передаваемую информацию и функцию-обработчик, которая выполняется после завершения отправки POST.

```
$.post(url_to_send, data, function(json){
});
```

Вспомогательный метод jQuery

URL-адрес, на который отправляются данные

Передаваемые данные (уже прошедшие сериализацию)

Выполняется эта функция обратного вызова.

Данные возвращаются в объекте с именем json. Не обращайте на него внимания, мы займемся им позднее в этой главе.

Развлечения с магнитами

Создайте слушателя события click для `#btnSave`, который получает данные формы и сериализует их. Информация отправляется серверу методом jQuery `post`. URL-адрес для отправки берется из атрибута `action` формы. Создайте функцию `clearInputs`, стирающую все данные из полей формы, если отправка была **успешной**. Необходимо отменить отправку данных формы по умолчанию (возвращая `false`), для чего следует использовать слушателя `.submit` формы с идентификатором `addRunner`.

```
$('_____________').click(function() {
   var data = $("#addRunner :input")._____________();
   $.post($("#addRunner").attr('action'), _________ , _____________(json){
      if (json.status == "fail") {
         alert(json._____________);
      }
      if (json.status == _____________) {
         alert(json.message);
         clearInputs();
      }
   }, "json");
});
function _____________{
   $("#addRunner :input").each(function(){
      $(this).val('');
   });
}
$("#addRunner")._____________(function(){
   return false;
});
```

serializeArray

"success"

message

data

#btnSave

function

submit

clearInputs()

my_scripts.js

Развлечения с магнитами. Решение

Создайте слушателя события click для `#btnSave`, который получает данные формы и сериализует их. Информация отправляется серверу методом jQuery `post`. URL-адрес для отправки берется из атрибута `action` формы. Создайте функцию `clearInputs`, стирающую все данные из полей формы, если отправка была **успешной**. Необходимо отменить отправку данных формы по умолчанию (возвращая `false`), для чего следует использовать слушателя `.submit` формы с идентификатором `addRunner`.

```
$('#btnSave').click(function() {
   var data = $("#addRunner :input").serializeArray();
   $.post($("#addRunner").attr('action'), data , function(json){
      if (json.status == "fail") {
         alert(json.message);
      }
      if (json.status == "success") {
         alert(json.message);
         clearInputs();
      }
   }, "json");
});
function clearInputs(){
   $("#addRunner :input").each(function(){
      $(this).val('');
   });
}
$("#addRunner").submit(function(){
   return false;
});
```

Готовим данные всех полей формы к отправке на сервер.

Получение атрибута action формы, которая должна отправить данные.

Проверка значения, возвращенного сервером (и заданного в коде PHP), позволяет узнать, успешно ли прошла отправка POST.

Используем фильтр элементов HTML для перебора всех полей input формы и стирания их содержимого.

Отменяем отправку данных формой по умолчанию, чтобы выполнить ее кодом jQuery в событии click кнопки.

my_scripts.js

ТЕСТ-ДРАЙВ

После внесения последних изменений внешний вид страницы не изменится. Тем не менее включите в файл *my_scripts.js* созданный код. Откройте страницу *index.html* в браузере, перейдите на вкладку «Network» (Chrome) или «Net» (Firebug); при каждом нажатии кнопки `btnSubmit` должна происходить отправка POST файлу *service.php*. Метод **POST** указывается в разделе **Request Method** вкладки **Headers**. Здесь же приводятся данные формы (**Form Data**). Осталось только найти место для их сохранения...

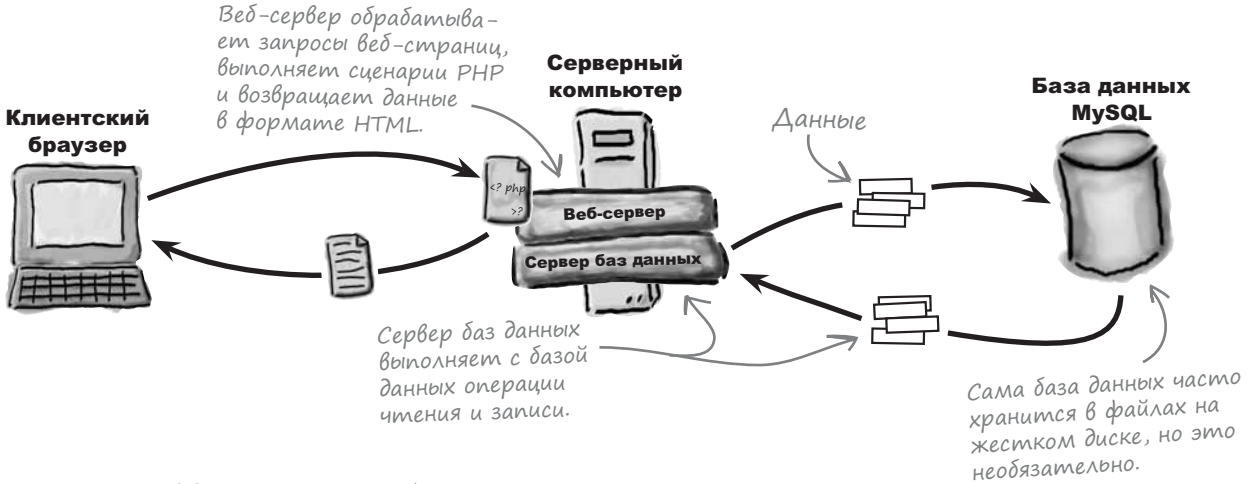
Хранение информации в базе данных MySQL

SQL — язык структурированных запросов (Structured Query Language).

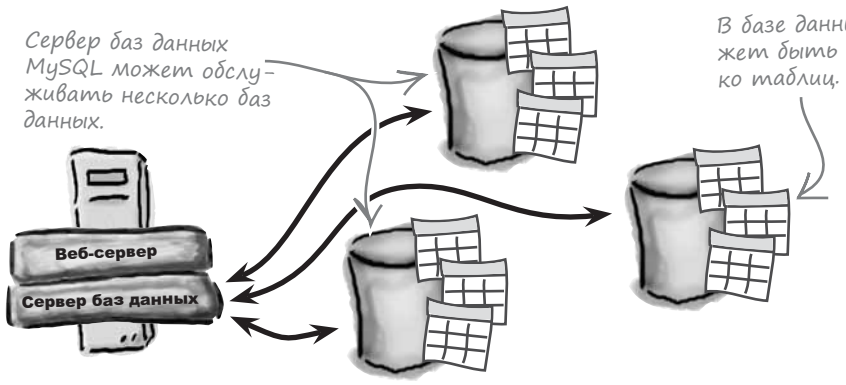
Реляционные системы управления базами данных (РСУБД) представляют собой высокоуровневые приложения, предназначенные для хранения и организации информации об отношениях между различными видами данных.

Существует великое множество разнообразных РСУБД (также часто называемых *серверами баз данных*) *разного назначения и размера* (*и разной стоимости*). *Для наших целей будет использоваться бесплатный сервер баз данных* **MySQL**. Для взаимодействия с сервером баз данных используется язык, понятный серверу, — в нашем случае это язык **SQL**. Сервер баз данных обычно работает в сочетании с веб-сервером (иногда на одном компьютере) над чтением и записью данных, а также выдачей готовых веб-страниц.

В MySQL данные хранятся в таблицах базы данных.



Базы данных MySQL делятся на *таблицы*, содержимое которых состоит из строк и столбцов. Большинство веб-приложений работает с одной или несколькими таблицами одной базы данных, их можно сравнить с книгами, стоящими на одной полке.



SQL — язык запросов, используемый для взаимодействия с базой данных MySQL.

Создание базы данных для информации об участниках

Стоп! А вы уже установили и настроили MySQL и PHP? Прежде чем продолжать, обязательно выполните инструкции по установке и настройке PHP и MySQL из приложения II.

Хорошо, проезжайте. Теперь вы сможете выполнить задания этой главы.

Готово к употреблению

Мы уже написали код создания базы данных, таблиц и пользователей за вас. Запустите MySQL Workbench, откройте новое подключение и выполните следующий код SQL:

```
create database hfjq_race_info;

CREATE USER 'runner_db_user'@'localhost' IDENTIFIED BY 'runner_db_password';
GRANT SELECT,INSERT,UPDATE,DELETE ON hfjq_race_info.* TO 'runner_db_user'@'localhost';

use hfjq_race_info;

CREATE TABLE runners(
      runner_id INT not null AUTO_INCREMENT,
      first_name VARCHAR(100) not null,
      last_name VARCHAR(100) not null,
      gender VARCHAR(1) not null,
      finish_time VARCHAR(10),
      PRIMARY KEY (runner_id)
   );
```

Создаем базу данных с именем hfjq_race_info.

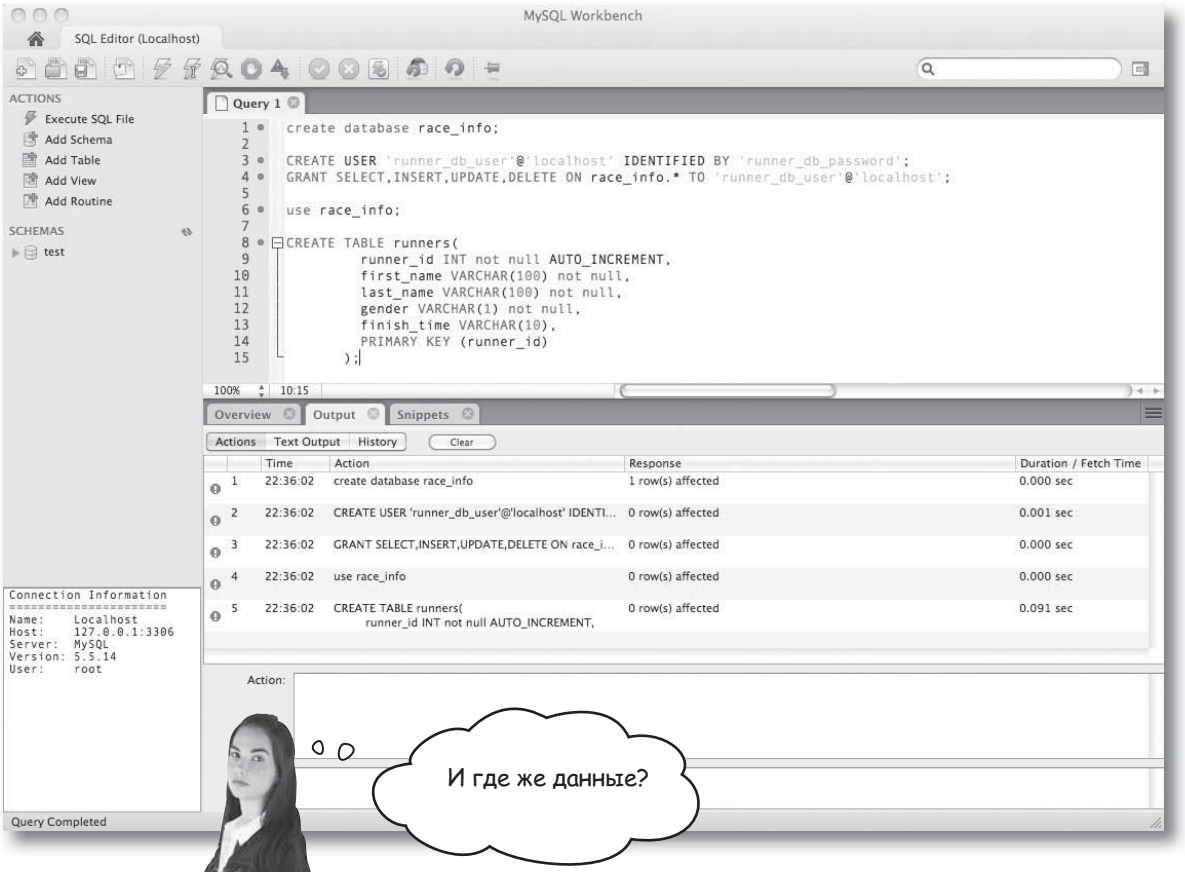
Создаем пользователя runner_db_user, задаем ему пароль и разрешаем этому пользователю получать, добавлять, обновлять и удалять данные из базы.

Указываем сценарию, что следующий фрагмент относится к нашей новой базе данных.

Создаем таблицу runners, в которой хранится вся необходимая информация об участниках марафона.

ТЕСТ-ДРАЙВ

Запустите MySQL Workbench и откройте подключение к серверу. Вставьте код SQL на панель Query и щелкните на значке с молнией, чтобы его выполнить. На панели Output в нижней части окна должны появиться сообщения об успешном выполнении операций.



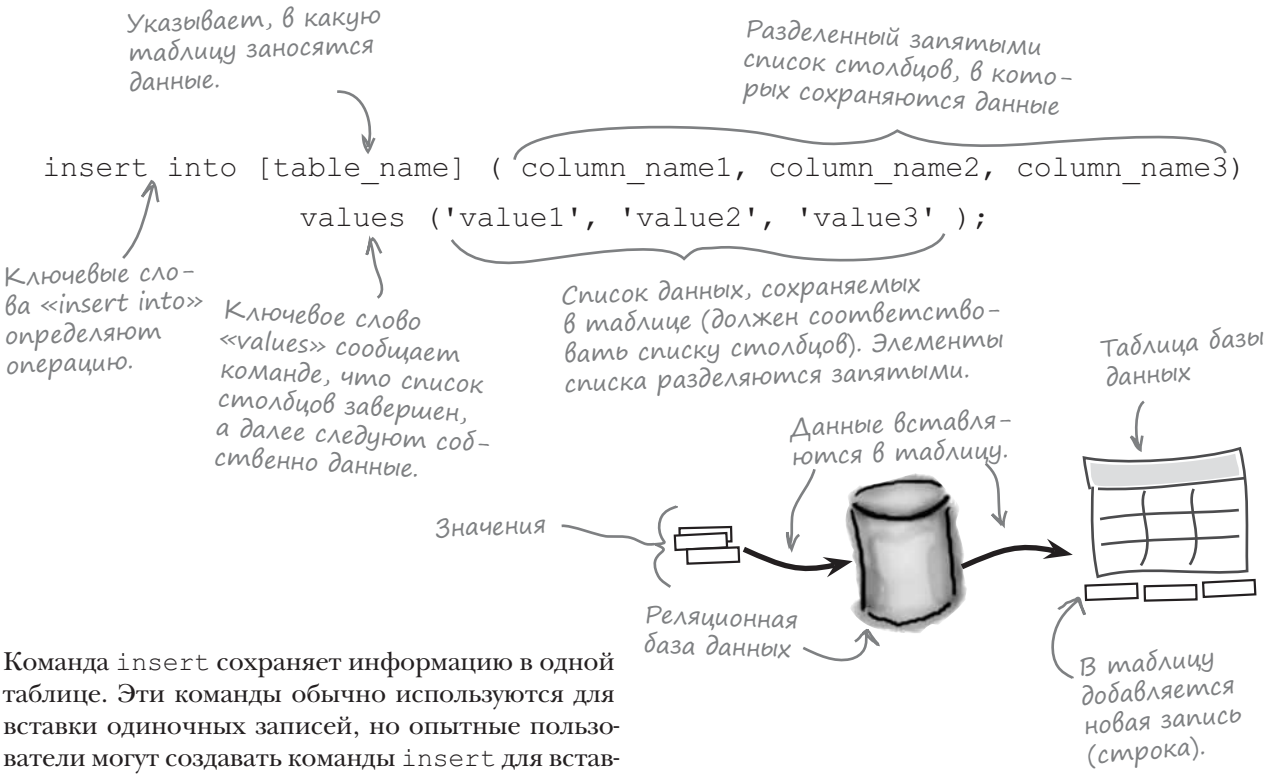
Сейчас мы ими займемся.

Только что выполненный нами код SQL только создает базу данных, создает пользователя, предоставляет новому пользователю доступ к базе данных и создает таблицу для хранения информации. Теперь давайте посмотрим, как сохранить в таблице данные.

Строение команды insert

Основные операции с базой данных — сохранение информации, ее изменение/обновление и последующее извлечение. Вскоре мы займемся извлечением (выборкой) данных, а пока нас прежде всего интересует то, как ***занести информацию*** в таблицы базы данных.

Включение данных в таблицы осуществляется командой `insert`.



Команда `insert` сохраняет информацию в одной таблице. Эти команды обычно используются для вставки одиночных записей, но опытные пользователи могут создавать команды `insert` для вставки сразу нескольких записей. Впрочем, в наших примерах будет использоваться только вставка одиночных записей.

В команде рекомендуется перечислить столбцы в порядке вставки данных, хотя это и необязательно. Если столбцы не указаны, со вставкой данных могут возникнуть проблемы, потому что первое значение автоматически помещается в первый столбец, второе — во второй столбец и т. д. Чтобы пользоваться сокращенной записью, необходимо хорошо знать структуру таблицы данных.

Будьте осторожны!

Порядок следования имен столбцов и значений важен!

Порядок перечисления значений должен соответствовать порядку столбцов. По нему база данных определяет, в какой столбец следует поместить то или иное значение.

Упражнение

Напишите команды SQL `insert` для вставки данных, уже хранящихся в файле XML. Каждая команда должна вставлять одну запись в созданную нами таблицу. Первую команду мы написали за вас.

```
insert into runners (first_name, last_name, gender, finish_time)
   values ('John','Smith','m','25:31') ;
```

Упражнение
Решение

Весь код SQL, необходимый для вставки информации об участниках в таблицы базы данных, написан. Запустите MySQL Workbench и выполните код.

```
insert into runners (first_name, last_name, gender, finish_time)
   values ('John','Smith','m','25:31') ;

insert into runners (first_name, last_name, gender, finish_time)
   values ('Jacob','Walker','m','25:54') ;

insert into runners (first_name, last_name, gender, finish_time)
   values ('Mary','Brown','f','26:01') ;

insert into runners (first_name, last_name, gender, finish_time)
   values ('Jenny','Pierce','f','26:04') ;

insert into runners (first_name, last_name, gender, finish_time)
   values ('Frank','Jones','m','26:08') ;

insert into runners (first_name, last_name, gender, finish_time)
   values ('Bob','Hope','m','26:38') ;

insert into runners (first_name, last_name, gender, finish_time)
   values ('Jane','Smith','f','28:04') ;

insert into runners (first_name, last_name, gender, finish_time)
   values ('Ryan','Rice','m','28:24') ;

insert into runners (first_name, last_name, gender, finish_time)
   values ('Justin','Jones','m','29:14') ;
```

Хорошо, информация об участниках сохранена в базе данных. Как теперь извлечь ее для веб-приложения?

Пора использовать новый язык: PHP.

Не беспокойтесь! Мы ограничимся только тем, что необходимо для взаимодействия на стороне сервера (включая общение с сервером баз данных).

Использование PHP для работы с данными

PHP — язык программирования, и для его использования необходима соответствующая среда: веб-сервер с поддержкой PHP. Сценарии PHP и веб-страницы, использующие сценарии, **должны размещаться на веб-сервере** (вместо простого запуска сценария из локальной файловой системы).

Если у вас имеется локально установленный веб-сервер с поддержкой PHP, вы сможете протестировать сценарии PHP прямо на локальном компьютере.

Браузеры ничего не знают о PHP, а следовательно, не могут выполнять сценарии PHP.

В отличие от страниц HTML, которые можно открыть в браузере локально, сценарии PHP всегда должны «открываться» по URL-адресу из веб-сервера.

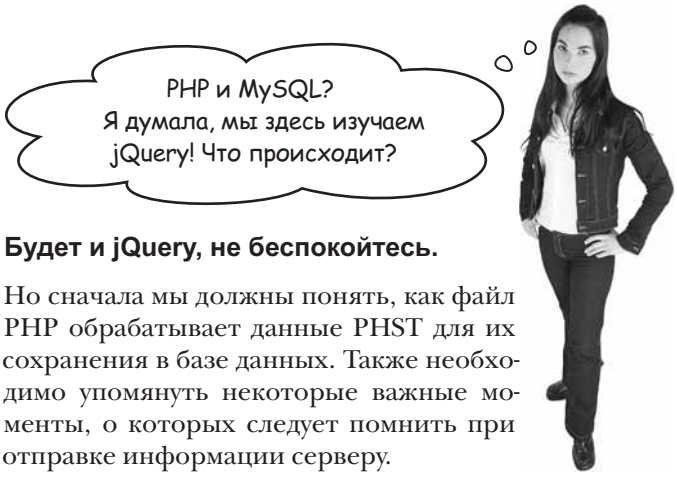
Для браузера этот сценарий PHP представляет собой бессмысленный набор символов.

Веб-сервер «понимает» код PHP и выполняет сценарий!

Веб-серверы с поддержкой PHP умеют выполнять сценарии PHP и преобразовывать их в веб-страницы HTML, понятные для браузера.

Как быстро определить, была ли веб-страница предоставлена веб-сервером? URL-адреса таких страниц начинаются с префикса «http:». Веб-страницы, открываемые как локальные файлы, всегда начинаются с префикса «file:».

Сценарии PHP должны выполняться на веб-сервере, иначе они работать не будут.



Будет и jQuery, не беспокойтесь.

Но сначала мы должны понять, как файл PHP обрабатывает данные PHST для их сохранения в базе данных. Также необходимо упомянуть некоторые важные моменты, о которых следует помнить при отправке информации серверу.

Обработка данных POST на сервере

Мы уже рассмотрели специальный объект `$_POST`, создаваемый для передачи информации веб-серверу от формы в браузере. Он представляет собой ассоциативный массив со всей отправленной информацией, при этом в качестве ключа используется имя (а не идентификатор!) элемента HTML, а содержимое элемента HTML сохраняется как значение ассоциативного массива. Код PHP на сервере читает объект `$_POST` и определяет, какая информация была отправлена серверу.

Информация извлекается *из* массива по ключу, с которым она была отправлена (имя элемента HTML). Так значения становятся доступными в сценарии PHP.

Значение выводится на экран.

Имя элемента HTML, в котором вводились данные на форме.

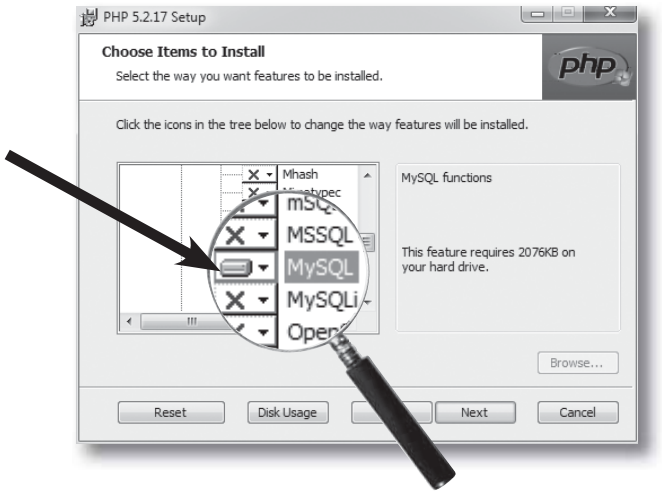
```
echo $_POST["txtFirstName"];
```

Имя массива, автоматически создаваемого для данных, отправляемых файлу PHP методом POST.

Мы почти добрались до той стадии, когда мы можем извлечь информацию из базы данных и понять, как вывести ее в списках участников. Но сначала нам придется написать еще немного кода PHP для подключения к базе данных...

Подключение к базе данных из кода PHP

Помните, как ближе к концу процесса установки PHP мы выбирали библиотеку?



Эта библиотека позволяет PHP взаимодействовать с базой данных MySQL. Мы используем ее для подключения к созданной ранее базе данных, чтобы прочитать из нее информацию об участниках.

Откройте теги PHP.

mysql_connect — функция библиотеки PHP, включенной в ходе установки PHP.

Имя сервера, на котором находится база данных MySQL

Пользователь MySQL, с правами которого вы подключаетесь к базе данных

Пароль MySQL для пользователя

```
<?php

    mysql_connect('127.0.0.1', 'runner_db_user', 'runner_db_password')
      OR die( 'Could not connect to database.');
    mysql_select_db('hfjq_race_info');

    echo "Connected!";

?>
```

Команда die выводит сообщение и завершает сценарий PHP.

mysql_select_db сообщает PHP, с какой базой данных мы будем работать.

Закрывающий тег PHP

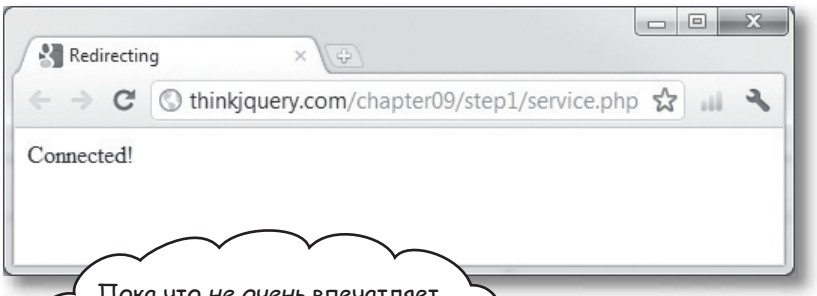
Если подключение к базе данных прошло успешно, сообщение выводится на экран, а если нет, выполнение сценария не достигнет этой точки.

service.php

ТЕСТ-ДРАЙВ

Откройте свой любимый текстовый редактор и добавьте код с предыдущей страницы. Сохраните файл под именем *service.php* в одном каталоге с файлом *index.html* этой главы. Откройте файл *service.php* в браузере и просмотрите результаты запроса к базе данных.

Не забудьте, что код PHP ***должен*** выполняться через веб-сервер, поэтому ваш URL-адрес должен начинаться с префикса *http://*, а не *file://*.



Пока что *не очень* впечатляет. Ну да, мы подключились... но я по-прежнему не вижу данных!

Вы правы.

Для чтения данных, как и для их вставки, существует специальная команда. Давайте посмотрим, как она работает.

Часто Задаваемые Вопросы

В: MySQL Workbench — единственный способ управления базой данных MySQL?

О: Нет! Существуют другие способы и другие инструменты. PHPMyAdmin — стандартный инструмент на базе веб-технологий для управления базами данных MySQL. Также базой данных можно управлять из окна терминала, вводя данные в командной строке.

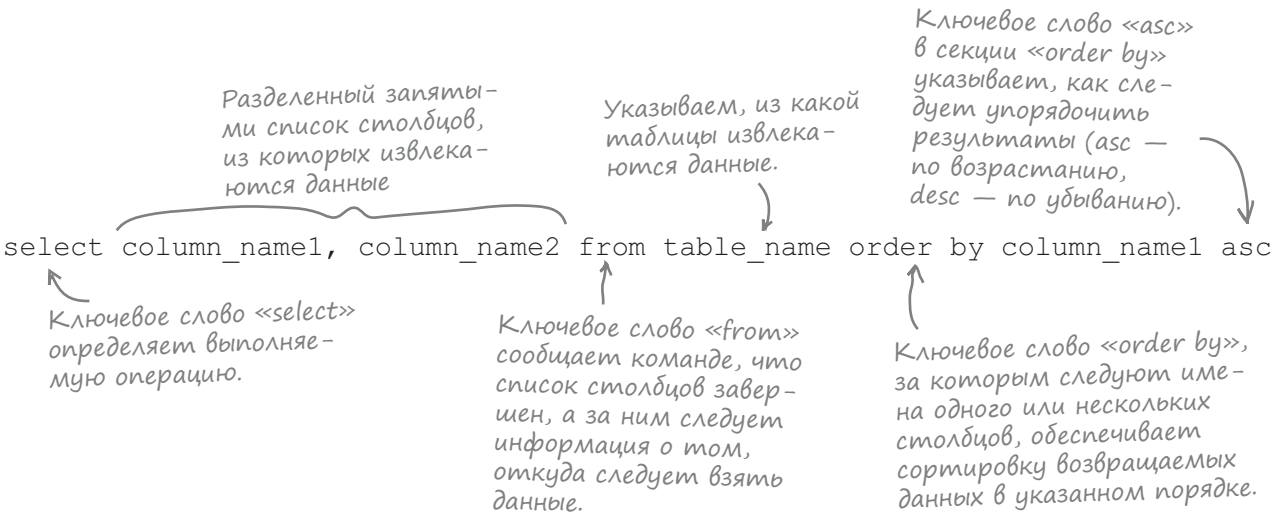
В: Какие еще существуют библиотеки PHP?

О: Существует много разных библиотек PHP для разных целей: SSL, работы с электронной почтой (SMTP или IMAP), сжатия данных, проверки подлинности, подключения к другим базам данных и т. д. Чтобы получить полный список, введите в своей любимой поисковой системе «библиотеки PHP».

Чтение из базы данных

Для чтения информации из базы данных используется команда `select`. Данные возвращаются в виде итогового набора (*resultset*) — совокупности всех данных, запрашиваемых в запросе `select`. Команда `select` также позволяет объединить информацию из нескольких таблиц; таким образом, итоговый набор может содержать данные из двух и более таблиц.

Более подробная информация о PHP, SQL, базах данных и таблицах приведена в книге «Head First PHP & MySQL».



Команда SQL select читает столбцы данных из одной или нескольких таблиц и возвращает данные в виде итогового набора.

Возьми в руку карандаш

Создайте команду `select` для получения данных, необходимых для отображения списка участников на сайте. Из таблицы `runners` нужно прочитать столбцы `first_name`, `last_name`, `gender` и `finish_time`. Отсортируйте данные по столбцу `finish_time`, от низких значений к высоким. Если понадобится, имена столбцов можно уточнить на с. 366, где мы создавали таблицу.

Возьми в руку карандаш
Решение

Вы только что создали собственную команду SQL для чтения информации об участниках из базы данных.

Список столбцов, из которых читаются данные

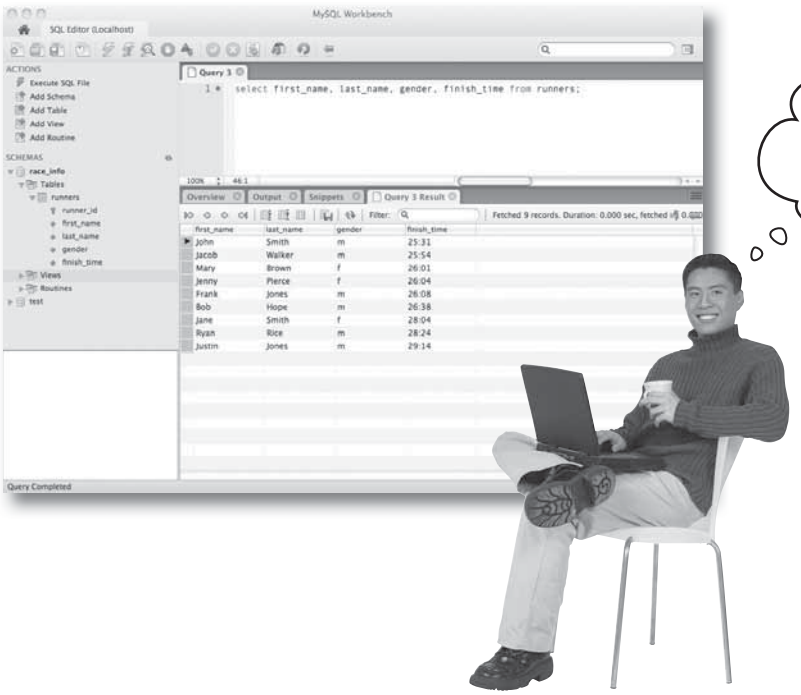
Способ упорядочения данных

```
SELECT first_name, last_name, gender, finish_time FROM runners order by finish_time ASC
```

Таблица, из которой читаются данные

ТЕСТ-ДРАЙВ

Выполните команду `select` в MySQL Workbench. Убедитесь в том, что все данные возвращаются в итоговом наборе.



Потрясающе! Я вижу данные в Workbench... но ведь они нужны нам на веб-странице?

Да, это верно.

Давайте посмотрим, как вывести на странице информацию, прочитанную из базы данных.

Доступ к данным в коде PHP

До настоящего момента мы рассмотрели некоторые простейшие (да и не только простейшие) возможности PHP. Вы уже умеете выводить на экран простые сообщения, умеете подключаться к базе данных и писать команды `select` для чтения информации. Теперь давайте разберемся, как вывести на экран информацию, прочитанную из базы данных.

Развлечения с магнитами

Сложите из магнитов код PHP, в котором определяется функция `db_connection` для подключения к базе данных. Создайте переменную `$query` и присвойте ей текст написанной ранее команды `select` для выборки информации всех участников из базы данных. Создайте переменную `$result`, которой присваивается результат вызова функции `db_connection` с передачей переменной `$query` в параметре. Наконец, переберите в цикле `while` все строки итогового набора (данные которого хранятся в ассоциативном массиве) и выведите их на экран.

```
<?php

   $query = "SELECT first_name, last_name, gender, finish_time _____ runners
order by _____________ASC ";
   $result = _____________________($query);

   while ($row = mysql_fetch_array(_____________, MYSQL_ASSOC)) {
     print_r(_______);
   }

   _________________db_connection(__________) {
     mysql_connect('127.0.0.1', 'runner_db_user', 'runner_db_password')
        OR _____ ('Could not connect to database.');
     _____________________('hfjq_race_info');

     return mysql_query($query);
   }
?>
```

$row

die

mysql_select_db

FROM

db_connection

$query

$result

function

finish_time

PHP

service.php

Развлечения с магнитами. Решение

Перед вами небольшой фрагмент кода PHP, который читает информацию из базы данных, перебирает содержимое полученного массива и выводит его на веб-странице.

```
<?php

   $query = "SELECT first_name, last_name, gender, finish_time FROM runners
order by finish_time ASC ";
   $result = db_connection($query);

   while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
     print_r($row);
   }

   function db_connection($query) {
     mysql_connect('127.0.0.1', 'runner_db_user', 'runner_db_password')
       OR die('Could not connect to database.');
     mysql_select_db('hfjq_race_info');

     return mysql_query($query);
   }
?>
```

service.php

Часто Задаваемые Вопросы

В: Команда `select` возвращает всю информацию из таблицы? Я могу ограничить состав возвращаемых столбцов, а как насчет строк?

О: Да, строки можно ограничить при помощи секции `where`. Мы поговорим об этом в главе 11, а сейчас достаточно сказать, что в секции `where` передается условие-фильтр, и команда `select` возвращает строки, которые удовлетворяют указанному условию.

В: Команда может читать данные только из одной таблицы?

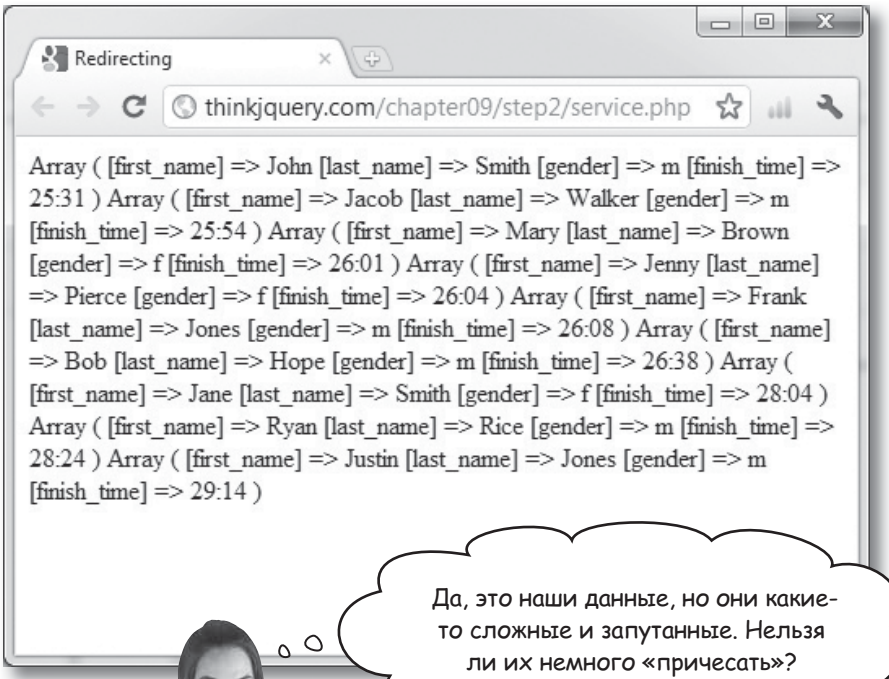
О: В запросе можно объединить сколько угодно таблиц, обычно для этого используется общий идентификатор или условие `where`. Объединение большого количества таблиц существенно замедляет запросы к базе данных, поэтому будьте осторожны при использовании этой возможности. Чтобы получить дополнительную информацию по этой теме, почитайте главу 8 книги «*Изучаем PHP & MySQL*» или главу 2 «*Изучаем SQL*».

В: Почему база данных находится по адресу 127.0.0.1? Я вижу свой сайт на «localhost». Откуда такие различия?

О: Хороший вопрос. Ответ: различий нет. Адрес 127.0.0.1 и имя «localhost» обозначают одно и то же — компьютер/сервер, на котором вы работаете в настоящий момент.

ТЕСТ-ДРАЙВ

Включите в файл *service.php* только что написанный код. Откройте страницу в браузере, чтобы увидеть результаты запроса к базе данных. Не забывайте, что код PHP должен выполняться через веб-сервер, поэтому URL-адрес должен иметь префикс *http://*, а не *file://*.



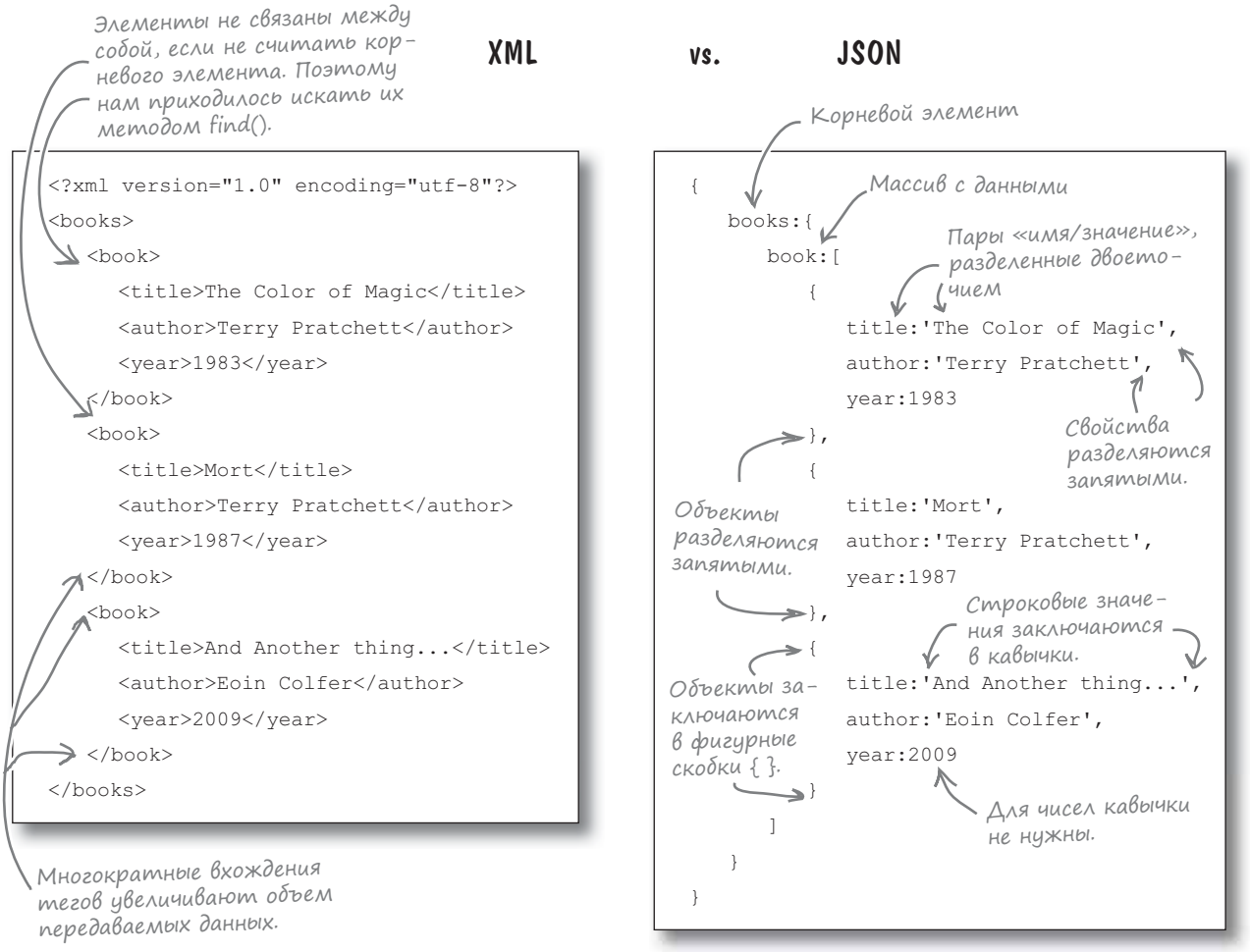
Результат: все данные успешно выведены на экран.

Безусловно!

То, что мы видим перед собой — это выведенное на экран содержимое массивов. Массивы содержат нужные данные, но не в том формате, который нам нужен. К счастью, существует эффективный способ организации данных в формате, идеально подходящем для определения структур данных.

На помощь приходит JSON!

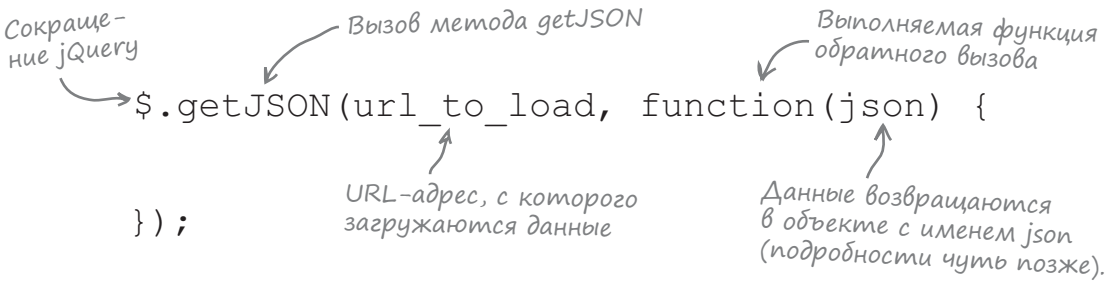
JSON (сокращение от JavaScript Object Notation) — облегченный формат передачи данных. Обычный человек сможет читать и писать в этом формате. Он легко разбирается и генерируется компьютером. Поэтому JSON идеально подходит для структурирования и передачи данных. JSON основан на подмножестве стандарта JavaScript и нейтрален к языку — может использоваться практически с любым языком программирования. Формат JSON превосходит XML по эффективности передачи данных; фактически он рассматривает пары «имя/значение» как ассоциативные массивы. Значения могут быть строками, числами, массивами, объектами, логическими значениями («истина/ложь») или null.



Для обращения к информации в объектах JSON используется та же запись, как при обращении к другим объектам — точечный синтаксис (.). Массивы в объектах JSON не отличаются от обычных массивов JavaScript и обладают теми же свойствами (например, `length`). В приведенном объекте JSON, чтобы узнать количество полученных книг нужно использовать конструкцию `books.book.length`. Объекты JSON имеют разную структуру, для обращения к объекту массива столько точек может и не понадобиться.

jQuery + JSON = потрясающе

Формат JSON очень широко распространен и прост в использовании, поэтому создатели jQuery предусмотрели специальное сокращение для работы с данными JSON: метод `getJSON`.



Если эта запись выглядит знакомо — это потому, что она почти не отличается от метода `post`, использованного нами ранее для получения данных от формы. Этот простой метод представляет собой сокращенную запись для вызова метода `ajax`, в которой несколько параметров уже заданы заранее. В полной записи этот вызов выглядел бы так:

```
$.ajax({
    url: url_to_load,
    dataType: 'json',
    data: json,
    success: function(json){

     };
});
```

Но наши данные не хранятся в формате JSON — это простой набор массивов. Можно ли преобразовать эти массивы в JSON?

Да, можно.

Нам снова повезло: создатели PHP уже позаботились об этом. Продолжим изучение базовых возможностей PHP, а затем посмотрим, как объединить их с другими функциями PHP для получения данных в формате JSON.

Несколько правил PHP...

По правде говоря, никто не любит правила программирования. Но у PHP есть еще несколько особенностей (большей частью из области синтаксиса), которые необходимо знать для подготовки данных к использованию в jQuery. К счастью, многие из этих концепций уже встречались нам в контексте JavaScript, так что изучение будет настолько быстрым и безболезненным, насколько это возможно...

Основные правила PHP

1. Весь код PHP должен быть заключен в теги<?php и ?> .
2. **PHP может чередоваться с HTML**, при этом код PHP заключается в теги <?php и ?>.
3. Все строки кода PHP должны заканчиваться точкой с запятой **(;)**.

```
<div><span> Hello
<?php
   echo "Bob";
?>
</span></div>
```

Правила для переменных

1. Имена всех переменных должны начинаться со знака доллара **($)**.
2. За $ следует как минимум одна буква или знак подчеркивания, далее идет произвольная комбинация букв, цифр и подчеркиваний.
3. Дефисы (-), пробелы ( ) и **любые специальные символы** (кроме $ и _) в именах переменных ***запрещены***.

```
<?php
$u = "USA"; // OK
$home_country = "Ireland"; // OK
$another-var = "Canada"; // Causes an error
?>
```

Правила для циклов

1. В PHP поддерживаются циклы `for`, `while` и `do...while` — с таким же синтаксисом, как в **JavaScript**.
2. В PHP также существует дополнительная разновидность циклов — так называемый цикл `foreach`, который последовательно перебирает элементы массива до обнаружения конца массива, после чего автоматически останавливается.

```
<?php
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
while ($j <= 10) {
    echo $j++;
}
$a = array(1, 2, 3, 17);
foreach ($a as $v) {
    echo "Current value: $v.\n";
}
?>
```

Правила PHP (еще немного)...

Осталось еще несколько правил, которые помогут вам получить нужные данные, правильно отформатировать их и вывести на веб-странице.

Правила для массивов

1. Новые массивы создаются ключевым словом `array` (как и в JavaScript).
2. Обращения к ячейкам массивов выполняются по индексу ячейки, который указывается в квадратных скобках [ ] (как и в JavaScript). Индексы начинаются с нуля, как и в JavaScript.
3. Массивы могут быть **ассоциативными**, обращение к ячейкам осуществляется по ключу, а не по индексу. Содержимое таких массивов представляет собой **пары «ключ/значение»**.
4. Чтобы сохранить в ассоциативном массиве новую пару «ключ/значение», используйте **оператор =>**.

```
<?php
$my_arr2 = array('USA', 'China',
'Ireland');
echo $my_arr2[2]; // Выводит "Ireland"

$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"]; // Выводит "bar"
echo $arr[12]; // Выводит true
?>
```

Правила условных конструкций

1. Команда `if` **имеет такой же синтаксис, как в JavaScript (это относится и к условию** `else`, и к `else if`).
2. Операторы сравнения работают так же, как в **JavaScript**.
3. **Логические операторы тоже не отличаются от JavaScript**, с одним дополнением: вместо операторов могут использоваться их эквиваленты `and`, `or` и `not`.

```
<?php
if ($x > $y){
    echo "x is greater than y";
}
elseif ($x == $y) {
    echo "x is equal to y";
}
else {
    echo "x is smaller than y";
}
?>
```

Правила вывода на экран

1. Для вывода на экран используются ключевые слова `echo` и `print`.
2. Содержимое **массива** выводится командой `print_r`.

```
<?php
    echo "Bob";
    print_r($my_arr2);
?>
```

Форматирование вывода средствами PHP

Хорошо, мы разобрались с основами, теперь посмотрим, что PHP может сделать для нас! Функция `json_encode` в PHP преобразует ассоциативный массив в строку значений, закодированную в формате JSON.

Значение записывается в источник вызова (браузер, вызов ajax и т. д.).

Вызов этой функции PHP кодирует массив в формате JSON.

Кодируемый массив

```
echo json_encode(array_name);
```

Но прежде чем данные можно будет закодировать, они должны оказаться в одном ассоциативном массиве. Мы уже видели метод для перебора итогового набора и вывода всех входящих в него ассоциативных массивов. Но сейчас нам нужно взять все эти массивы и объединить их в один массив. При помощи функции PHP `array_push` мы можем добавлять новые элементы в конец массива.

Создаем новый, пустой массив.

```
$my_array = array();
```

Здесь передается любая информация, добавляемая в массив. В данном случае в массив $my_array включается другой ассоциативный массив.

```
array_push($my_array, array('my_key' => 'my_val'));
```

Вызываем функцию array_push с параметрами.

Приемный массив передается в первом параметре.

Пара «имя/значение», включаемая в этот массив

Для любознательных

Функция `json_encode` стала доступна только в PHP версии 5.2. Если вы используете более раннюю версию PHP, либо обновите ее, либо введите строку «json_encode PHP альтернативы» в своей любимой поисковой системе и найдите описание этой функции от создателей PHP. После этого вы сможете написать собственную реализацию функции, чтобы пользоваться всеми ее замечательными возможностями.

Часто Задаваемые Вопросы

В: Кто придумал JSON? Создатели jQuery?

О: Нет. Дуглас Крокфорд, специалист по JavaScript из Yahoo!, изобрел JSON как то, что он назвал «обезжиренной альтернативой XML». Он приводит обоснования этой характеристики в статье: *http://www.json.org/fatfree.html*.

В: Разве JSON это не JavaScript?

О: И да, и нет. JSON базируется на подмножестве JavaScript, ECMA 262 Third Edition, но может использоваться в разных языках для передачи данных. Список языков, поддерживающих JSON, приведен по сайте *http://www.json.org/*.

В: Выходит, JavaScript и PHP имеют похожий синтаксис. Почему же я не могу просто использовать JavaScript?

О: Как мы уже упоминали, PHP является *серверным* языком сценариев и может взаимодействовать с веб-сервером и базами данных от вашего имени. Код выполняется на сервере и генерирует разметку HTML, которая затем передается клиенту. JavaScript, напротив, работает только в вашем браузере, а все его взаимодействия осуществляются на стороне *клиента*.

В: Понятно. Еще раз, что такое PHP?

О: PHP (рекурсивное сокращение от «PHP: Hypertext Preprocessor») — популярный, распространяемый с открытым кодом язык сценариев общего назначения, который особенно хорошо подходит для веб-программирования и может встраиваться в HTML.

В: Откуда взялся PHP?

О: Язык PHP появился в 1994 году. Он был создан Расмусом Лердорфом и предназначался для отображения его резюме в Интернете. В июне 1995 года был опубликован исходный код, что позволило другим разработчикам заняться обновлением и исправлением ошибок. Проект ждало большое будущее: сейчас он используется на более чем 20 миллионах сайтов по всему миру.

Расслабьтесь

Вы узнали много нового о PHP, MySQL и JSON. Сейчас мы возьмемся за большое упражнение, которое поможет сложить все фрагменты воедино, так что отдохните и выпейте чашку кофе, прогуляйтесь, сделайте что-нибудь такое, чтобы ваш мозг отдохнул и был готов к тому, что его ожидает. А когда все будет сделано, переверните страницу, и... вперед!

Длинные упражнения

Включите в файл *my_scripts.js* новую функцию `getDBRacers`, которая обращается с вызовом к файлу *service.php*. Вызов должен возвращать объект JSON, а затем отображать сообщение с количеством возвращенных участников. Также измените таймер `startAJAXCalls`, чтобы вместо функции `getXMLRunners` вызывалась эта новая функция. Наконец, измените файл *service.php*, чтобы он возвращал данные участников, прочитанные из базы данных и закодированные в формате JSON.

```
function startAJAXcalls(){
   if(repeat){
      setTimeout( function() {
      ..................
      startAJAXcalls();
      },
      FREQ
   );
   }
}

function getDBRacers(){
   $.getJSON(.......... function(.....) {
      .....(json.runners........);
   });
   getTimeAjax();
}
```

my_scripts.js

```
<?php

   $query = "SELECT first_name, last_name, gender, finish_time FROM runners
order by finish_time ASC ";
   $result = ..................($query);

   $runners = array();

   while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
      .............($runners, array('fname' => $row['first_name'], 'lname' =>
$row['last_name'], 'gender' => $row['gender'], 'time' => $row['finish_time']));
   }
   echo .............(array("runners" => .............));
   exit;

   function db_connection($query) {
      mysql_connect('127.0.0.1', 'runner_db_user', 'runner_db_password')
        OR die(fail('Could not connect to database.'));
      mysql_select_db..................

      return mysql_query($query);
   }

   function fail($message) {
      die(json_encode(array('status' => 'fail', 'message' => $message)));
   }
   function success($message) {
      die(json_encode(array('status' => 'success', 'message' => $message)));
   }
?>
```

service.php

Решение длинных упражнений

В файле *my_scripts.js* появилась новая функция `getDBRacers`, которая обращается с вызовом к файлу *service.php*. Старая функция `getXMLRunners` стала лишней, ее можно удалить. Новая функция получает данные JSON, возвращенные файлом *service.php*, и выводит количество полученных участников. Функция `startAJAXCalls` также изменилась; теперь в ней вызывается новая функция. Обновленный файл *service.php* возвращает участников, прочитанных из базы данных, закодированных в формат JSON и упорядоченных по возрастанию значений `finish_time`.

```
function startAJAXcalls(){
   if(repeat){
      setTimeout( function() {
      getDBRacers();
      startAJAXcalls();
      },
      FREQ
   );
   }
}

function getDBRacers(){
   $.getJSON("service.php", function(json) {
      alert(json.runners.length);
   });
   getTimeAjax();
}
```

Планирование вызова новой функции

Данные, полученные в результате вызова getJSON

Метод jQuerygetJSON обращается с вызовом к файлу service.php.

Как и другие массивы, этот также имеет свойство length.

Объект json содержит массив с именем runners, полученным при вызове метода json_encode в PHP.

my_scripts.js

```
<?php
   $query = "SELECT first_name, last_name, gender, finish_time FROM runners
order by finish_time ASC ";
   $result = db_connection($query);

   $runners = array();

   while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
      array_push.($runners, array('fname' => $row['first_name'], 'lname' =>
$row['last_name'], 'gender' => $row['gender'], 'time' => $row['finish_time']));
   }
   echo json_encode(array("runners" => $runners));
   exit;

   function db_connection($query) {
      mysql_connect('127.0.0.1', 'runner_db_user', 'runner_db_password')
        OR die(fail('Could not connect to database.'));
      mysql_select_db ('hfjq_race_info');.

      return mysql_query($query);
   }
   function fail($message) {
      die(json_encode(array('status' => 'fail', 'message' => $message)));
   }
   function success($message) {
      die(json_encode(array('status' => 'success', 'message' => $message)));
   }
?>
```

Запрос к базе данных для получения информации об участниках

Создаем новый массив для хранения полученных данных.

Перебираем итоговый набор, получаем ассоциативные массивы.

Полученные данные помещаются в ассоциативный массив.

Ассоциативный массив кодируется в JSON и записывается в источник вызова.

Функция для взаимодействия с базой данных

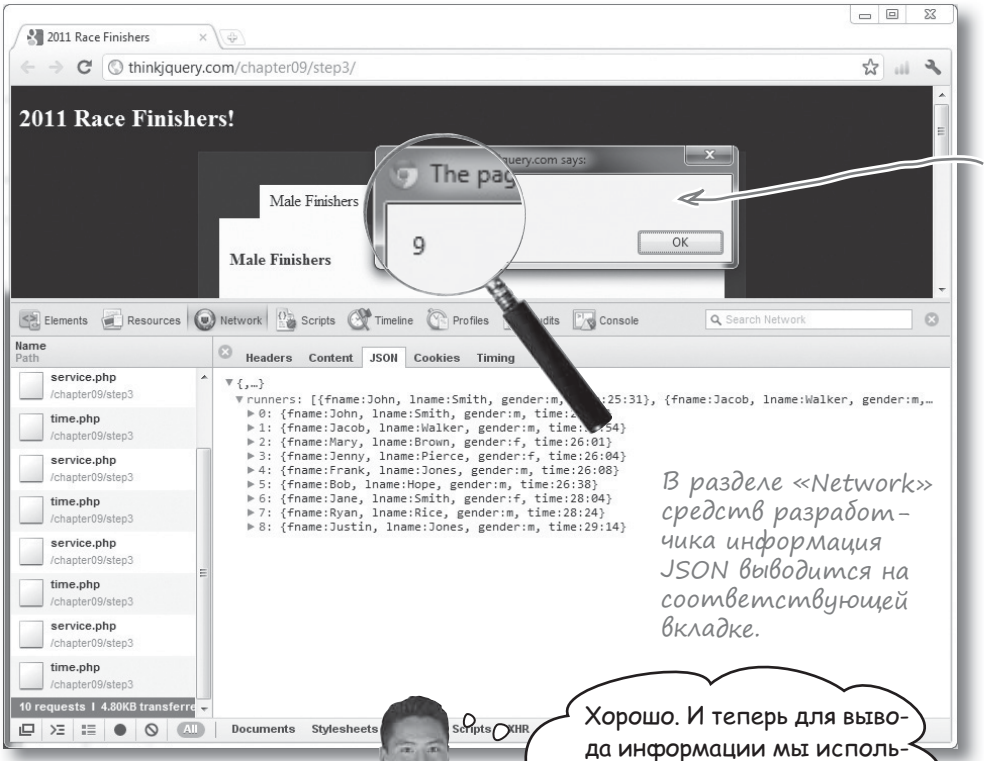
Итоговый набор возвращается стороне, вызвавшей эту функцию.

Функции обработки ошибки или успешного завершения в наших сценариях

service.php

ТЕСТ-ДРАЙВ

Включите новый код в файлы *service.php* и *my_scripts.js*, откройте файл *index.html* в браузере. Откройте вкладку «Network» в средствах разработчика; убедитесь в том, что информация JSON загружается успешно.



функция выдает количество записей, возвращенных в объекте JSON.

В разделе «Network» средств разработчика информация JSON выводится на соответствующей вкладке.

Хорошо. И теперь для вывода информации мы используем **find** и **each**, как при работе с XML?

Не совсем.

Мы знаем, что данные хранятся в формате JSON, потому что они были созданы в нашем коде PHP. Пора (наконец-то) плотно заняться объектом JSON, о котором мы так много говорили, и вывести из него полученные данные.

Работа с данными в объекте JSON

Функция PHP `json_encode` преобразует ассоциативный массив в строку значений, закодированную в формате JSON. Далее с этими данными можно работать в коде JavaScript как с ассоциативными массивами; таким образом, мы можем перебрать их и обработать так же, как мы обрабатываем данные в других массивах.

При работе с XML нам приходилось перебирать данные для поиска следующего участника. Таким образом, после обнаружения участника нам приходилось проводить повторный поиск для проверки пола. Помните объект JSON, который возвращается функцией `json_encode`? Мы можем напрямую обращаться к свойствам этого объекта при помощи точечного синтаксиса (.). Массив `runners` содержится в этом объекте в виде свойства. А получив доступ к массиву, мы можем по ключу ассоциативного массива определить пол участника, что намного эффективнее повторного поиска.

Возьми в руку карандаш

Измените функцию `getDBRunners` так, чтобы она получала объект JSON от файла *service.php*. Выбирайте список, в который нужно занести участника, используя условную конструкцию. Будьте внимательны! Проверьте, содержит ли объект JSON данные хотя бы одного участника.

```
function getDBRacers(){
   $.getJSON(......................, function(json) {
      if (json.runners.....................> 0) {
         $('#finishers_m').empty();
         $('#finishers_f').empty();
         $('#finishers_all').empty();
         $......(json.runners,function() {
            var info = '<li>Name: ' + this['fname'] + ' ' + this['lname'] + '. Time: ' +
this[.....] + '</li>';
            if(this['gender'] == 'm'){
               $('.....').append( info );
            }else if(this['gender'] == 'f'){
               $('#finishers_f').append(.....);
            }else{}
            $('.....').append( info );
         });
      }
   });
   getTimeAjax();
}
```

my_scripts.js

Возьми в руку карандаш
Решение

Используя условную конструкцию и информацию, полученную в объекте JSON, можно определить, в какой список следует включить участника. Как и прежде, все участники должны включаться в общий список `all_finishers`.

```
function getDBRacers(){
   $.getJSON("service.php", function(json) {
      if (json.runners.length> 0) {
         $('#finishers_m').empty();
         $('#finishers_f').empty();
         $('#finishers_all').empty();
         $.each(json.runners,function() {
            var info = '<li>Name: ' + this['fname'] + ' ' + this['lname'] + '. Time: ' +
this['time'] + '</li>';
            if(this['gender'] == 'm'){
               $('#finishers_m').append( info );
            }else if(this['gender'] == 'f'){
               $('#finishers_f').append(info);
            }else{}
            $('finishers_all').append( info );
         });
      }
   });
getTimeAjax();
}
```

Получаем информацию от файла service.php.

Проверяем, имеются ли данные в массиве runners.

Очищаем все списки.

Проверяем свойство gender текущего объекта (m или f).

Включаем участника в список all_runners.

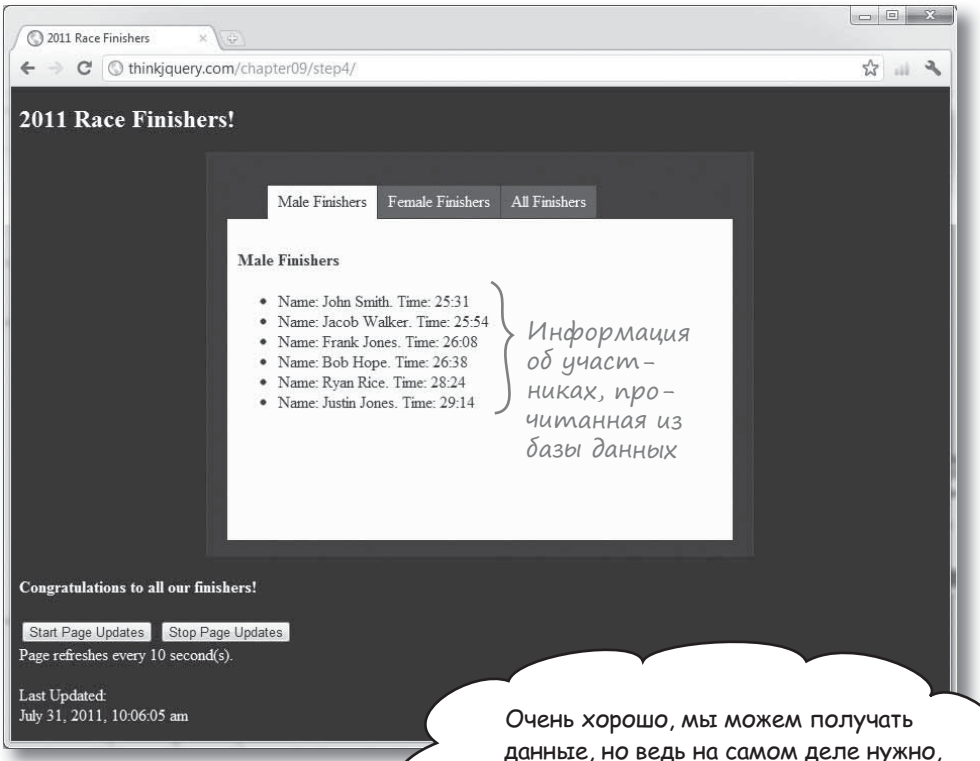
my_scripts.js

Для любознательных

Для перебора всех элементов массива, возвращенного в объекте JSON, можно воспользоваться методом `each`. Этот метод несколько отличается от метода `(селектор).each` тем, что он позволяет перебирать содержимое массивов, не относящихся к jQuery, таких как наш массив `runners`.

ТЕСТ-ДРАЙВ

Измените функцию `getDBRacers` **в файле** *my_scripts.js*. Откройте файл *index.html* и убедитесь в том, что данные участников успешно загружаются из базы данных MySQL с использованием Ajax, JSON и PHP.



Очень хорошо, мы можем получать данные, но ведь на самом деле нужно, чтобы этот способ работал с созданной нами формой. Верно?

Очень верное замечание.

А когда это будет сделано — добро пожаловать на Гавайи! Но сначала нужно убедиться в том, что вводимые данные не причинят вреда нашему приложению.

Проверка и чистка данных в PHP

В последнее время развелось слишком много спам-ботов и хакеров, пытающихся взять под контроль ваши данные для неблаговидных целей. Никогда не доверяйте данным, введенным на веб-формах! ***Всегда*** выполняйте *проверку* и чистку информации, передаваемой серверу, перед вставкой в базу данных. Эти операции гарантируют, что тип данных соответствует типу конкретного поля (проверка), а полученные данные не содержат информации, опасной для вашего сервера или базы данных (чистка). Подобные меры помогают защититься от внедрения SQL, межсайтовых сценарных атак и многих других неприятностей, о которых можно узнать в Интернете. Мы воспользуемся методами PHP, которые предотвратят возможные проблемы и гарантируют, что используемые данные будут корректны.

Преобразует некоторые специальные символы HTML в формат, безопасный для баз данных

```
<?php
    htmlspecialchars($_POST["a"]) ; // Замена символов безопасными комбинациями
    empty($_POST["b"]) ; // Метод "empty" проверяет пустые значения
    preg_match('',$var); //"Регулярное выражение". $var проверяется на соответствие шаблону.
?>
```

Выявление пустых строк

Функция проверки по регулярному выражению. Шаблоны регулярных выражений могут быть предельно точными, что позволяет жестко контролировать вводимые данные.

Существует много других функций, используемых для чистки данных: `htmlentities`, `trim`, `stripslashes`, `mysql_real_escape_string` и многие другие. За более подробной информацией обращайтесь к главе 6 книги «Изучаем PHP & MySQL».

Использование одного файла PHP для разных целей

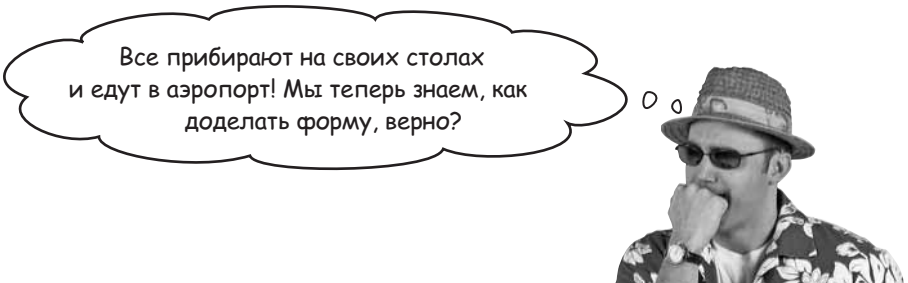
Мы рассмотрели два способа отправки данных серверу для обработки файлом PHP: POST и GET. При помощи условной конструкции можно определить, какой метод использовался при вызове файла PHP, и выполнить соответствующие действия. Помните скрытое поле, которое мы добавили на форму несколько страниц назад?

```
<input type="hidden" name="action" value="addRunner" id="action">
```

Если это значение присутствует, значит выполняется отправка данных формы методом POST. В этом случае нужно выполнить функции проверки и чистки данных и убедиться, что получены все необходимые данные. Если обновить функцию `getJSON` для чтения участников из базы данных с параметром из URL-адреса (для объекта `$_GET` PHP), можно отделить и выполнить только этот фрагмент кода в файле PHP. Значит, нам придется сопровождать только один файл PHP.

```
$.getJSON("service.php?action=getRunners", function(json) {
```

Приказываем функции PHP выполнить код, связанный с чтением информации участников из базы данных.



Готово к употреблению

Внесите в файл *service.php* изменения. Файл будет обрабатывать оба вида запросов, GET и POST. Включите в него функции `db_connection`, `success` и `fail`, которые мы создали ранее.

```
<?php
   if ($_POST['action'] == 'addRunner') {
      $fname = htmlspecialchars($_POST['txtFirstName']);
      $lname = htmlspecialchars($_POST['txtLastName']);
      $gender = htmlspecialchars($_POST['ddlGender']);
      $minutes = htmlspecialchars($_POST['txtMinutes']);
      $seconds = htmlspecialchars($_POST['txtSeconds']);
      if(preg_match('/[^\w\s]/i', $fname) || preg_match('/[^\w\s]/i', $lname)) {
         fail('Invalid name provided.');
      }
      if( empty($fname) || empty($lname) ) {
         fail('Please enter a first and last name.');
      }
      if( empty($gender) ) {
         fail('Please select a gender.');
      }
      $time = $minutes.":".$seconds;
      $query = "INSERT INTO runners SET first_name='$fname', last_name='$lname',
gender='$gender', finish_time='$time'";
      $result = db_connection($query);
      if ($result) {
         $msg = "Runner: ".$fname." ".$lname." added successfully" ;
         success($msg);
      } else {  fail('Insert failed.');} exit;
   }elseif($_GET['action'] == 'getRunners'){
      $query = "SELECT first_name, last_name, gender, finish_time FROM runners order by
finish_time ASC ';
      $result = db_connection($query);
      $runners = array();
      while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
         array_push($runners, array('fname' => $row['first_name'], 'lname' => $row['last_name'],
'gender' => $row['gender'], 'time' => $row['finish_time']));
      }
      echo json_encode(array("runners" => $runners));
      exit;
   }
```

Проверяем, было ли отправлено на сервер значение addRunner (из скрытого поля).

Чистка данных в массиве $_POST

Проверка данных подтверждает, что поля формы были заполнены.

Если проверка не прошла, вызывается функция fail.

Приказываем базе данных вставить новую запись...

...и проверить, успешно была выполнена операция или нет.

Проверяем, было ли отправлено в строке URL-адреса значение getRunners.

Получаем и возвращаем данные участников.

service.php

```
function getDBRacers(){
   $.getJSON("service.php?action=getRunners", function(json) {
      if (json.runners.length > 0) {
         $('#finishers_m').empty();
.
.
      }
   });
   getTimeAjax();
}
```

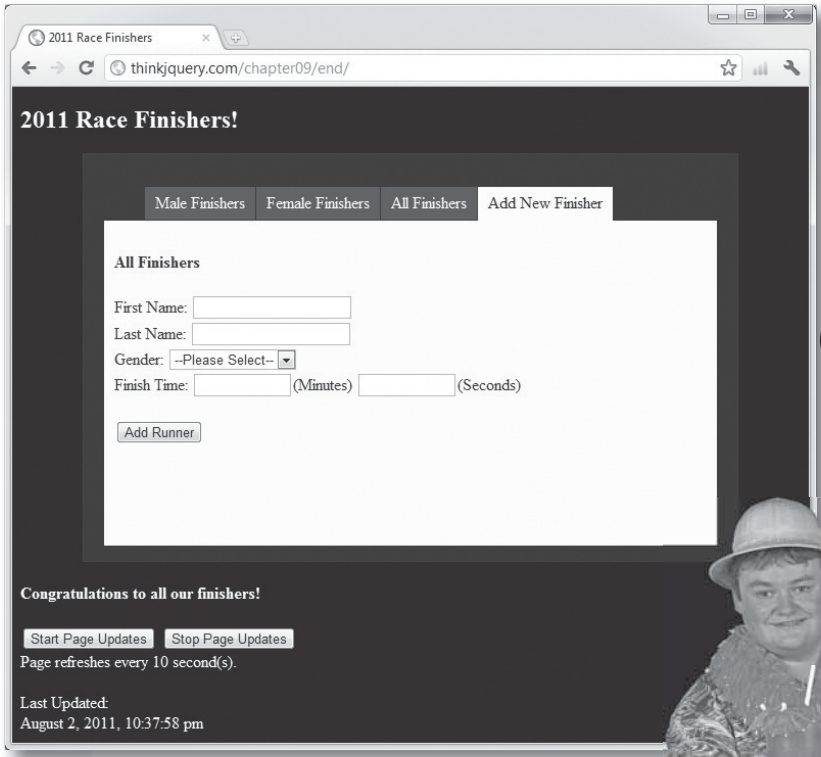
my_scripts.js

Задание!

Обновите вызов `getJSON`, чтобы в нем передавался параметр URL-адреса `action` со значением `getRunners`. По этому параметру файл *service.php* узнает, что он должен вернуть данные участников.

ТЕСТ-ДРАЙВ

Обновите файлы *service.php* и *my_scripts.js*, откройте файл *index.html* в браузере. Убедитесь в том, что данные участников были загружены успешно. Вы также можете ввести данные новых участников в форме на новой вкладке.



Ваш инструментарий jQuery/Ajax/PHP/MySQL

Глава 9 осталась позади. Вы освоили азы работы с PHP, MySQL и JSON, а также расширили свои познания в Ajax.

MySQL

Позволяет хранить информацию в базах данных и таблицах, с выполнением операций чтения и записи информации на языке SQL.

SQL

Язык запросов для взаимодействия с приложениями баз данных (такими как MySQL).

JSON

Функция getJSON используется для получения от сервера данных, закодированных в формате JSON.

Для отправки данных формой используется метод POST. Перед отправкой данные необходимо отформатировать функцией serializeArray.

PHP

Серверный язык сценариев для обработки содержимого веб-страницы до ее передачи клиентскому браузеру.

Сценарий PHP

Текстовый файл с кодом PHP для выполнения операций на веб-сервере.

<?php ?>

Теги, в которые должен быть заключен весь код PHP в ваших сценариях PHP.

echo

Команда PHP для отправки выходных данных в окно браузера.
Синтаксис:

```
echo 'Hello World';
```

$_POST

Специальная переменная для хранения данных формы.

json_encode

Команда преобразует массив в данные, закодированные в формате JSON (необходимо по правилам jQuery).

Переработка форм

Пользователи и их данные — жизнь и смерть веб-приложений. Ввод данных пользователем — серьезная задача, которая может отнять много времени у веб-разработчика. Вы уже видели, как jQuery упрощает построение веб-приложений, использующих Ajax, PHP и MySQL. Теперь давайте посмотрим, как jQuery упрощает построение пользовательского интерфейса форм для ввода данных пользователем. Заодно вы узнаете много полезного о jQuery UI — официальной библиотеке пользовательского интерфейса для jQuery.

Cryptozoologists.org нуждается в переработке

Доктор Паттерсби и доктор Гимли стремятся собрать как можно больше информации о встречах с криптидами по всему миру. Их сайт cryptozoologists.org пользуется уважением как у профессиональных криптозоологов, так и у любителей. У них есть для вас особо важное поручение: обновление устаревшей формы для ввода информации о встречах с криптидами.

Криптиды — существа, неизвестные науке или не признаваемые научным сообществом. Сбор данных о встречах с ними играет важнейшую роль в наших исследованиях.

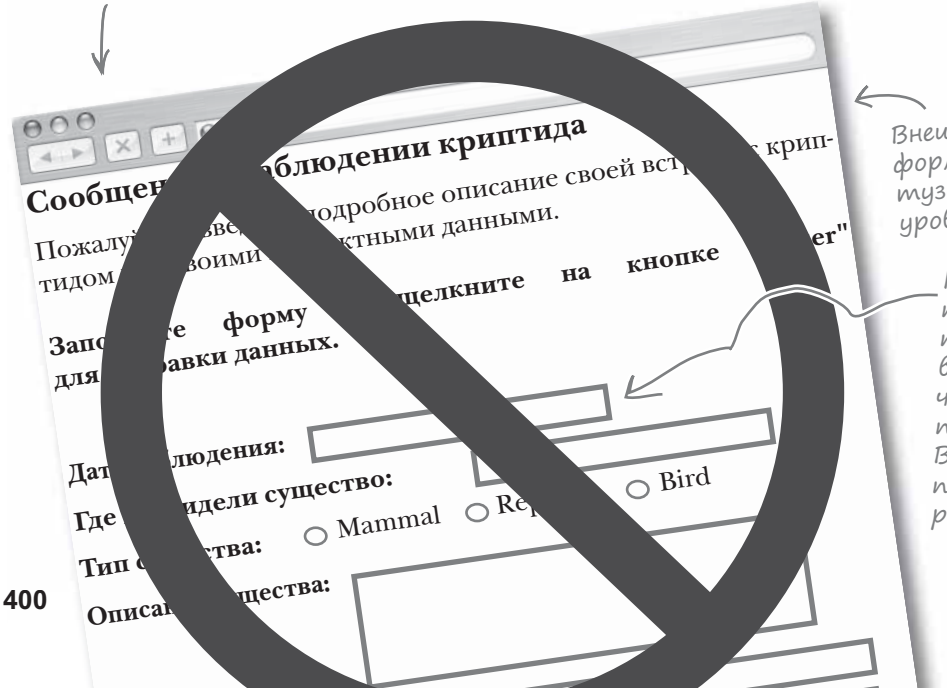
Доктор Паттерсби

Знаменитые криптиды — снежный человек, чупакабра и лохнесское чудовище. Нам нужна более совершенная форма для ввода информации о том, где люди видели этих существ.

Доктор Гимли

Немедленно уберите репортеров!

Криптозоологи хотят избавиться от своей старой, неудобной формы HTML.

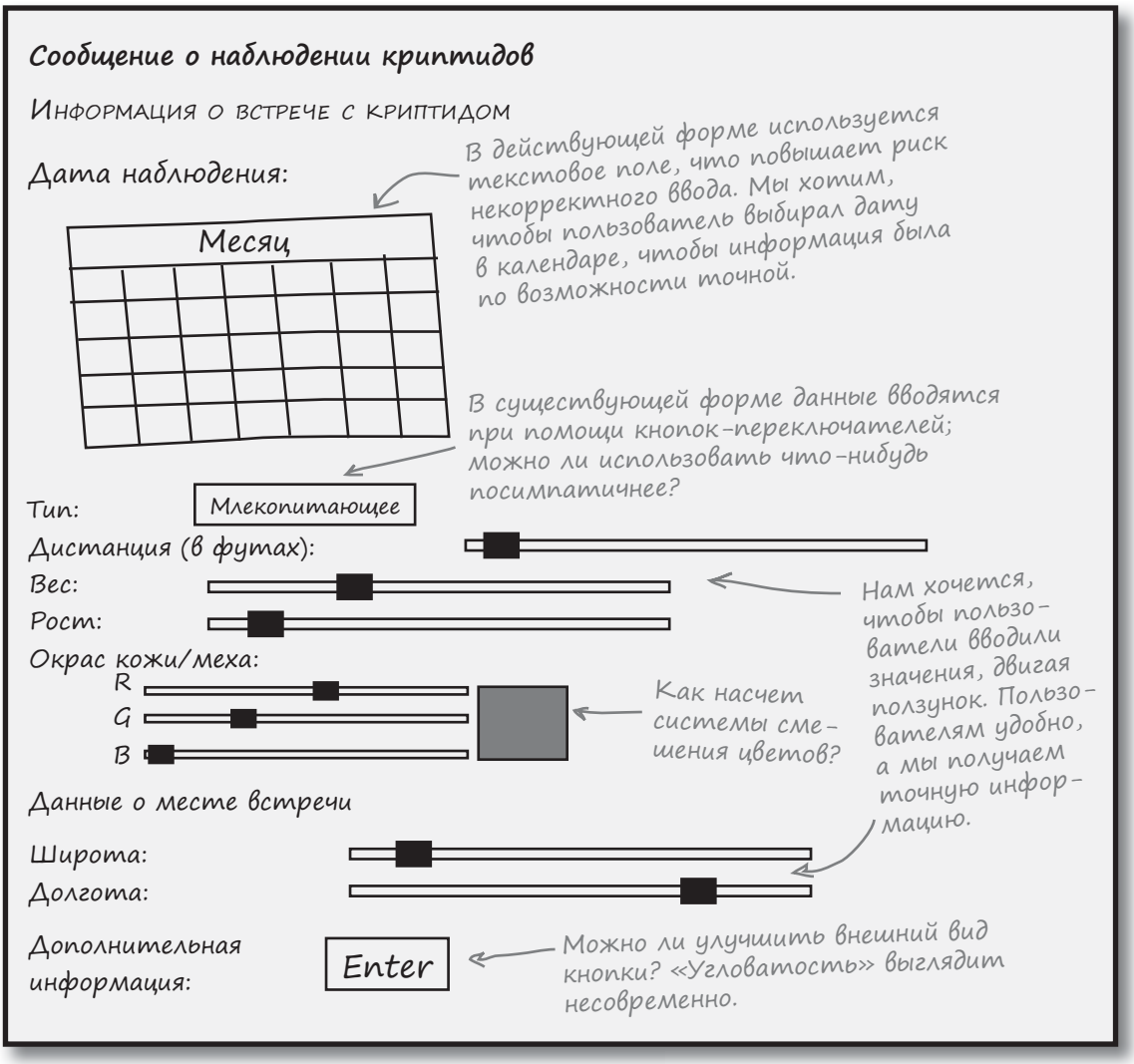


Внешний вид и поведение формы не внушает энтузиазма. Она сделана на уровне конца 1990-х годов.

Нет никаких визуальных подсказок, которые бы помогали пользователю вводить данные. Впрочем, возможности HTML по этой части невелики. В результате доктора получают массу некорректных данных.

Новая форма HTML

Ниже приведен эскиз того, как, по мнению криптозоологов, должна выглядеть новая форма с несколькими дополнительными замечаниями.



МОЗГОВОЙ ШТУРМ

Криптозоологи поставили перед нами непростую задачу. Фактически они хотят построить такой же пользовательский интерфейс, как в настольном приложении. Как вы думаете, jQuery поможет ее решить?

Фрэнк: Я видел. Сейчас они используют форму HTML, но возможностей HTML и CSS недостаточно для новой формы, которую они от нас требуют.

Джим: И не говори. Когда-нибудь пытались применять к элементам форм стили CSS? Визит к стоматологу и то приятнее.

Фрэнк: Да уж, а насчет jQuery... Я пока не видел в jQuery ничего, что помогло бы нам строить такие компоненты интерфейса.

Джо: Придется что-нибудь придумать. Пользователи уже привыкли к современным компонентам, так что мы должны найти способ их создавать.

Фрэнк: Вероятно, для их реализации нам придется использовать комбинацию JavaScript, jQuery и CSS.

Джим: Сколько логики придется написать... Только для временного окна календаря потребуется множество строк кода и сложные стили CSS.

Джо: Хм... Для таких задач должен существовать какой-нибудь модуль расширения jQuery.

Джим: Точно, модули расширения! Мы уже использовали такой модуль пару глав назад для создания вкладок на странице результатов гонки. Значит, вкладки — это еще не все?

Джо: Да, если в jQuery отсутствует функциональность, необходимая разработчику, то разработчик может написать модуль расширения и опубликовать его для использования в сообществе jQuery. Этим он избавляет других разработчиков от лишней работы.

Джим: Так может, какой-нибудь разработчик или группа разработчиков уже занимались этой задачей?

Фрэнк: Это существенно упростило бы нам жизнь.

Джо: Давайте пороемся на jQuery.com и посмотрим, что нам удастся найти.


А как было бы замечательно, если бы существовала библиотека модулей расширения jQuery для пользовательского интерфейса... Но это, конечно, всего лишь мечты...

jQuery UI экономит время и силы

К счастью для всех разработчиков, в jQuery имеется официальная библиотека модулей расширения пользовательского интерфейса для подобных проектов. Эта библиотека называется jQuery UI, и в нее входят три основных типа модулей расширения базовой функциональности jQuery: эффекты, взаимодействия и виджеты.

Эффекты

jQuery UI дополняет jQuery многими новыми эффектами. Заставьте свои элементы прыгать, взрываться, пульсировать или трястись! В jQuery UI также включены *функции плавности* — сложные математические операции, которые делают анимацию более реалистичной.

Взаимодействия

Взаимодействия наделяют веб-приложения более сложным поведением. Для элементов можно разрешить перетаскивание мышью, сортировку и т. д.

В работе с пользовательским интерфейсом в этой главе наше внимание будет уделяться виджетам.

Виджеты

Виджет (widget) представляет собой самостоятельный компонент, который расширяет функциональность веб-приложения. Виджеты экономят время и сокращают сложность кода, и при этом в вашей программе используются работоспособные, эффективные элементы пользовательского интерфейса.

Механизм модулей расширения jQuery позволяет веб-разработчикам расширять базовую функциональность библиотеки jQuery.

ТЕСТ-ДРАЙВ

Опробуйте некоторые эффекты, взаимодействия и виджеты jQuery UI. Посетите следующие URL-адреса и выполните инструкции.

URL	Инструкции
http://jqueryui.com/demos/animate/#default	Щелкните на кнопке Toggle Effect.
http://jqueryui.com/demos/effect/default.html	Выберите эффект из раскрывающегося списка, щелкните на кнопке Run Effect.
http://jqueryui.com/demos/draggable/#default	Нажмите кнопку мыши в прямоугольнике с надписью «Drag me around». Перемещая мышь, двигайте прямоугольник в выделенной области экрана.
http://jqueryui.com/demos/accordion/#default	Щелкая в разных секциях, проследите за тем, как они разворачиваются и сворачиваются.
http://jqueryui.com/demos/dialog/#animated	Щелкните на кнопке Open Dialog, чтобы открыть пользовательское диалоговое окно jQuery UI. Намного лучше обычного, скучного окна сообщения JavaScript, не правда ли?

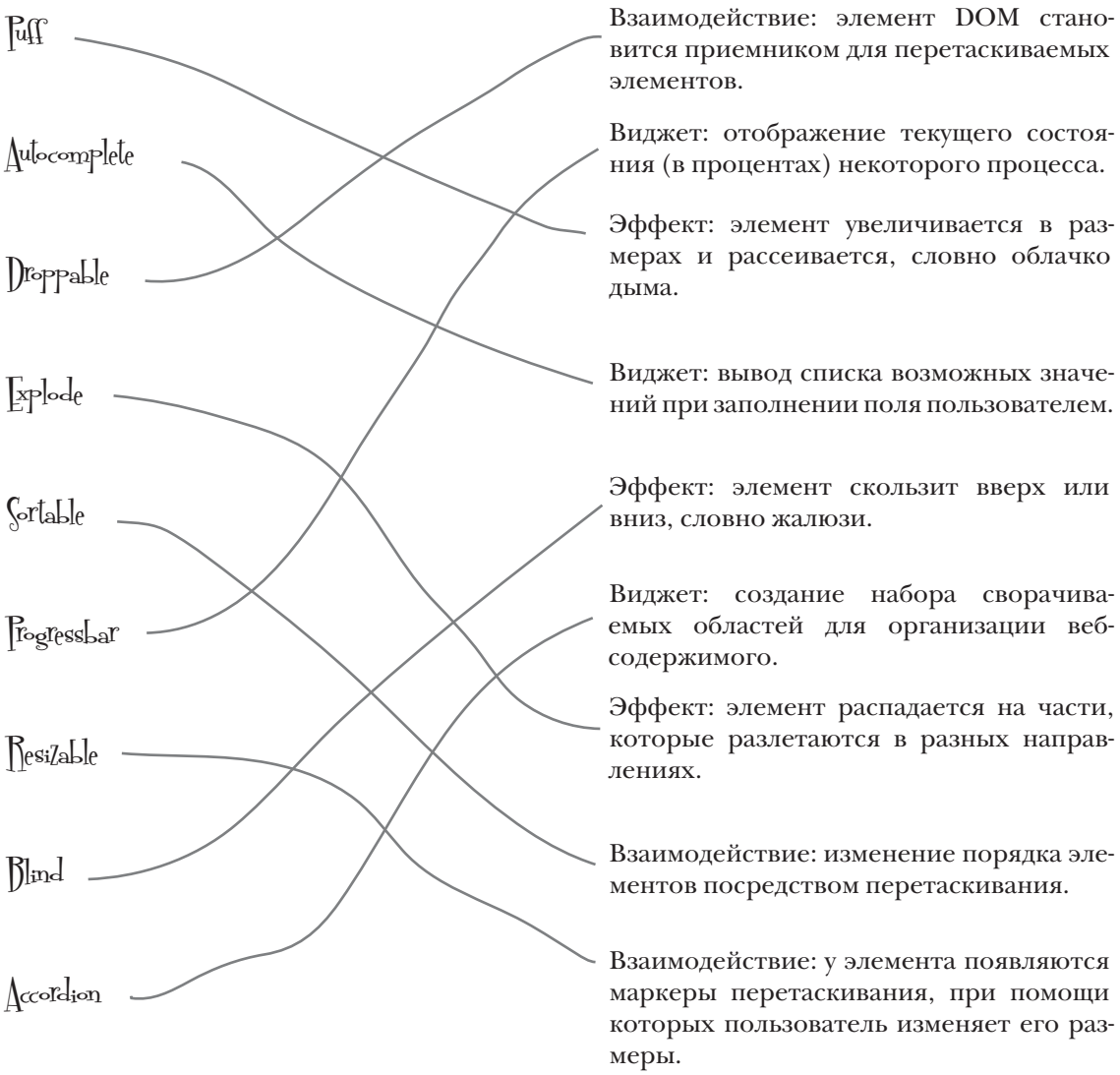
Соедините каждый модуль расширения jQuery UI с описанием типа и действия. Подсказка: если у вас возникнут сомнения, поэкспериментируйте с демонстрационным сайтом Test Drive (см. предыдущую страницу).

Puff	Взаимодействие: элемент DOM становится приемником для перетаскиваемых элементов.
Autocomplete	Виджет: отображение текущего состояния (в процентах) некоторого процесса.
Droppable	Эффект: элемент увеличивается в размерах и рассеивается, словно облачко дыма.
Explode	Виджет: вывод списка возможных значений при заполнении поля пользователем.
Sortable	Эффект: элемент скользит вверх или вниз, словно жалюзи.
ProgressBar	Виджет: создание набора сворачиваемых областей для организации вебсодержимого.
Resizable	Эффект: элемент распадается на части, которые разлетаются в разных направлениях.
Blind	Взаимодействие: изменение порядка элементов посредством перетаскивания.
Accordion	Взаимодействие: у элемента появляются маркеры перетаскивания, при помощи которых пользователь изменяет его размеры.

КТО И ЧТО ДЕЛАЕТ?

РЕШЕНИЕ

Соедините каждый модуль расширения jQuery UI с описанием типа и действия.

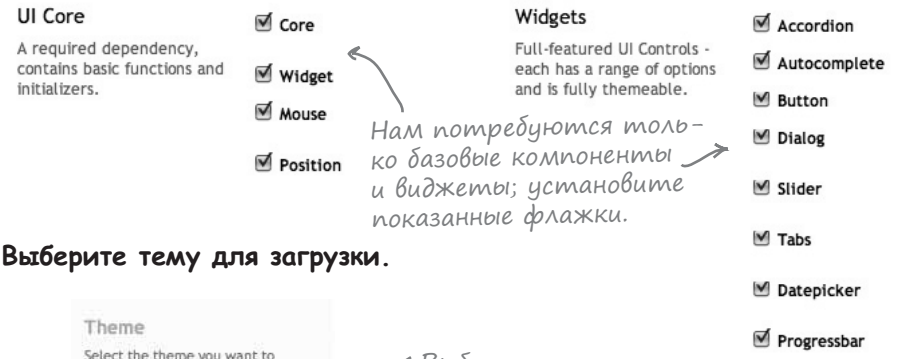


Задание! → Прежде чем сделать что-либо с jQuery UI, необходимо настроить нужные компоненты, выбрать тему и загрузить ее копию. Выполните следующие действия:

1 **Откройте в браузере страницу загрузки jQuery UI.**

http://jqueryui.com/download

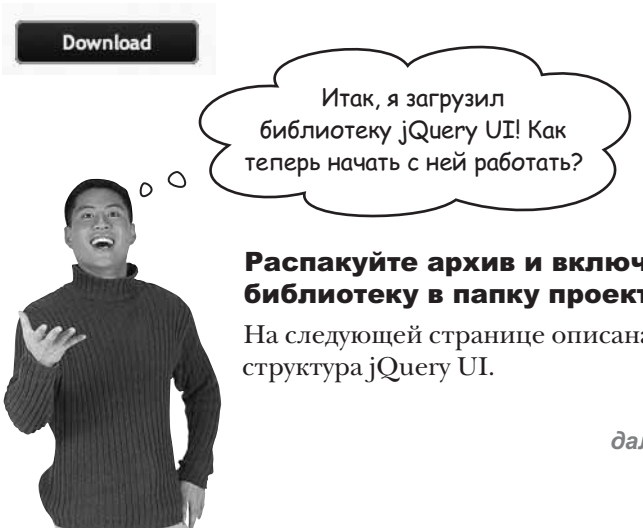
2 **Выберите загружаемые компоненты.**



3 **Выберите тему для загрузки.**

Одно из главных достоинств jQuery UI — темы оформления. Группа разработчиков jQuery UI включила весь код CSS для создания интерфейса профессионального уровня. Вы даже можете создать собственную тему оформления при помощи приложения jQuery UI «theme roller». Галерею всех тем jQuery UI можно просмотреть по URL-адресу http://jqueryui.com/themeroller/#themeGallery

4 **Нажмите кнопку Download.**

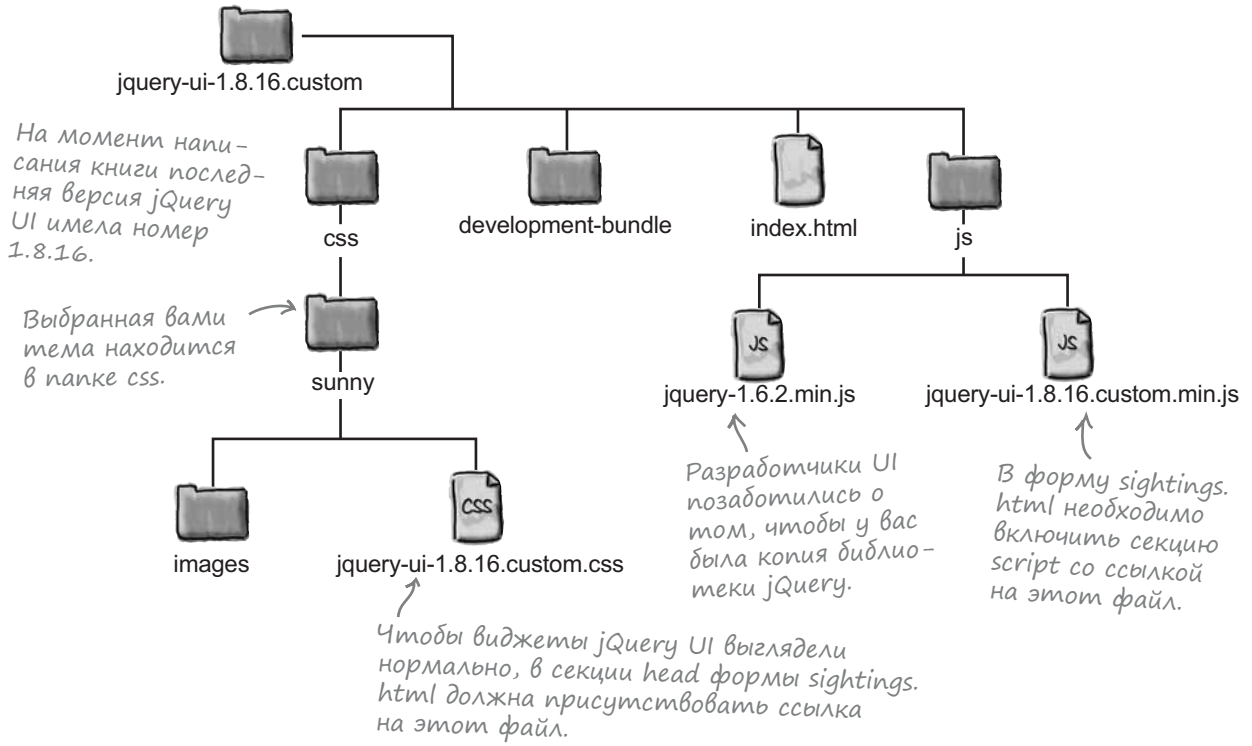


Распакуйте архив и включите библиотеку в папку проекта.

На следующей странице описана структура jQuery UI.

Содержимое пакета jQuery UI

После загрузки и распаковки jQuery вы увидите, что пакет имеет следующую структуру:



Мы включили папку jQuery UI в архив кода, загруженный вами в начале книги. Вы найдете ее в папке *end* внутри папки *ch10*.

Список задач проекта

jQuery UI делает многое за вас, но для создания новой формы нам придется решить ряд промежуточных задач. Перед вами контрольный список того, что мы должны сделать.

- [ ] 1. Создать календарь для выбора даты наблюдения.
- [ ] 2. Создать более привлекательные кнопки-переключатели для выбора типа существа.
- [ ] 3. Создать ползунки для ввода расстояния, веса и роста существа, широты и долготы места наблюдения.
- [ ] 4. Создать компонент для выбора цвета существа.
- [ ] 5. Создать более симпатичную кнопку отправки данных.

Построение календаря

Вы не поверите, как легко включаются виджеты jQuery UI в форму HTML.
Начнем с календаря:

1. **Создайте ссылку на CSS-файл jQuery UI:**

```
<link type="text/css" href="jquery-ui-1.8.16.custom/css/sunny/
jquery-ui-1.8.16.custom.css" rel="stylesheet" />
```

2. **Создайте тег <script> со ссылкой на jQuery UI.**

```
<script src="jquery-ui-1.8.16.custom/js/jquery-ui-1.8.16.
custom.min.js"></script>
```

3. **Возьмите обычное поле HTML `input`.**

```
<input type="text" name="sighting_date">
```

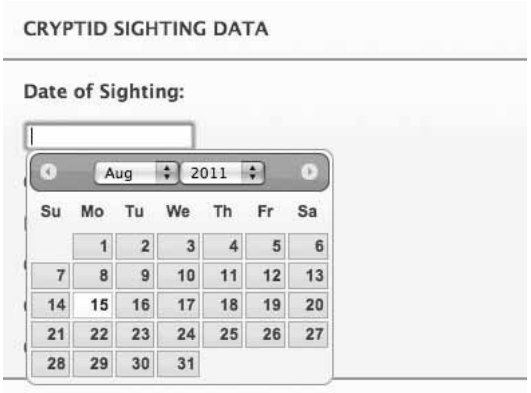
4. **Включите в тег <input> атрибут id со значением "datepicker".**

```
<input type="text" name="sighting_date" id="datepicker">
```

5. **Создайте файл JavaScript и включите следующий фрагмент между фигурными скобками `$(document).ready(function(){}`.**

```
$('#datepicker').datepicker();
```

6. **Откройте файл в своем любимом браузере и щелкните на поле `input`.**



Вот и все! Вы только что добавили на форму интерактивный виджет.

Незаметное вмешательство jQuery UI

На первый взгляд происходящее кажется каким-то волшебством, но в действительности jQuery UI представляет собой хорошо спроектированный и написанный блок кода, который не придется писать *вам*. Давайте разберемся в том, как он работает.

1 **Как и любой другой код jQuery, написанный вами, datepicker использует селектор и метод.**

```
$("#datepicker").datepicker();
```

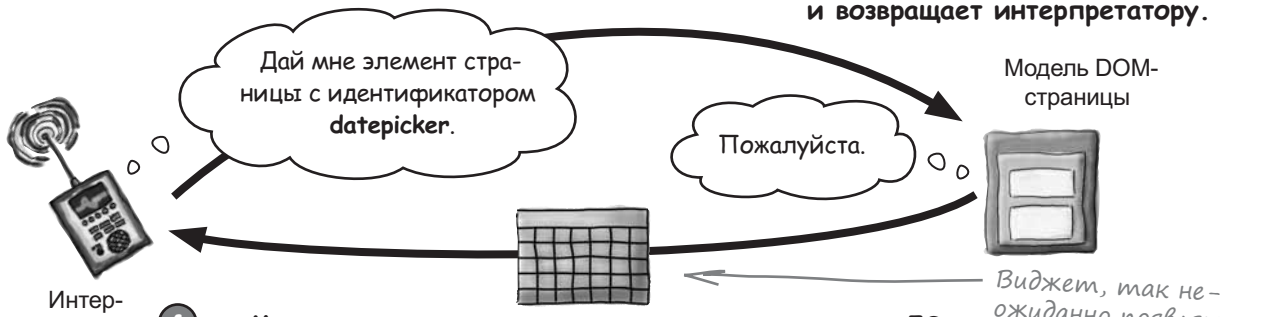
Основные события происходят здесь, в методе datepicker.

Код jQuery

За сценой

2 **Интерпретатор JavaScript запрашивает у DOM элемент с идентификатором `datepicker`.**

3 **DOM получает выбранный элемент, выполняет для него элемент `datepicker` и возвращает интерпретатору.**



Виджет, так неожиданно появляющийся на странице, представляет собой таблицу, вложенную в несколько областей div.

4 **Метод `datepicker` приказывает интерпретатору JS построить календарь «на ходу», вместе со всем кодом HTML, CSS и встроенной интерактивностью.**

Здорово, все происходит само собой, а мне достаточно написать немного кода HTML и jQuery. Но я вынуждена использовать конкретный вариант оформления и функций календаря? А если я захочу что-нибудь другое?

Не беспокойтесь, у вас есть выбор.

Давайте посмотрим, что можно сделать.

Изменение параметров виджета

Немного повозившись с изучением виджета datepicker, вы увидите, что он обладает многочисленными дополнительными возможностями, которые вы можете настроить по своему усмотрению.

Настройка виджета datepicker

Так как библиотека jQuery UI построена на базе jQuery, вам не придется писать большой объем кода, чтобы приспособить виджет datepicker к вашим потребностям. На момент написания книги виджет datepicker поддерживал 46 разных настраиваемых параметров.

Виджет datepicker содержит множество настраиваемых параметров. Параметр stepMonths определяет расстояние перехода в месяцах.

```
$("#datepicker").datepicker({
      stepMonths: 3
});
```

Если в виджете открыт календарь на август, то кнопки перехода сдвинуты на три месяца вперед или назад соответственно.

```
$("#datepicker").datepicker({
      changeMonth: true
});
```

Если задать параметру changeMonth значение true, то пользователь сможет выбрать месяц в раскрывающемся списке.

Упражнение

Напишите фрагмент кода, который позволит пользователю виджета datepicker выбрать и год, и месяц в раскрывающемся списке. Подсказка: если вы задаете значения нескольких параметров, разделите их запятыми.

.....

.....

.....

Упражнение
Решение

Напишите фрагмент кода, который позволит пользователю виджета datepicker выбрать и год, и месяц в раскрывающемся списке. Подсказка: если вы задаете значения нескольких параметров, разделите их запятыми.

```
$('#datepicker').datepicker({
        changeMonth: true, changeYear: true
});
```

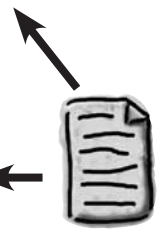
Готово к употреблению

Найдите файл *sightings_begin.html* в папке *begin* внутри папки *ch10*. Сохраните его под именем *sightings_end.html* в папке *end* главы 10. Включите код, выделенный ниже жирным шрифтом, в файлы *sightings_end.html* и *my_scripts.js*.

В начале файла sightings_end.html

```
<head>
    <title>Submit Your Cryptid Sighting</title>
    <link rel="stylesheet" type="text/css" href="style/form.css" />
  <link type="text/css" href="jquery-ui-1.8.16.custom/css/sunny/jquery-ui-1.8.16.custom.
css" rel="stylesheet" />
</head>
```

Чтобы виджеты выглядели так, как положено, необходимо подключить CSS-файл jQuery UI.



sightings_end.html

```
<h3>Date of Sighting:</h3>
      <input  type="text" name="sighting_date" id="datepicker" />
```

```
      <script src="scripts/jquery-1.6.2.min.js"></script>
      <script src="scripts/my_scripts.js"></script>
      <script src="jquery-ui-1.8.16.custom/js/jquery-ui-1.8.16.custom.min.js"></
script>
      </body>
</html>
```

Включаем библиотеку jQuery UI, чтобы пользоваться ее классными возможностями пользовательского интерфейса.

Ближе к концу файла sightings_end.html

```
$(document).ready(function(){
      $('#datepicker').datepicker({ changeMonth: true, changeYear: true});
});//end doc ready
```

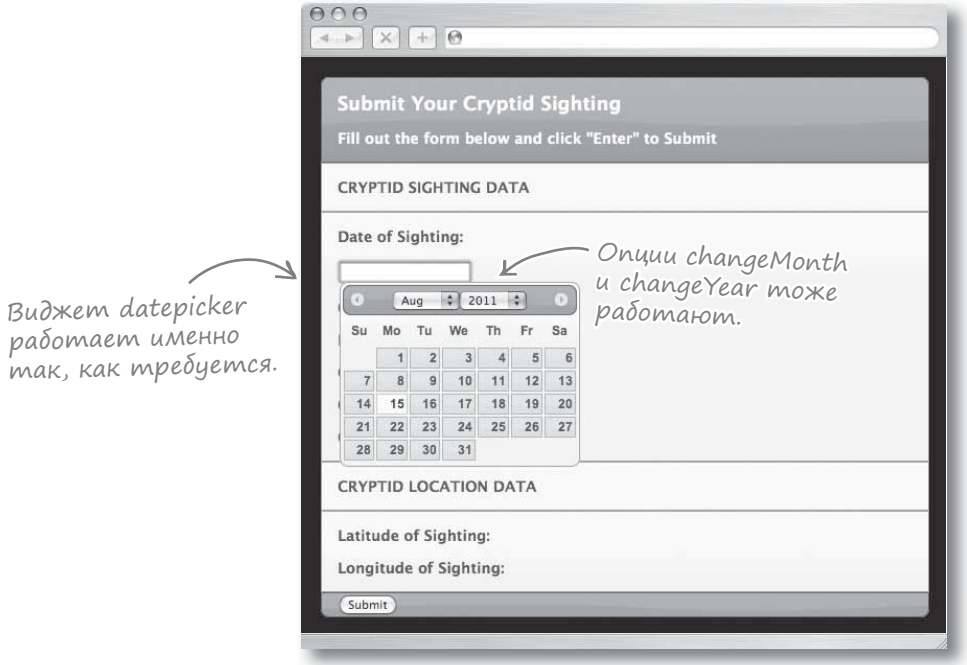
Код datepicker

my_scripts.js

Тест-драйв

Введите код с предыдущей страницы, откройте файл *sightings_end.html* в своем любимом браузере и протестируйте виджет. Пощелкайте на кнопках «вперед» и «назад», откройте списки месяца и года и убедитесь в том, что все работает нормально.

Календарь работает!



Вычеркиваем...

Пункт 1 выполнен, переходим к пункту 2.

- [x] 1. Создать календарь для выбора даты наблюдения.
- [ ] 2. Создать более привлекательные кнопки-переключатели для выбора типа существа.
- [ ] 3. Создать ползунки для ввода расстояния, веса и роста существа, широты и долготы места наблюдения.
- [ ] 4. Создать компонент для выбора цвета существа.
- [ ] 5. Создать более симпатичную кнопку отправки данных.

Стильные кнопки

Что значит «более привлекательные» применительно к кнопкам? В основном это вопрос стиля: если кнопка будет хорошо смотреться, пользователю захочется щелкнуть на ней. В библиотеку jQuery UI входит чрезвычайно полезный виджет button. Его метод `button` помогает создавать более привлекательные элементы форм — кнопки отправки данных, переключатели и флажки.

Код HTML одной кнопки-переключателя

Этот элемент будет обновлен при щелчке пользователя.

```
<input type="radio" id="radio1" name="radio" />
<label for="radio1">Choice 1</label>
```

Виджет button при помощи стилевого оформления придает элементу label вид кнопки.

И соответствующая команда jQuery

```
$( "#radio1" ).button();
```

Метод button преобразует классическую кнопку-переключатель HTML в эффектную, более интерактивную кнопку.

Не забудьте, что код разметки кнопок input должен находиться в теге HTML.

Группировка виджетов button

Для построения групп кнопок в jQuery UI существует метод `buttonset`, который преобразует отдельные элементы в группу по ссылке на контейнерный элемент группы.

Размещаем кнопки-переключатели внутри контейнерного элемента.

```
<div id="radio">
    <input type="radio" id="radio1" name="radio" />
        <label for="radio1">Choice 1</label>
    <input type="radio" id="radio2" name="radio" />
        <label for="radio2">Choice 2</label>
    <input type="radio" id="radio3" name="radio" />
        <label for="radio3">Choice 3</label>
</div>
```

В коде jQuery выбираем контейнерный элемент.

```
$( "#radio" ).buttonset();
```

Метод buttonset автоматически группирует кнопки и вызывает метод button для каждого элемента.

Развлечения с магнитами

Расставьте магниты в правильном порядке, чтобы сложить из них код набора кнопок для выбора типа существа. Мы разместили несколько фрагментов за вас.

sightings_end.html

```
<div id="type_select">
<input type="radio" id="radio1" name="creature_type" />
<label for="radio1">Chupacabras</label>
```

my_scripts.js

```
<input type="radio" id="radio2" name="creature_type" />
</div>
<label for="radio3">Loch Ness Monster</label>
$( "#type_select" ).buttonset();
<label for="radio4">Sasquatch</label>
<label for="radio2">Jersey Devil</label>
<input type="radio" id="radio4" name="creature_type" />
<input type="radio" id="radio3" name="creature_type" />
```

Развлечения с магнитами. Решение

Мы создали набор кнопок-переключателей, соответствующих общей теме формы.

```
<div id="type_select">
  <input type="radio" id="radio1" name="creature_type" />
  <label for="radio1">Chupacabras</label>
  <input type="radio" id="radio2" name="creature_type" />
  <label for="radio2">Jersey Devil</label>
  <input type="radio" id="radio3" name="creature_type" />
  <label for="radio3">Loch Ness Monster</label>
  <input type="radio" id="radio4" name="creature_type" />
  <label for="radio4">Sasquatch</label>
</div>
```

sightings_end.html

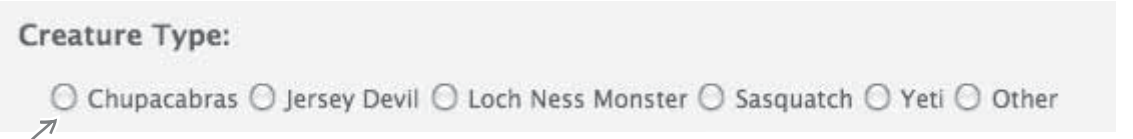
В папке code мы добавили остальные кнопки (Yeti и Other). Здесь они не приведены из-за нехватки места.

```
$( "#type_select" ).buttonset();
```

my_scripts.js

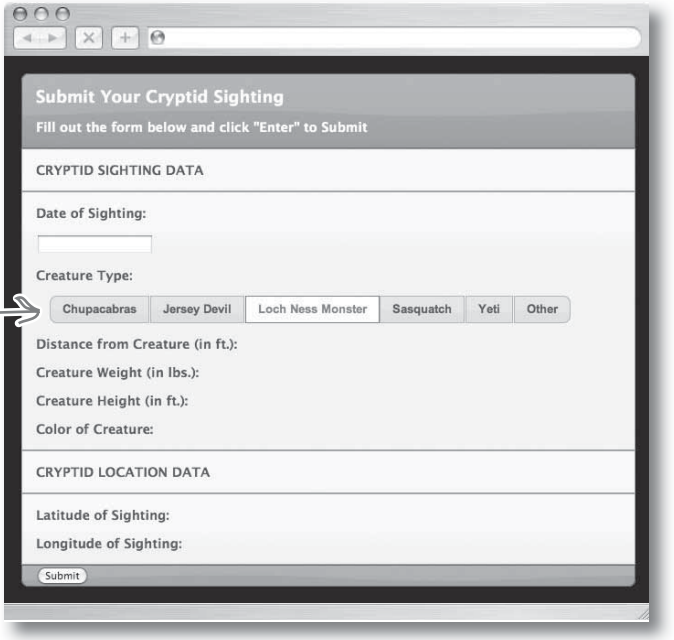
ТЕСТ-ДРАЙВ

Включите приведенные строки кода в файлы *sightings_end.html* и *my_scripts.js*. Затем откройте страницу в любимом браузере и убедитесь в том, что она работает правильно.



Берем обычные кнопки-переключатели HTML...

...и превращаем их в элегантный набор современных кнопок.



Как видите, ничего сложного. Что там следующее в списке?

- [x] 1. Создать календарь для выбора даты наблюдения.
- [x] 2. Создать более привлекательные кнопки-переключатели для выбора типа существа.
- [ ] 3. Создать ползунки для ввода расстояния, веса и роста существа, широты и долготы места наблюдения.
- [ ] 4. Создать компонент для выбора цвета существа.
- [ ] 5. Создать более симпатичную кнопку отправки данных.

Ограничение ввода числовых данных

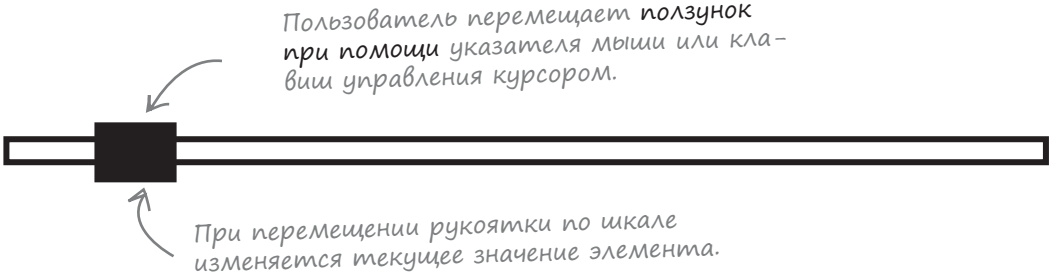
Расширения jQuery UI позволяют использовать для ввода данных ползунки, которыми пользователь управляет при помощи мыши или клавиатуры. Эти элементы также помогают контролировать диапазон вводимых чисел. Как вы уже видели, при наличии библиотеки jQuery UI построить виджет для работы с календарем проще простого. С ползунком (виджет slider) дело обстоит ничуть не сложнее.

Создание ползунка в файле HTML

```
<div id="slide_me"></div>
```

Сопутствующий код jQuery

```
$("#slide_me"). slider();
```



Ползунки также поддерживают многочисленные возможности настройки. Предположим, пользователь должен ввести набор чисел. Наименьшее допустимое значение равно 0, а наибольшее 100. Кроме того, пользователи должны вводить числа с приращением 5. Соответствующие значения параметров виджета slider выглядят так:

```
$( "#slide_me" ).slider({
value:0,
min: 0,
max: 100,
step: 5,
orientation: 'vertical'
});
```

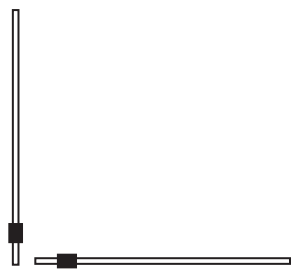
Параметр **value** *задает начальное значение ползунка.*

Параметр **min** *задает наименьшее значение, которое может быть введено пользователем.*

Параметр **max** *задает наибольшее значение, которое может быть введено пользователем.*

Параметр **step** *задает величину приращения.*

Параметр **orientation** *определяет ориентацию виджета (горизонтальная или вертикальная).*



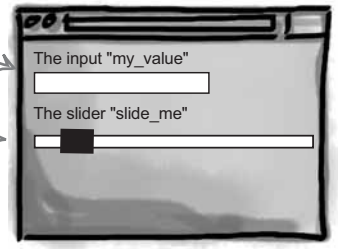
Нужно связать ползунок с одним из обработчиков событий виджета slider.

Мы уже видели, как задаются параметры виджета, но не рассматривали другую, полезную возможность jQuery UI. Многие компоненты jQuery поддерживают обработчики событий, ползунок не является исключением. На момент написания книги виджет slider из библиотеки jQuery UI поддерживал пять обработчиков событий: `create`, `start`, `slide`, `change` и `stop`. Чтобы связать ползунок с полем ввода формы, воспользуемся обработчиком события `slide`.

Разметка HTML для виджета slider

Чтобы запретить пользователю вводить числа, используйте значение «readonly».

```
<input type="text" id="my_value" readonly="readonly"/>
      <div id="slide_me"></div>
```



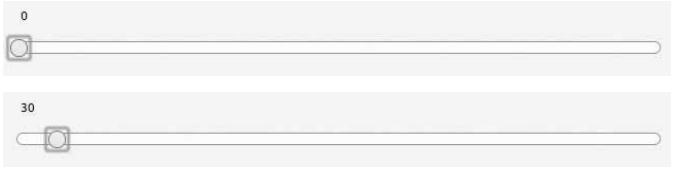
Сценарный код jQuery для виджета slider

Это обработчик события slide. Событие slide инициируется при перемещении ползунка.

Событие slide связывается с функцией обратного вызова. При выполнении эта функция задает значение поля ввода методом jQuery val.

```
$( "#slide_me" ).slider({
      slide: function( event, ui ) {
            $( "#my_value" ).val( ui.value);
      }
});
$( "#my_value" ).val( $( "#slide_me" ).slider( "value" ));
```

Когда пользователь двигает ползунок, вызывается функция, а элемент input обновляется текущим значением ползунка.



Длинные упражнения

Заполните пропуски в коде полей input. Криптозоологи оставили замечания по поводу настройки ползунков.

Расстояние от существа (в футах):

Начальное значение = 0.

Минимальное расстояние = 0.

Максимальное расстояние = 500.

Приращение = 10 футов.

```
<h3>Distance from Creature (in ft.):</h3>
<input type="text" id="……." class="just_display" name="creature_distance"
                                                    readonly="readonly"/>
            <div id="……."></div>
      </div>
</div>
```

sightings_end.html

```
$( "#slide_dist" ).slider({
                    …….
                    …….
                    …….
                    …….
                     slide: function( event, ui ) {
                            $( "#distance" ……. )
                     }
});
```

my_scripts.js

Вес существа (в фунтах):

Начальное значение = 0.

Минимальный вес = 0.

Максимальный вес = 5000.

Приращение = 5 фунтов.

Рост существа (в футах):

Начальное значение = 0.

Минимальное значение = 0.

Максимальное значение= 20.

Приращение = 1 фут.

```
<h3>Creature Weight (in lbs.):</h3>
       <input  type="text" id="weight" class="just_display" name="creature_weight"
                                                            readonly="readonly"/>
                     <div id="           "></div>

<h3>Creature Height (in ft.):</h3>
       <input  type="text" id="height" class="just_display" name="creature_height"
                                                            readonly="readonly"/>
                     <div id="slide_height"></div>
```

sightings_end.html

```
$( "              " ).slider({
                        ..............
                        ..............
                        ..............
                        ..............
       slide: function( event, ui ) {
          $( "          " ).val( ui.value);
       }
});
```

my_scripts.js

```
$( "#slide_weight" ).slider({
                        ..............
                        ..............
                        ..............
                        ..............
                  ....................................
          $( "#weight" ).val( ui.value);
       }
});
```

my_scripts.js

Решение длинных упражнений

Заполните пропуски в коде полей input. Криптозоологи оставили свои замечания по поводу настройки ползунков.

Расстояние от существа (в футах):

Начальное значение = 0.

Минимальное расстояние = 0.

Максимальное расстояние = 500.

Приращение = 10 футов.

```
<h3>Distance from Creature (in ft.):</h3>
<input type="text" id="distance" class="just_display" name="creature_distance"
                                                    readonly="readonly"/>
            <div id="slide_dist"></div>
      </div>
</div>
```

sightings_end.html

```
$( "#slide_dist" ).slider({
                    value:0,
                    min:0,
                    max:500,
                    step: 10,
                     slide: function( event, ui ) {
                            $( "#distance".val( ui.value); )
                     }
});
```

my_scripts.js

Вес существа (в фунтах):

Начальное значение = 0.

Минимальный вес = 0.

Максимальный вес = 5000.

Приращение = 5 фунтов.

Рост существа (в футах):

Начальное значение = 0.

Минимальное значение = 0.

Максимальное значение= 20.

Приращение = 1 фут.

```
<h3>Creature Weight (in lbs.):</h3>
       <input  type="text" id="weight" class="just_display" name="creature_weight"
                                                          readonly="readonly"/>
                    <div id="slide_weight"></div>

<h3>Creature Height (in ft.):</h3>
       <input  type="text" id="height" class="just_display" name="creature_height"
                                                          readonly="readonly"/>
                    <div id="slide_height"></div>
```

sightings_end.html

```
$( "#slide_height" ).slider({
                    value:0,
                    min:0,
                    max:20,
                    step: 1,
      slide: function( event, ui ) {
         $( "#height" ).val( ui.value);
      }
});
```

my_scripts.js

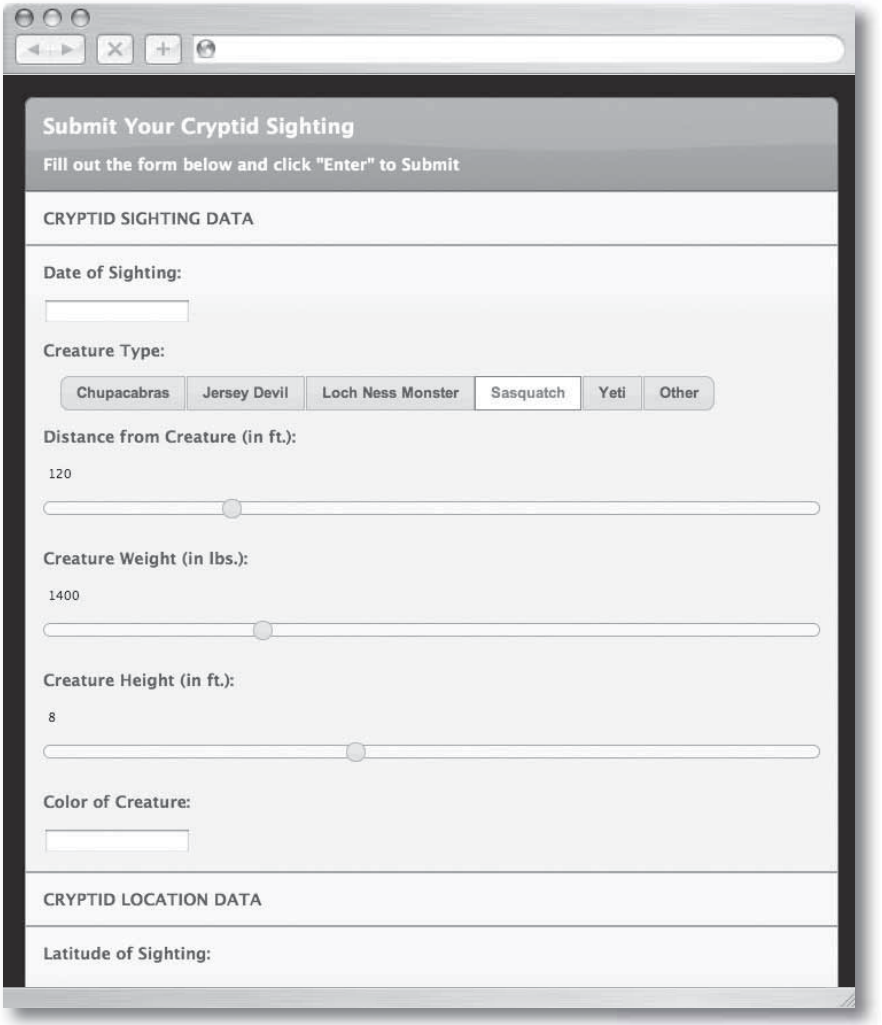
```
$( "#slide_weight" ).slider({
                    value:0,
                    min:0,
                    max:5000,
                    step: 10,
      slide: function( event, ui ) {
         $( "#weight" ).val( ui.value);
      }
});
```

my_scripts.js

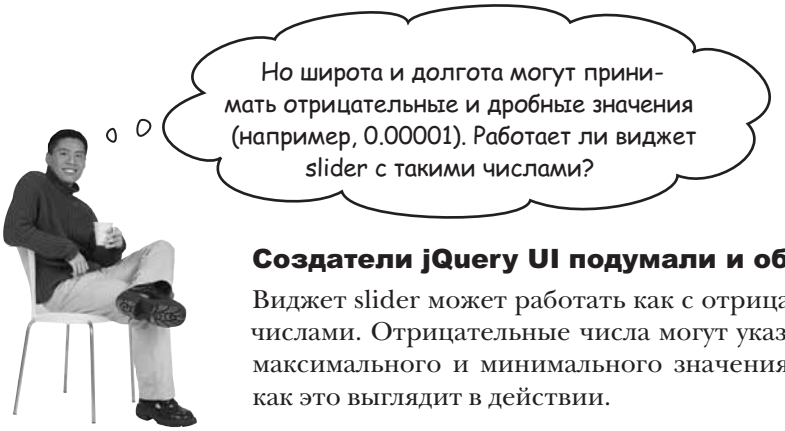
ТЕСТ-ДРАЙВ

Включите код из длинного упражнения на предыдущих страницах в свои файлы, после чего откройте файл *sightings_end.html* в своем любимом браузере. Проверьте, как работают ползунки, с использованием мыши и клавиатуры (клавиши со стрелками влево и вправо должны изменять состояние ползунка на одно приращение).

Теперь пользователи могут вводить данные, перемещая рукоятки ползунков, а криптозоологи могут лучше контролировать вводимые данные.



Осталось построить еще два числовых ползунка: для ввода широты и долготы... И мы сможем вычеркнуть еще один пункт в списке задач!



Создатели jQuery UI подумали и об этом.

Виджет slider может работать как с отрицательными, так и с дробными числами. Отрицательные числа могут указываться в качестве текущего, максимального и минимального значения. Попробуйте и посмотрите, как это выглядит в действии.

Возьми в руку карандаш

Заполните пропуски в коде jQuery, чтобы создать на форме ползунки для ввода широты и долготы. Ползунок широты должен работать в диапазоне от –90 до 90 с приращением 0.00001. Ползунок долготы должен работать в диапазоне от –180 до 180 с приращением 0.00001.

```
                    value:0,.....
.....

                    .............
                    .............
                slide: function( event, ui ) {
                        $( "          " ).val( ui.value);
                }               .............
        });


                    value:0,.....
.....

                    .............
                    .............
                slide: function( event, ui ) {
                        $( "          " ).val( ui.value);
                }           .............
        });
```

my_scripts.js

Возьми в руку карандаш
Решение

Заполните пропуски в коде jQuery, чтобы создать на форме ползунки для ввода широты и долготы. Ползунок широты должен работать в диапазоне от –90 до 90 с приращением 0,00001. Ползунок долготы должен работать в диапазоне от –180 до 180 с приращением 0,00001.

```
$( "#slide_lat" ).slider({
                    value:0,
                      min: -90,
                      max: 90,
                      step: 0.00001,
                    slide: function( event, ui ) {
                           $( "    latitude" ).val( ui.value);
                    }
              });

$( "#slide_long" ).slider({
                    value:0,
                      min: -180,
                      max: 180,
                      step: 0.00001,
                    slide: function( event, ui ) {
                           $( " longitude " ).val( ui.value);
                    }
              });
```

my_scripts.js

К настоящему моменту мы вычеркнули уже несколько пунктов списка задач.

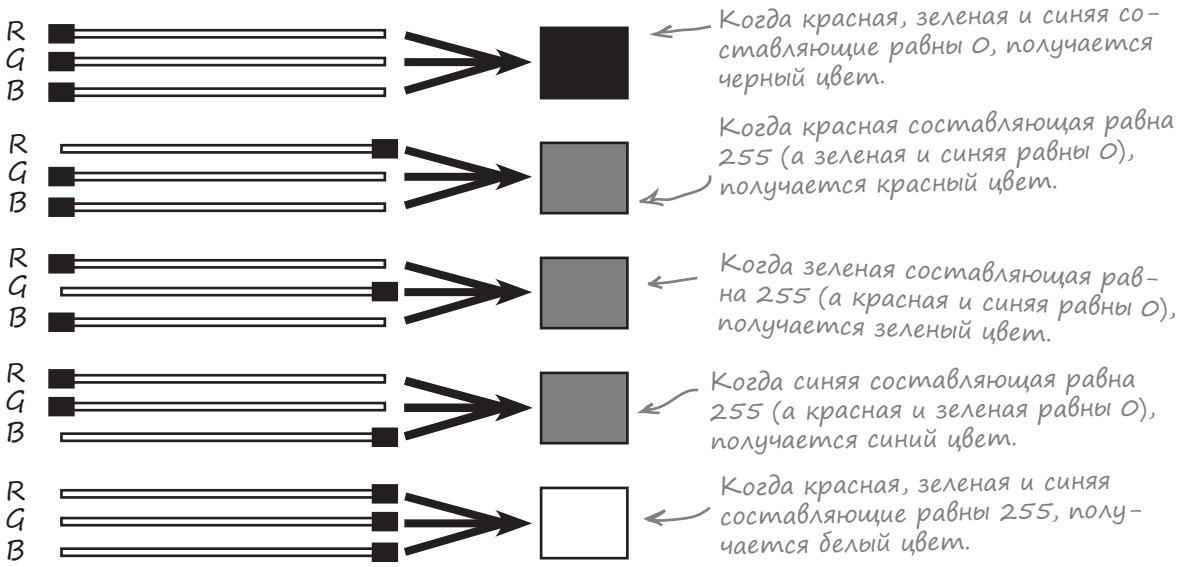
- [x] 1. Создать календарь для выбора даты наблюдения.
- [x] 2. Создать более привлекательные кнопки-переключатели для выбора типа существа.
- [x] 3. Создать ползунки для ввода расстояния, веса и роста существа, широты и долготы места наблюдения.

Что дальше? Нужно создать интерфейс для ввода цвета существа. Компонент определения цвета будет состоять из ползунков, соответствующих трем цветовым составляющим (красная, зеленая и синяя).

- [ ] 4. Создать компонент для выбора цвета существа.
- [ ] 5. Создать более симпатичную кнопку отправки данных.

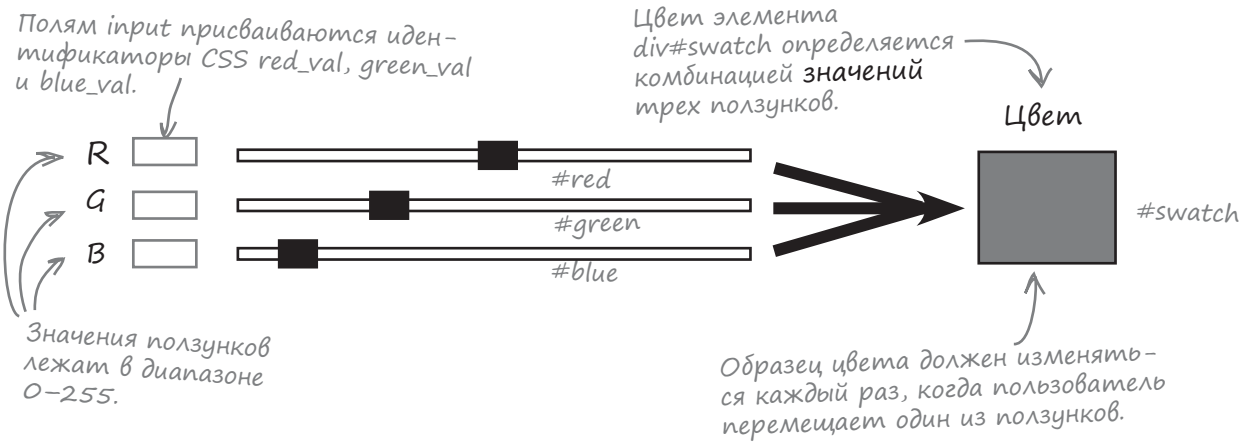
Создание цвета по трем составляющим

Значения интенсивности красного, зеленого и синего лежат в диапазоне от 0 до 255. Когда все три интенсивности находятся на минимуме (то есть равны 0), получается черный цвет. При максимальных значениях цветовых составляющих (все три составляющие равны 255) получается белый цвет.



Ползунки должны работать по тому же принципу

Итак, мы должны построить три ползунка для трех цветовых составляющих: красной, зеленой и синей. Затем текущие значения всех ползунков объединяются в один цвет. Давайте посмотрим, что должен делать каждый из трех виджетов slider.



Готово к употреблению

Включите код, выделенный жирным шрифтом, в файлы *sightings_end.html* и *form.css*, чтобы подготовить их к построению компонента выбора цвета.

```
//color slider styles
   #slide_dist, #slide_weight, #slide_height {
   margin-bottom:14px;
   }
   #swatch {
      width: 75px;
      height: 75px;
      background-image: none;
   }

   #red .ui-slider-range { background: #ef2929; }
   #red .ui-slider-handle { border-color: #ef2929; }

   #green .ui-slider-range { background: #8ae234; }
   #green .ui-slider-handle { border-color: #8ae234; }

   #blue .ui-slider-range { background: #729fcf; }
   #blue .ui-slider-handle { border-color: #729fcf; }
```

Этот идентификатор стиля CSS определяет образец, в котором будет отображаться текущий цвет при перемещении ползунка.

Эти стили используют цвет ползунка в его оформлении.

form.css

```
<h3>Color of Creature (use the color sliders to enter):</h3>

Color (in hexadecimal):<input  type="text" class="just_display" name="creature_color"
id="color_val" readonly="readonly"/><br /><br />
   <div id="swatch" class="ui-widget-content ui-corner-all"></div>

Red:<input  type="text" class="just_display" name="creature_color" id="red_val"
readonly="readonly"/>
   <div id="red"></div>

Green:<input type="text" class="just_display" name="creature_color" id="green_val"
readonly="readonly"/>
   <div id="green"></div>

Blue:<input type="text" class="just_display" name="creature_color" id="blue_val"
readonly="readonly"/>
   <div id="blue"></div>
```

Поле input, которое будет содержать шестнадцатеричное значение цвета

Область div для образца цвета

Область div для красного ползунка

Область div для зеленого ползунка

Область div для синего ползунка

sightings_end.html

Возьми в руку карандаш

Следующий фрагмент сценарного кода задает конфигурацию цветовых ползунков (красного, зеленого и синего). Прочитайте каждую строку и подумайте, что она может делать, на основании того, что вы уже знаете о jQuery и jQuery UI. Запишите справа свои описания. Если вы не уверены, запишите хотя бы предположения. Мы заполнили одну строку за вас.

```
$( "#red, #green, #blue" ).slider({
       orientation: "horizontal",
       range: "min",
       max: 255,
       value: 127,
       slide: refreshSwatch,
       change: refreshSwatch
});
```

Диапазон: пользователь может указывать только максимум.

Часто Задаваемые Вопросы

В: А что это за код CSS, прилагаемый к jQuery UI?

О: Одно из преимуществ jQuery UI в том, что разработчики уже написали много сложного кода CSS и вам не придется этим заниматься. Дополнительная информация об этом коде содержится в файле пакета jQuery UI *jquery-ui-1.8.16.custom/css/sunny/jquery-ui-1.8.16.custom.css*. За более подробными описаниями классов CSS обращайтесь к документации jQuery UI по адресу *http://jqueryui.com/docs/Theming/API*.

В: Вы сказали, что мы можем создать собственную тему для jQuery UI. Как это сделать?

О: Используйте приложение Theme Roller для jQuery UI. Сначала откройте страницу приложения в браузере: http://jqueryui.com/themeroller/

Щелкните на вкладке Roll your own. Здесь определяются такие параметры конфигурации, как шрифты, состояния нажатия кнопок, тени и т. д. Задайте нужные изменения, и лементы пользовательского интерфейса изменяется в соответствии с заданными значениями.

Когда тема будет настроена так, как вам нужно, щелкните на кнопке Download Theme. Открывается страница Build Your Download, на которой создается пользовательский пакет jQuery UI. Если вы захотите сохранить свою тему на будущее, просто создайте закладку для страницы Build Your Download.

В: Не пойму, что это за виджеты взаимодействий. Для чего они нужны?

О: Взаимодействия используются для реализации интерактивных функций, свойственных настольным приложениям.

Виджеты Draggable обеспечивают возможность перетаскивания элементов.

Виджет Droppable может использоваться в качестве приемника для перетаскиваемых элементов.

Виджет Resizable включает возможность масштабирования элемента. Размеры элемента изменяются перетаскиванием угла, правой или нижней стороны.

Виджеты Selectable обеспечивают поддержку выделения элементов. Посетитель сайта может захватить эти элементы мышью, как при выделении группы файлов на рабочем столе.

Виджеты Sortable могут переставляться пользователем в интерактивном режиме.

Возьми в руку карандаш
Решение

Следующий фрагмент сценарного кода задает конфигурацию цветовых ползунков (красного, зеленого и синего). Прочитайте каждую строку и подумайте, что она может делать, используя ваши знания о jQuery и jQuery UI. Запишите справа свои описания. Если вы сомневаетесь, запишите ваши предположения. Ниже приведены наши ответы.

```
$( "#red, #green, #blue" ).slider({   Преобразует области div в виджеты slider.
      orientation: "horizontal",      Выбирает горизонтальную ориентацию ползунка.
      range: "min",                   Диапазон: пользователь может указывать только максимум.
      max: 255,                       Максимальное значение 255 (ограничения цветовой модели).
      value: 127,                     Маркер ползунка устанавливается в середине.
      slide: refreshSwatch,           При перемещении ползунка вызывается функция refreshSwatch.
      change: refreshSwatch           Эта функция вызывается при изменении любого значения.
});
```

Функция refreshSwatch

Чтобы завершить построение цветового компонента, необходимо создать функцию JavaScript для обновления образца цвета. Ниже приведена заготовка функции, а также некоторые ключевые вопросы, над которыми следует подумать до написания кода.

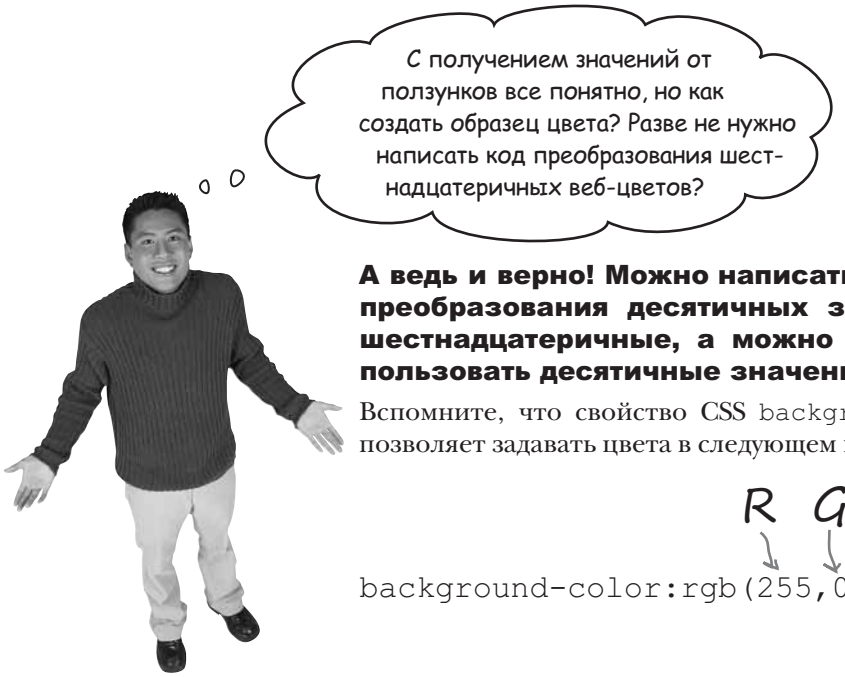
```
function refreshSwatch() {
      var red = ???
      var green = ???
      var blue = ???
      var my_rgb = ???
      $( "#swatch" ).???;
      $( "#red_val" ).val(red );
      $( "#blue_val" ).val( blue);
      $( "#green_val" ).val( green);
      $( "#color_val" ).val(my_rgb);
}
```

Как перенести текущие значения ползунков в эти переменные?

В этой переменной следует объединить значения RGB для задания цвета образца.

Какой метод jQuery позволит нам задать цвет образца?

Ничего сложного. Достаточно просто воспользоваться методом jQuery val, чтобы в полях input отображалось значение ползунков в процессе изменения. Пользователь будет видеть, какое значение установлено на ползунке.



А ведь и верно! Можно написать функцию преобразования десятичных значений в шестнадцатеричные, а можно просто использовать десятичные значения.

Вспомните, что свойство CSS `background-color` позволяет задавать цвета в следующем виде:

R G B

```
background-color:rgb(255,0,255)
```

Впрочем, это лишь подсказка для одного из вопросов. Чтобы написать всю функцию, нужно будет немного поработать мозгами.

Упражнение

Заполните пропуски в коде функции `refreshSwatch`.

```
function refreshSwatch() {
      var red = ..........
      var green = ..........
      var blue = ..........
      var my_rgb = ..........
      $( "#swatch" )...........;
      $( "#red_val" ).val(red );
      $( "#blue_val" ).val( blue);
      $( "#green_val" ).val( green);
      $( "#color_val" ).val(my_rgb);
}
```

Упражнение
Решение

Заполните пропуски в коде функции `refreshSwatch`.

```
function refreshSwatch() {
      var red =  $( "#red" ).slider( "value" );
      var green =  $( "#green" ).slider( "value" );
      var blue =   $( "#blue" ).slider( "value" );
      var my_rgb =  "rgb(" + red + "," + green + "," + blue + ")";
      $( "#swatch" ). $( "#swatch" ).css( "background-color", my_rgb );
      $( "#red_val" ).val(red );
      $( "#blue_val" ).val( blue);
      $( "#green_val" ).val( green);
      $( "#color_val" ).val(my_rgb);
}
```

Объединяя значения RGB в этой переменной...

...мы задаем в CSS образца комбинацию значений трех цветов.

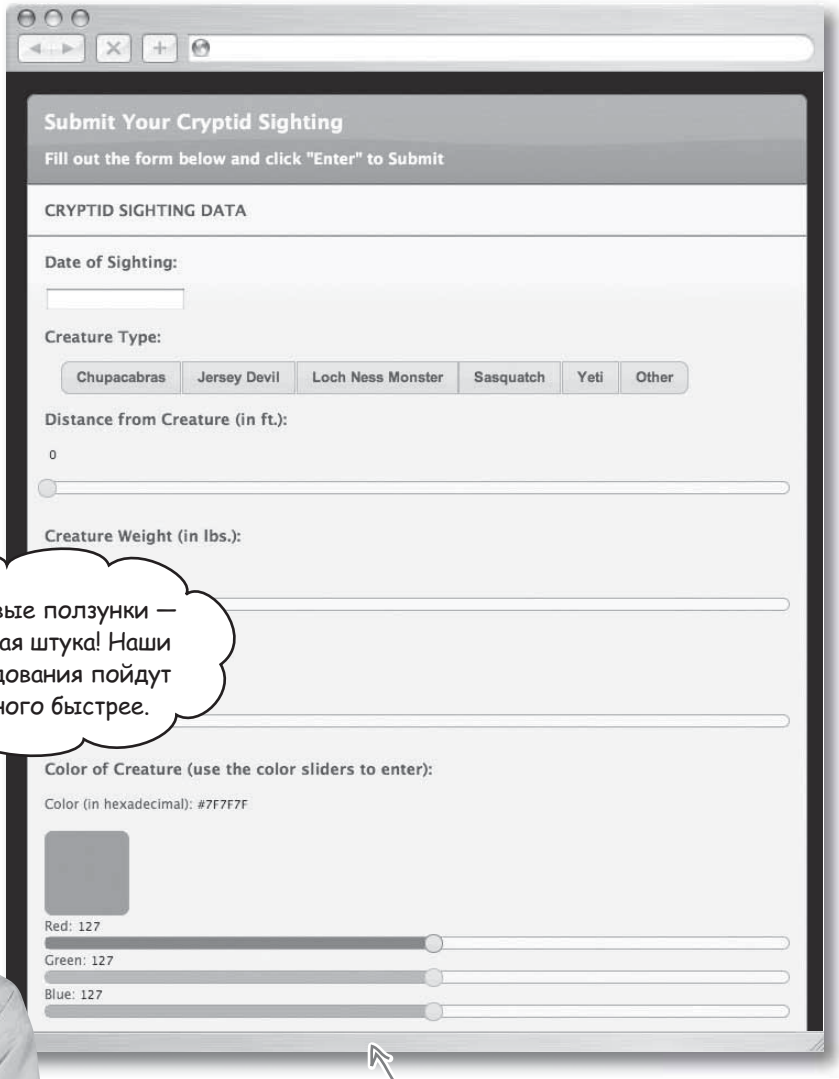
Готово
к употреблению

Включите в файл *my_scripts.js* код ползунков и функции `refreshSwatch`. Также добавьте фрагмент, приведенный ниже. Он инициирует выполнение функции `refreshSwatch` при загрузке веб-страницы, чтобы страница загружалась с образцом цвета вместо пустого поля.

```
$( "#red" ).slider( "value", 127 );
$( "#green" ).slider( "value", 127 );
$( "#blue" ).slider( "value", 127 );
```

Тест-драйв

Откройте файл *sightings_end.html* в любимом браузере. Проверьте, как работают ползунки, с использованием мыши и клавиатуры (клавиши со стрелками влево и вправо должны изменять состояние ползунка на одно приращение).



Теперь пользователи могут вводить данные, перемещая рукоятки ползунков, а криптозоологи смогут получать более качественные данные.

И последнее...

В нашем списке задач остался последний пункт.

1. ☑ *Создать календарь для выбора даты наблюдения.*
2. ☑ *Создать более привлекательные кнопки-переключатели для выбора типа существа.*
3. ☑ *Создать ползунки для ввода расстояния, веса и роста существа, широты и долготы места наблюдения.*
4. ☑ *Создать компонент для выбора цвета существа.*
5. ☐ *Создать более симпатичную кнопку отправки данных.*

Мы уже работали с кнопками в этой главе, а вы использовали их начиная с главы 1, так что эта задача должна быть простой!

Часто Задаваемые Вопросы

В: Мы использовали запись `RGB` для свойства `background-color` построенных нами цветовых ползунков. А если мы захотим создать ползунок, использующий стандартные шестнадцатеричные обозначения цветов?

О: Мы постарались создать самый элегантный и простой код цветового ползунка. Но если вам нужен ползунок с вводом данных в шестнадцатеричном формате, считайте, что вам повезло. Вы можете использовать пример кода, размещенный на сайте jQuery UI. Откройте в браузере следующий URL-адрес:

http://jqueryui.com/demos/slider/#colorpicker

Найдите и откройте ссылку View Source. Выделите и скопируйте код HTML, CSS и jQuery в текстовом поле, сохраните его в новом документе. Не забудьте включить этот документ в соответствующие файлы CSS и jQuery своего пакета jQuery. Вот и все, у вас появился компонент выбора цвета, использующий шестнадцатеричные коды цветов.

В: Мне нужно поле формы, которое предлагает возможные условия поиска в процессе ввода. В jQuery UI есть что-нибудь подобное?

О: Да! Одним из новшеств jQuery UI на момент написания книги (август 2011 года) стал виджет Autocomplete, который предлагает варианты во время ввода данных в поле формы.

Откуда берутся эти варианты? Вы передаете их в массиве JavaScript, через URL-адрес или при помощи функции обратного вызова, которая получает данные с сервера при помощи методов Ajax, описанных в главах 8 и 9. За дополнительной информацией об этом виджете обращайтесь на страницу демонстрационного приложения jQuery UI:

http://jqueryui.com/demos/autocomplete/

В: jQuery UI поддерживает проверку данных форм?

О: Нет. К сожалению, jQuery UI не поддерживает проверку форм, но в jQuery есть модуль расширения, который хорошо справляется с этой задачей. Модуль расширения можно найти по адресу:

http://bassistance.de/jquery-plugins/jquery-plugin-validation/

За дополнительной информацией обращайтесь к приложению I, в котором этот модуль расширения рассматривается более подробно.

Верное замечание.

Мы использовали метод jQuery `click` и метод jQuery UI `button`, но не занимались *выбором элементов форм*, например кнопок отправки данных. Ниже приведена краткая справка по выбору таких элементов.

КЛЮЧЕВЫЕ МОМЕНТЫ

- `$(":input")` = Выбор всех элементов input
- `$(":text")` = Выбор всех элементов типа `text`
- `$(":radio")` = Выбор всех элементов типа `radio`
- `$(":checkbox")` = Выбор всех элементов типа `checkbox`
- `$(":submit")` = Выбор всех элементов типа `submit`
- `$(":reset")` = Выбор всех элементов типа `reset`
- `$(":checked")` = Выбор всех полей input в состоянии checked
- `$(":selected")` = Выбор всех полей input в состоянии selected
- `$(":enabled")` = Выбор всех незаблокированных полей input
- `$(":disabled")` = Выбор всех заблокированных полей input
- `$(":password")` = Выбор всех полей input, предназначенных для ввода пароля

Упражнение

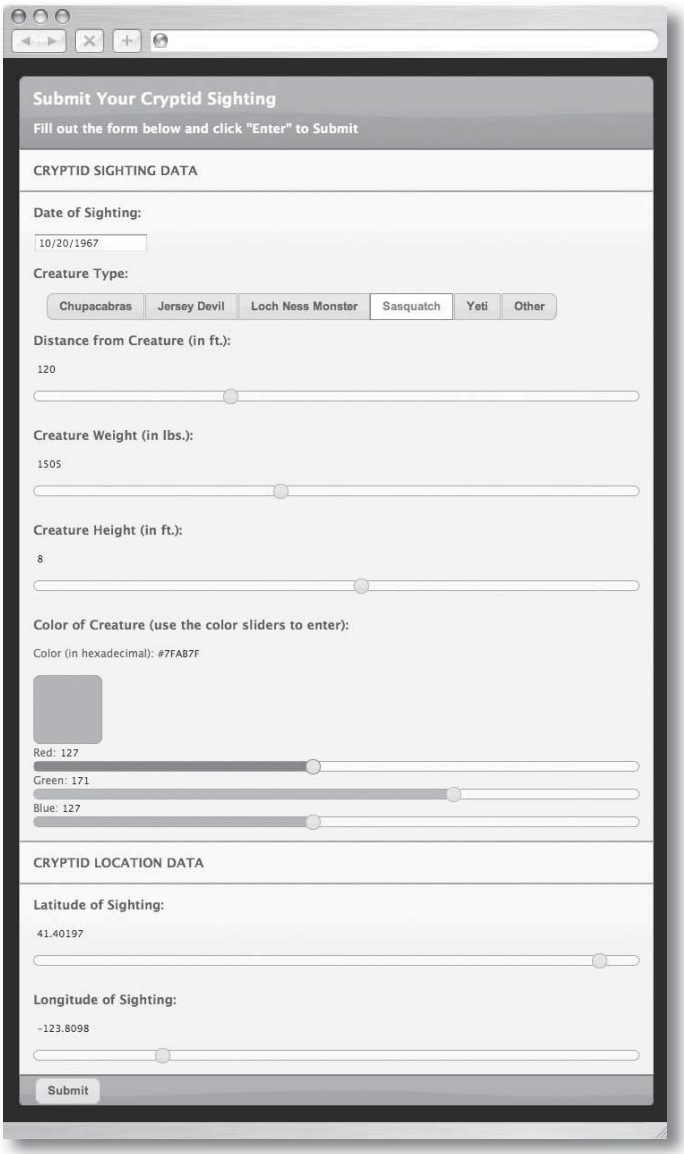
Напишите одну строку кода, которую следует добавить в файл *my_scripts.js* для преобразования «классической» кнопки в кнопку jQuery UI, соответствующей теме оформления.

.....

Упражнение
Решение

Не забудьте включить эту строку в файл *my_scripts.js* и протестировать ее, открыв файл *sightings_end.html* в своем любимом браузере.

```
$( "button:submit" ).button();
```



Упражнение

Дополнительный материал

Форма отлично смотрится, но на данный момент она не *отправляет* никакие данные. Впрочем, вы уже узнали все необходимое в главе 9. Пора подумать, как сделать стильную форму полностью функциональной.

Папка *end* в архиве кода этой главы содержит все файлы, необходимые для работы: *sightings.html*, *service.php* и *sightings.sql*. Конечно, вам придется проделать часть работы по настройке самостоятельно, но ведь в этом и заключается задача веб-разработчика, не так ли? Мы включили методы Ajax и JSON, описанные в главе 9, чтобы построенная вами форма могла отправлять данные.

Всю необходимую подготовку вы должны провести самостоятельно (т. е. запустить сценарий *sightings.sql* или же создать базу данных с полями из сценария самостоятельно и включить методы Ajax и JSON в файл *my_scripts.js*). Также до перехода к главе 11 в базе данных необходимо создать несколько записей. Если вам понадобится освежить в памяти информацию о MySQL, PHP и AJAX, обращайтесь к главам 8 и 9.

У этого упражнения нет решения. Если у вас возникнут проблемы, вы всегда можете задать их форуму книги по адресу www.headfirstlabs.com и пообщаться с авторами или другими читателями.

ГЛАВА 10

Ваш инструментарий jQuery

Глава 10 осталась позади. Ваш творческий потенциал дополнился навыками использования jQuery UI.

jQuery UI

Официальная библиотека jQuery с тремя основными типами расширений для базовой функциональности jQuery: эффекты, взаимодействия и виджеты.

Виджет

Компонент, расширяющий функциональность веб-приложения.

Экономит время при программировании, упрощает создание удобных, практичных элементов пользовательского интерфейса.

Виджет button

Предоставляет метод button для создания более привлекательных элементов форм — кнопок отправки данных, переключателей и флажков.

Виджет datepicker

Метод datepicker приказывает интерпретатору JS построить календарь «на ходу» — вместе со всем кодом HTML, CSS и встроенной интерактивностью.

Поддерживает большое количество настраиваемых параметров.

Ползунки

Элементы пользовательского интерфейса, которыми можно управлять как мышью, так и клавиатурой; ограничивают набор вводимых данных.

Поддерживают пять обработчиков событий, которые могут использоваться для связывания ползунка с полем input формы: create, start, slide, change и stop.

11 jQuery и API

Объекты, сплошные объекты

Даже самый талантливый разработчик не сможет сделать всю работу в одиночку... Мы уже видели, как включать расширения jQuery (такие как jQuery UI или вкладки) для повышения уровня функциональности приложения. Чтобы поднять наше приложение на следующий уровень — использовать Интернет и информацию из крупных информационных систем типа Google, Twitter или Yahoo! — понадобится... нечто большее. Эти компании предоставляют вам программные интерфейсы (API, Application Programming Interface). В этой главе мы рассмотрим основы работы с API, а также используем очень распространенный сервис Google Maps API.

Где видели снежного человека?

Доктор Паттерсби и доктор Гимли хотят добавить на свой сайт новые классные возможности. Как думаете, вы справитесь с этой задачей?

От: **Доктор Гимли [gimli@cryptozoologists.org]**
Тема: **Обновления на сайте**

Привет,

Спасибо вам за то, что помогли сделать наш сайт более удобным и упростили сбор данных о наблюдениях. Трафик заметно вырос, и мы едва справляемся с обработкой всех собранных данных.

У нас есть несколько пожеланий для того, чтобы сделать информацию более доступной для широких масс. Многие люди интересуются данными наблюдений, и мы бы хотели предоставить им возможность собранных данных.

Вот что нам нужно:

1) нужно предоставить пользователю возможность просмотреть данные отдельного наблюдения и всю информацию, связанную с ним. Наряду с информацией о существе нам хотелось бы видеть информацию на карте Google;

2) выбирая маркер на карте Google, пользователь получает дополнительную информацию о наблюдении;

3) нам хотелось бы, чтобы при выборе типа существа из списка отображалась информация обо всех существах, связанных с выбранным типом в нашей базе данных. При этом информация обо всех существах должна отображаться на карте Google, чтобы мы могли найти места частых наблюдений для более внимательного изучения. Можно ли сделать так, чтобы все эти точки тоже реагировали на щелчки для получения дополнительной информации (как и пункты списка существ)?

Надеюсь, мы хотим не слишком многого, ведь вся информация уже хранится в базе данных?

Ждем ответа!

--
Доктор Гимли и доктор Паттерсби
cryptozoologists.org

Джо: Ты говоришь о карте Google, которую нам предстоит создать?

Фрэнк: Ага. Тривиально, в общем.

Джим: Тривиально? Они хотят полноценную карту Google!

Фрэнк: Угу.

Джим: С несколькими маркерами на карте — по одному для каждого криптида. И эти маркеры должны реагировать на щелчки выводом дополнительной информации.

Фрэнк: Да! И я, кажется, знаю, как это делать.

Джим: А еще щелчки в списке должны взаимодействовать с маркерами — на карте должны появляться временные окна с дополнительной информацией.

Фрэнк: Это уже сложнее... Я пока не знаю, как это сделать.

Джо: Беспокоиться не о чем. Разработчики Google Maps уже предоставили API, с которым мы все сделаем.

Джим: Чего предоставили?

Фрэнк: API... Программный интерфейс, короче. Так, компании типа Google, Netflix и Yahoo! дают нам возможность использовать свой сервис на наших сайтах.

Джим: Это хорошо, конечно, но есть ли в нем временные окна и все остальное, что нужно выводить на карте?

Джо: *Должно* быть. Наверное, нам стоит поближе познакомиться с интерфейсом Google Maps API и понять, как он работает.

Google Maps API

Использование любого API на вашем сайте начинается с поиска документации и примеров кода в Интернете. Мы загрузили этот пример по адресу *http://code.google.com/apis/maps*.

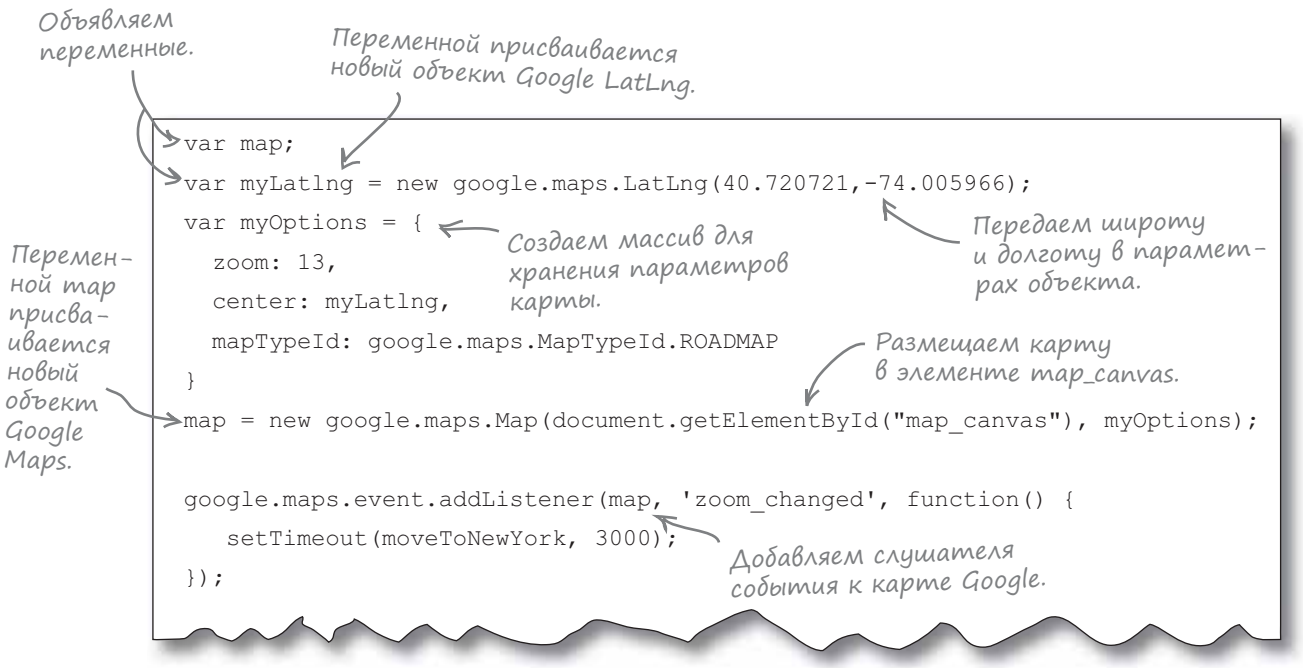
```
var map;
function initialize() {
  var myLatlng = new google.maps.LatLng(40.720721,-74.005966);
  var myOptions = {
    zoom: 13,
    center: myLatlng,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  }
  map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);

  google.maps.event.addListener(map, 'zoom_changed', function() {
    setTimeout(moveToNewYork, 3000);
  });

  var marker = new google.maps.Marker({
      position: myLatlng,
      map: map,
      title:"Hello World!"
  });
  google.maps.event.addListener(marker, 'click', function() {
    map.setZoom(8);
  });
}

function moveToNewYork() {
  var NewYork = new google.maps.LatLng(45.526585, -122.642612);
  map.setCenter(NewYork);
}
```

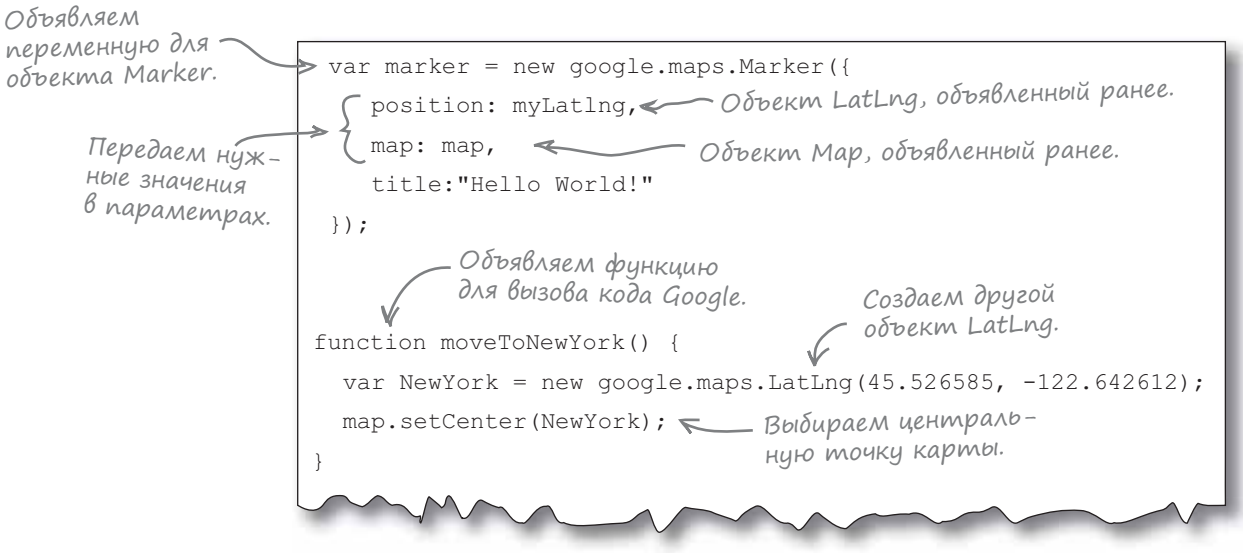
Это код Google.

Но все не так страшно, как может показаться. Давайте разберемся, что же происходит в этом коде.

В API используются объекты

В главе 6 мы создавали объекты JavaScript. Используя свойства и методы этих объектов, мы организуем хранение и обработку информации так, как считаем нужным. Многие компании — Google, Netflix, Microsoft и Yahoo! — тоже создают объекты для работы со своими данными через API. Если вы успели забыть, что такое объекты и как они работают, вернитесь к главе 6 и перечитайте ее.





Значит, в APIs все делается при помощи объектов? И почему мы должны использовать объекты, созданные кем-то другим?

Потому что это ускоряет работу.

Из главы 6 вы знаете, что информацию удобно хранить в объектах. Объекты используются для хранения переменных, относящихся к одной сущности. Фактически API — это набор конструкторов объектов, которые позволяют создавать собственные экземпляры объектов, определенных другими разработчиками. Если есть экземпляр объекта, вы можете использовать все его свойства и методы в коде!

Внедрить в приложение поддержку географических карт достаточно сложно; именно поэтому Google использует объекты в своем API. Это позволяет разработчикам создавать объекты с множеством разных методов и свойств, необходимых для построения карты.

Объект карты содержит свойства `zoom`, `center` и `mapTypeId`, а также много других свойств, в предыдущем примере не использовавшихся. Объект карты также поддерживает многочисленные методы, например `setCenter`.

Задание! Создайте страницу *display_one.html* и сохраните ее в папке проекта этой главы.

display_one.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>View Cryptid Sightings</title>
      <link type="text/css" href="style/form.css" rel="stylesheet"/>
      <link type="text/css" href="jquery-ui-1.8.16.custom/css/sunny/jquery-ui-1.8.16.custom.css" />
   </head>
   <body>
      <div class="ui-widget-header ui-corner-top form_pad">
         <h2>View Cryptid Sightings</h2>
      </div>
      <div class="ui-widget-content form_pad">
         <div id="map_canvas"></div>
      </div>
      <script src="http://maps.google.com/maps/api/js?sensor=false"></script>
      <script src="scripts/jquery-1.6.2.min.js"></script>
      <script src="scripts/maps.js"></script>
   </body>
</html>
```

Создаем область для размещения карты.

Включаем Google Maps API.

Включаем библиотеку jQuery.

Включаем файл maps.js.

Включение карт Google в страницу

Сначала скопируйте все необходимые файлы, которые были созданы в конце предыдущей главы. Наша работа продолжится с того момента, на котором она прервалась ранее. В этом коде (вместе с новым файлом *display_one.html*) присутствуют два важных изменения.

- Область `div` с идентификатором `map_canvas`.
- Код Google Maps API, добавленный с включением фрагмента **`<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>`**.

Чтобы создать карту Google на странице, необходимо создать файл *maps.js* и включить в него функцию с вызовом кода API.

Развлечения с магнитами

Расставьте магниты так, чтобы у вас получился код функции `initialize`. Эта функция должна создавать новый экземпляр объекта Google Maps `map` с использованием параметров, определенных в коде. Объект `map` размещается в элементе `map_canvas` страницы. Также включите в существующий файл *form.css* определения стилей контейнера карты.

```
#map_canvas {
    _____________ :left;
    height: 450px;
    width: _____________
}
```

form.css

```
$(document).ready. _____________{
   var _____________
   function _____________{
      var latlng = new google.maps.LatLng. _____________________;
      var mapOpts = {
         _____________: 13,
         center: latlng,
         mapTypeId: google.maps.MapTypeId._____________
      };
      map = ____________google.maps.Map(document.getElementById(_____________), mapOpts);
   }
   initialize();
});
```

maps.js

```
map;
zoom
float
HYBRID
(45.519098,-122.672138)
"map_canvas"
new
450px;
initialize()
(function()
```

Развлечения с магнитами. Решение

Расставьте магниты так, чтобы получился код функции `initialize`. Эта функция должна создавать новый экземпляр объекта Google Maps `map` с параметрами, определенными в коде. Объект `map` размещается в элементе `map_canvas` страницы. Также включите в существующий файл *form.css* определения стилей контейнера карты.

```
#map_canvas {
  float: left;
  height: 450px;
  width: 450px;
}
```

form.css

```
$(document).ready(function() {
  var map;
  function initialize() {
    var latlng = new google.maps.LatLng(45.519098,-122.672138);
    var mapOpts = {
      zoom: 13,
      center: latlng,
      mapTypeId: google.maps.MapTypeId.HYBRID
    };
    map = new google.maps.Map(document.getElementById("map_canvas"), mapOpts);
  }
  initialize();
});
```

maps.js

Часто Задаваемые Вопросы

В: А что это за объект `LatLng` и свойство `mapOpts`?

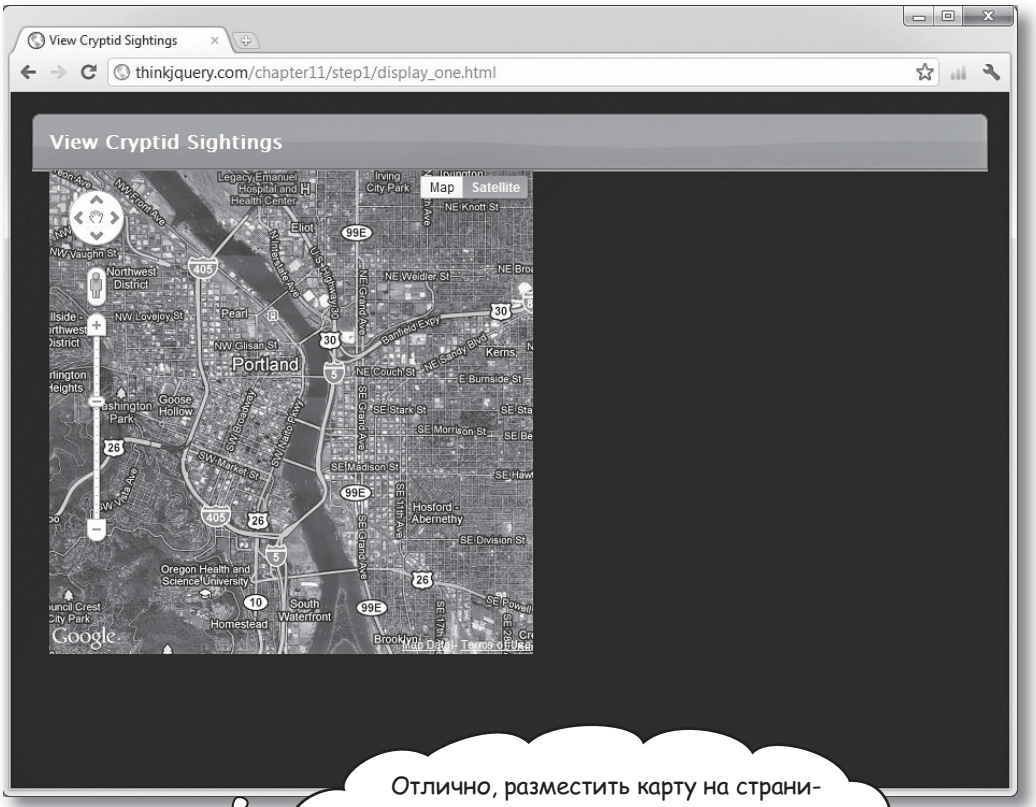
О: За дополнительной информацией об объектах, методах и прочих аспектах API обращайтесь по адресу *http://code.google.com/apis/maps/documentation/javascript/reference.html*. На сайте имеются примеры кода и более подробная информация о разных объектах и методах, которые понадобятся вам при программировании.

В: Google Maps — это единственный способ создания карт?

О: Конечно, нет! Однако это самый популярный способ, поэтому мы здесь им и воспользовались. Другие компании (такие как Yahoo!, Microsoft, MapQuest и OpenLayers) тоже предлагают свои картографические API.

Тест-драйв

Включите в файл *maps.js* функцию `initialize`, собранную из магнитов. Также убедитесь в том, что файл *maps.js* включен в *display_one.html*. Затем откройте файл *display_one.html* в браузере. Весь код должен выполняться через веб-сервер, поэтому URL-адрес должен начинаться с префикса *http://*, а не *file://*.



Отлично, разместить карту на странице было несложно... Но на ней нет никаких данных криптидов. Их нужно загрузить из базы данных, верно?

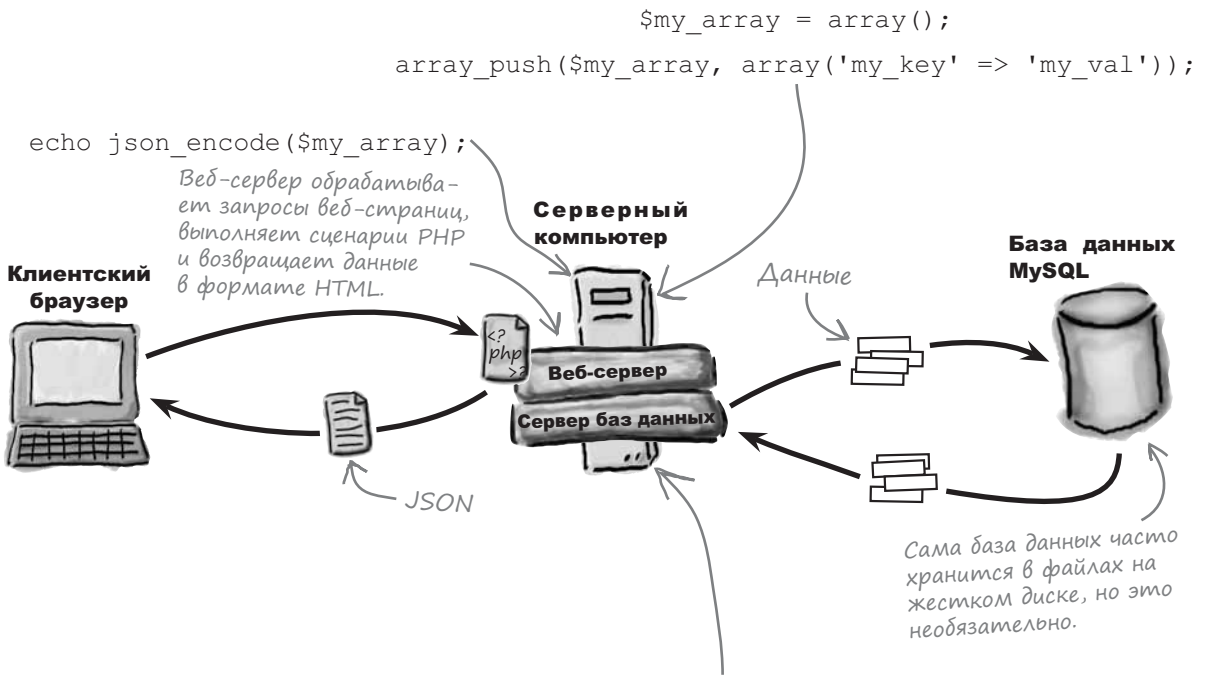
Точно.

В главе 9 вы научились читать информацию из базы данных MySQL с использованием кода jQuery, Ajax, JSON, PHP и MySQL. Хотя список задействованных технологий получается довольно длинным, в результате мы получаем именно то, что нужно. Давайте посмотрим, как использовать эти технологии на этот раз.

Чтение данных JSON средствами SQL и PHP

В главе 9 было показано, как команда SQL `select` извлекает нужную информацию из базы данных, чтобы файл PHP мог преобразовать ее в формат JSON и вернуть странице с использованием Ajax.

Вы также научились использовать Ajax для получения от файла PHP информации, закодированной в формате JSON. С возвращением данных JSON страницей PHP все было просто — функция `json_encode` получала массив и возвращала данные JSON, с которыми можно было работать средствами jQuery.



Весь код PHP и SQL для этой главы мы написали за вас. Скопируйте базу данных MySQL из главы 10 и двигайтесь дальше! Остальной код SQL и PHP находится в архиве кода главы; выполните его на своем сервере. Весь код PHP и SQL, объединенный в один файл, можно загрузить по адресу *http://thinkjquery.com/chapter11/end/service.zip*.

Развлечения с магнитами: jQuery, HTML и CSS

Расставьте магниты в коде файлов *display_one.html*, *forms.css* и *maps.js* так, чтобы обеспечить загрузку данных в формате JSON и их отображение на экране. Добавьте элементы `div` и `ul` для хранения данных, немного кода CSS для стилевого оформления списка, а также функцию получения данных (через JSON) для включения криптидов в список.

```
function getAllSightings(){
   $.getJSON("service.php?action=getAllSightings",_____________________ {
      if (_____________length > 0) {
         $("#sight_list")_____________________;
         _____________________(json.sightings,function() {
            var info = 'Date: ' + this['date'] + ', Type: ' + this['type'];
            var $li = $("<li />");
            _____________________
            $li.addClass(_____________________);
            $li.attr('id', this['id']) ;
            $li.appendTo(_____________________);
         });
      }
   });
}
```

maps.js

"map_canvas"

"sightings"

json.sightings.

li.sightings:hover {

$li.html(info);

```
#sight_nav{
   float:left;
}
ul#sight_list{
   width:150px;
   padding:0px;
   margin:0px;
}
li.sightings {
   padding:4px;
   background:#7B7382;
   border:1px #000 solid;
   color:#fff;
   _____________________
}
  _____________________
   background:#eee;
   color:#000;
}
```

form.css

```
<div class="ui-widget-content form_pad">
   <div id= _____________________></div>
   <div id="sight_nav">
      <ul id= _____________________>
      </ul>
   </div>
</div>
```

display_one.html

.empty()

$.each

list-style:none;

"sight_list"

"#sight_list"

function(json)

Развлечения с магнитами: jQuery, HTML и CSS. Решение

Расставьте магниты в файлах *display_one.html*, *forms.css* и *maps.js* так, чтобы загружать данные в формате JSON и отображать их на экране. Добавьте элементы `div` и `ul` для хранения данных, немного кода CSS для стилевого оформления списка, а также функцию получения данных (через JSON) для включения криптидов в список.

```
function getAllSightings(){
   $.getJSON("service.php?action=getAllSightings", function(json) {
      if (json.sightings.length > 0) {
         $("#sight_list").empty();
         $.each(json.sightings,function() {
            var info = 'Date: ' + this['date'] + ', Type: ' + this['type'];
            var $li = $("<li />");
            $li.html(info);
            $li.addClass("sightings");
            $li.attr('id', this['id']) ;
            $li.appendTo("#sight_list");
         });
      }
   });
}
```

maps.js

```
<div class="ui-widget-content form_pad">
   <div id="map_canvas"></div>
   <div id="sight_nav">
      <ul id="sight_list">
      </ul>
   </div>
</div>
```

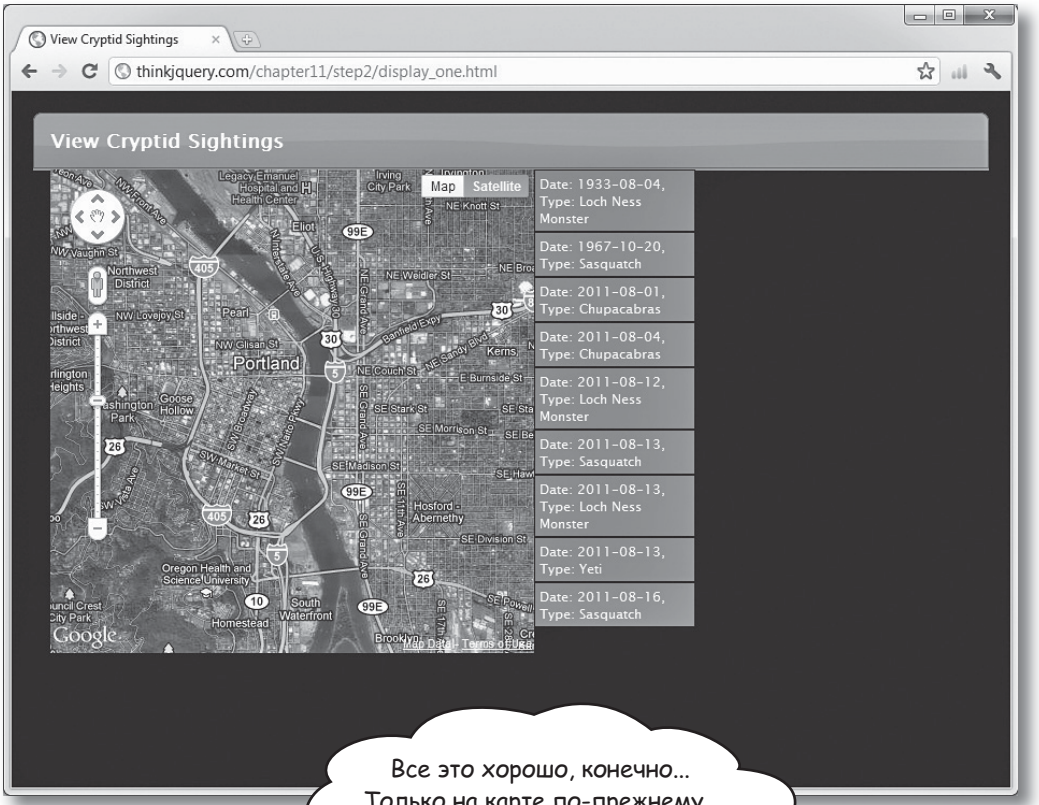
display_one.html

```
#sight_nav{
   float:left;
}
ul#sight_list{
   width:150px;
   padding:0px;
   margin:0px;
}
li.sightings {
   padding:4px;
   background:#7B7382;
   border:1px #000 solid;
   color:#fff;
   list-style:none;
}
li.sightings:hover {
   background:#eee;
   color:#000;
}
```

form.css

ТЕСТ-ДРАЙВ

Включите в файл *maps.js* только что созданную функцию `getAllSightings`. Включите ее вызов в конец функции `initialize`. Откройте файл *display_one.html* в браузере. Предполагается, что в главе 10 вы уже включили информацию о нескольких существах в базу данных; если вы еще этого не сделали, обязательно сделайте сейчас. Помните, что весь код должен выполняться через веб-сервер; соответственно URL-адрес должен начинаться с префикса *http://*, а не *file://*.



Все это хорошо, конечно... Только на карте по-прежнему ничего не отображается.

Точно. Нужно нанести данные криптидов на карту.

В интерфейса Google Maps для этой цели существует очень простой метод. Давайте посмотрим, как он работает.

Точки на карте — маркеры

Естественно, Google поддерживает возможность нанесения точек на карту, только они называются не точками, а *маркерами* (marker). Маркеры представляют собой объекты (как и все остальное в Google Maps API); они также обладают методами и свойствами, через которые с ними выполняются необходимые операции.

Определяем новый объект LatLng из Google Maps API.

Передаем фактические значения широты и долготы в параметрах.

```
var myLatLng = new google.maps.LatLng(45.519098,-122.672138);
```

Определяем объект с именем my_marker как экземпляр объекта маркера Google.

Вызываем конструктор объекта Google Marker.

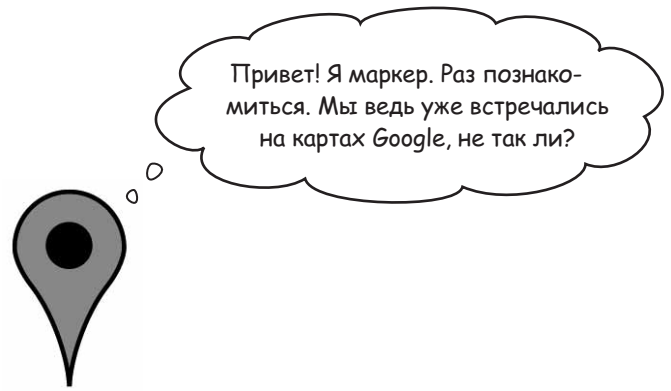
```
var my_marker = new google.maps.Marker({
    position: myLatlng,
    map: map,
    title:"Hi! I'm a marker. Seen me around before?"
});
```

Определенный ранее объект LatLng передается в параметре.

Готовый объект карты. Вы уже видели, как он работает.

Передаем аргументы для задания свойств объекта my_marker.

Здесь мы задаем позицию маркера, карту, на которой он размещается, и название маркера.



Возьми в руку карандаш

Включите в функцию `getAllSightings` добавление слушателя события click к элементу списка *до того*, как он будет включен в список. Событие click должно вызывать `getSingleSighting` с параметром (идентификатор наблюдения, на котором щелкнул пользователь). Новая функция загрузит информацию о наблюдении и добавит его на карту в виде маркера.

```
function getAllSightings(){
   $.getJSON("service.php?action=getAllSightings", function(json) {
      if (json.sightings.length > 0) {
         $("#sight_list").empty();
         $.each(json.sightings,function() {
            var info = 'Date: ' + this['date'] + ', Type: ' + this['type'];
            var $li = $("<li />");
            $li.html(info);
            $li.addClass("sightings");
            $li.attr('id', this['id']) ;
            $li.click(function(){
               _____________________ this['id'] );
            });
            $li.appendTo("#sight_list");
         });
      }
   });
}

function getSingleSighting(_____________________){
   $.getJSON("service.php?action=getSingleSighting&id="+id, function(json) {
      if (json.sightings.length > 0) {

         ___________________________________
            var loc = new google.maps.LatLng(this['lat'], this['long']);
            var my_marker = new google.maps_____________________({
               _____________________loc,
               map: map,
               title:this['type']
            });
            ___________________setCenter(loc, 20);
         });
      }
   });
}
```

maps.js

Возьми в руку карандаш Решение

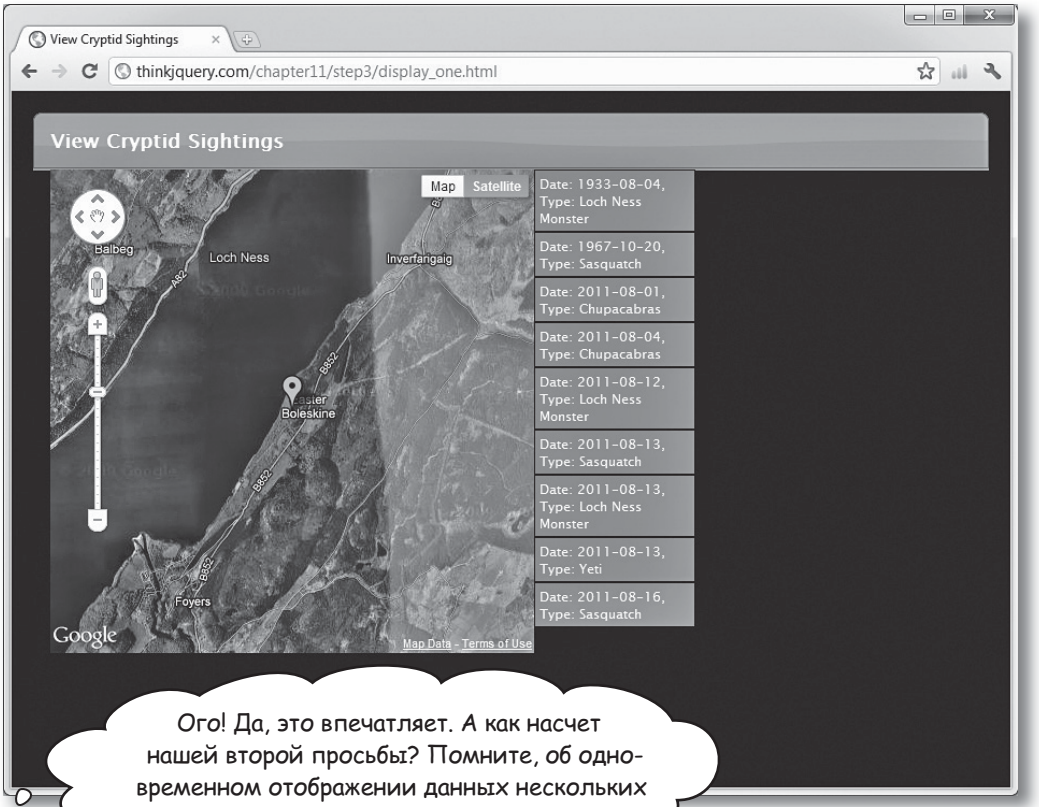
После включения всего кода элементы списка реагируют на щелчки, они загружают данные о криптиде, на котором щелкнул пользователь, и размещают их на карте.

```
function getAllSightings(){
   $.getJSON("service.php?action=getAllSightings", function(json) {
      if (json.sightings.length > 0) {
         $("#sight_list").empty();
         $.each(json.sightings,function() {
            var info = 'Date: ' + this['date'] + ', Type: ' + this['type'];
            var $li = $("<li />");
            $li.html(info);
            $li.addClass("sightings");
            $li.attr('id', this['id']) ;
            $li.click(function(){
               getSingleSighting( this['id'] );
            });
            $li.appendTo("#sight_list");
         });
      }
   });
}

function getSingleSighting(id ){
   $.getJSON("service.php?action=getSingleSighting&id="+id, function(json) {
      if (json.sightings.length > 0) {
         $.each(json.sightings,function() {
            var loc = new google.maps.LatLng(this['lat'], this['long']);
            var my_marker = new google.maps.Marker ({
               position: loc,
               map: map,
               title:this['type']
            });
           map.setCenter(loc, 20);
         });
      }
   });
}
```

maps.js

ТЕСТ-ДРАЙВ

Включите в файл *maps.js* только что созданные функции `getAllSightings` и `getSingleSighting`. Откройте файл *display_one.html* в браузере (как и прежде, с использованием префикса *http://*).



Ого! Да, это впечатляет. А как насчет нашей второй просьбы? Помните, об одновременном отображении данных нескольких наблюдений?

С первыми двумя требованиями мы уже разобрались. Пора заняться последним требованием в списке криптозоологов.

3) Нам хотелось бы, чтобы при выборе типа существа из списка отображалась информация обо всех существах, связанных с выбранным типом в нашей базе данных. При этом информация о существах должна отображаться на карте Google, чтобы найти места частых наблюдений для внимательного изучения. Можно ли сделать так, чтобы все эти точки тоже реагировали на щелчки для получения дополнительной информации (как и пункты списка существ)?

Список задач для отображения нескольких существ

Вот что нужно сделать для реализации последнего требования.

1. *Создать раскрывающий список типов существ (информация из базы данных).*
2. *При изменении состояния списка получить соответствующий список существ.*
3. *Вывести информацию обо всех существах из базы данных, в списке и на карте.*
4. *И список, и маркеры на карте должны реагировать на щелчки, чтобы на карте отображалась дополнительная информация.*

Готово к употреблению

Создайте новую страницу с именем *display_type.html* и сохраните ее в одном каталоге с другими файлами HTML этого проекта. Новый файл будет отображать список доступных типов существ. Все существа выбранного типа отображаются на карте. По структуре и стилю новая страница почти не отличается от старой, если не считать добавления элемента `select` с идентификатором `ddlTypes`.

```
<!DOCTYPE html>
<html>
   <head>
      <title>View Cryptid Sightings</title>
      <link type="text/css" href="style/form.css" rel="stylesheet" />
      <link type="text/css" href="jquery-ui-1.8.16.custom/css/sunny/
jquery-ui-1.8.16.custom.css"/>
   </head>
   <body>
      <div class="ui-widget-header ui-corner-top form_pad">
         <h2>View Cryptid Sightings</h2>
      </div>
      <div class="ui-widget-content form_pad">
         <div id="map_canvas"></div>
         <div id="sight_nav">
            <select id="ddlTypes">
               <option value="">-- Please Select --</option>
            </select>
            <ul id="sight_list"></ul>
         </div>
      </div>
      <script src="http://maps.google.com/maps/api/js?sensor=false"></script>
      <script src="scripts/jquery-1.6.2.min.js"></script>
      <script src="scripts/maps.js"></script>
   </body>
</html>
```

Добавляем раскрывающийся список, заполняемый типами существ.

Включаем Google Maps API.

Включаем файл maps.js.

Включаем библиотеку jQuery.

display_type.html

Развлечения с магнитами

Расставьте магниты так, чтобы заполнить пропуски в функции `getAllTypes`, которая вызывает файл *service.php* (включенный в архив кода этой главы) для получения списка разных типов существ в базе данных. Полученные типы включаются в раскрывающийся список с идентификатором `ddlTypes`. Для раскрывающегося списка определяется слушатель события `change`, который выводит выбранное значение. Поскольку файл *maps.js* используется для двух файлов HTML, в функцию `initialize` включается проверка существования раскрывающегося списка. Если список существует, то вызывается функция `getAllTypes`, а если нет — функция `getAllSightings`.

```
function initialize(){
.
.
map = new google.maps.Map(document.getElementById(_________________), mapOpts);
   if ( $('#ddlTypes').length ) {
      _____________________
   }else{
      _____________________
   }
}
function getAllTypes(){
   $.getJSON("service.php?action=getSightingsTypes", function(json_types) {
      if (_____________________creature_types.length > 0) {
         $.each(json_types.creature_types,_____________________
            var info = this['type'];
            var $li = _____________________
            $li.html(info);
            $li_____________________("#ddlTypes");
         });
      }
   });
}

_____________________change(function() {
   if($(this).val() != ""){
      alert( $(this).val() );
   }
});
```

$("<option />");

getAllSightings();

"map_canvas"

getAllTypes();

.appendTo

json_types.

function() {

$('#ddlTypes').

maps.js

Развлечения с магнитами. Решение

Теперь в нашем приложении имеется раскрывающийся список, связанный с картой Google, и функция получения информации (в формате JSON) о типах существ, а приложение выводит сообщение о выбранном типе существа.

```
function initialize(){
.
.
map = new google.maps.Map(document.getElementById( "map_canvas" ), mapOpts);
   if ( $('#ddlTypes').length ) {
      getAllTypes();
   }else{
      getAllSightings();
   }
}
function getAllTypes(){
   $.getJSON("service.php?action=getSightingsTypes", function(json_types) {
      if ( json_types.creature_types.length > 0) {
         $.each(json_types.creature_types, function() {
            var info = this['type'];
            var $li = $("<option />");
            $li.html(info);
            $li.appendTo("#ddlTypes");
         });
      }
   });
}

$('#ddlTypes').change(function() {
   if($(this).val() != ""){
      alert( $(this).val() );
   }
});
```

Проверка существования элемента по свойству length.

Получение типов из базы данных с использованием JSON и PHP.

Задаем текст пункта раскрывающегося списка.

Присоединяем новый пункт к раскрывающемуся списку.

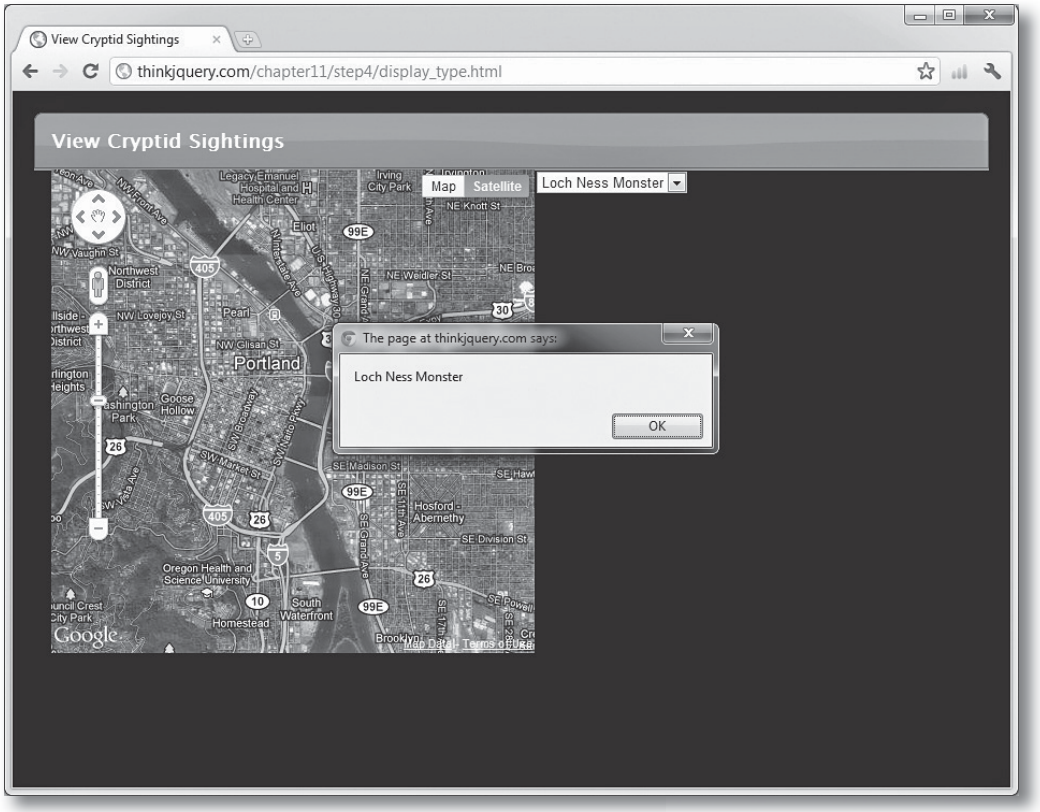
Добавляем слушателя для события change раскрывающегося списка.

Значение выбранного пункта списка.

maps.js

ТЕСТ-ДРАЙВ

Включите в файл *maps.js* функцию `getAllTypes` и слушателя события `change` раскрывающегося списка. Также внесите необходимые изменения в функцию `initialize`. Наконец, откройте файл *display_type.html* (снова с использованием префикса *http://*).



Работа идет полным ходом, карта готова в кратчайшие сроки!
Пора вычеркнуть пару требований из списка.

1. ~~Создать раскрывающийся список типов существ (информация читается из базы данных).~~
2. ~~При изменении состояния раскрывающегося списка получить из базы список существ выбранного типа.~~
3. Вывести информацию обо всех существах, полученную из базы данных, в списке и на карте.
4. И список, и маркеры на карте должны реагировать на щелчки, чтобы на карте отображалась дополнительная информация.

Эй, не так быстро! Когда я выбираю один из пунктов раскрывающегося списка, никакие существа не отображаются. Я вижу только окно сообщения! По-моему, второй пункт еще не готов.

А ведь и верно!

Нужно изменить наш код так, чтобы при изменении состояния раскрывающегося списка информация загружалась из базы данных (вместо простого вывода типа существа в окне сообщения).

И *тогда* этот элемент списка может будет вычеркнуть. Но раз уж мы занялись этой задачей, заодно выполним третий пункт из списка задач. Закатайте рукава, нам придется изрядно повозиться, чтобы собрать все воедино.

Длинные упражнения

Заполните пропуски в коде функции `getSightingsByType`. Этой функции передается один параметр: тип существа, по которому просматривается информация. Функция получает данные в формате JSON, перебирает возвращенных существ (если они есть) и добавляет на карту маркеры. Создайте еще две глобальные переменные: массив с именем `markerArray`, и новый объект Google Maps `LatLngBounds` с именем `bounds`, и функцию, которая удаляет предыдущие маркеры перед добавлением новых, когда выбор в списке меняется.

```
var markersArray = [];
var bounds = new google.maps_________________;
function getSightingsByType(type){
   $.getJSON("service.php?action=getSightingsByType&type="+type, function(json)
{
      if (_________sightings.length > 0) {
         $('#sight_list').empty();
         $.each(json.sightings,function() {
            var loc = new google.maps_______________(this['lat'], this['long']);
            var opts = {
               map: map,
               position:_________________
            };
```

```
        var marker = new google.maps._________________(opts);
           markersArray.push(_________________);
           var $li = $("<li />");
           $li.html('Date: ' + this['date'] + ', Type: ' + this['type']);
           $li_________________("sightings");
           $li.appendTo("#sight_list");
           bounds.extend(loc);
        });
        map.fitBounds(bounds);
     }
  });
}
$('#ddlTypes').change(function() {
  if($(this).val() != ""){
     clearOverlays();
     _________________( $(this).val() );
  }
});
function _________________ {
  if (markersArray) {
     for (i in markersArray) {
        markersArray[i].setMap(null);
     }
     markersArray.length = 0;
     bounds = null;
     bounds = new_________________LatLngBounds();
  }
}
```

maps.js

Решение длинных упражнений

После добавления двух новых глобальных переменных и вызовов функций Google Maps приложение добавляет и удаляет маркеры на карте при изменении выбора в списке. Маркеры, добавленные на карту, включаются в массив `markerArray` и используются для расширения границ карты `bounds`. Таким образом, карта масштабируется по точкам при помощи функции `fitBounds`. Функция `getSightingsByType`, теперь вызываемая при любом изменении выбора в раскрывающемся списке, добавляет маркеры на карту и включает существо в список на странице.

```
var markersArray = [];
var bounds = new google.maps.LatLngBounds(); 
function getSightingsByType(type){
   $.getJSON("service.php?action=getSightingsByType&type="+type, function(json)
{
      if (json.sightings.length > 0) {
         $('#sight_list').empty();
         $.each(json.sightings,function() {
            var loc = new google.maps.LatLng (this['lat'], this['long']);
            var opts = {
               map: map,
               position:loc
            };

            var marker = new google.maps.Marker (opts);
            markersArray.push(marker);
            var $li = $("<li />");
            $li.html('Date: ' + this['date'] + ', Type: ' + this['type']);
            $li.addClass ("sightings");
            $li.appendTo("#sight_list");
            bounds.extend(loc);
         });
         map.fitBounds(bounds);
      }
   });
}
```

Новый объект LatLngBounds.

Получаем данные в формате JSON.

Создаем новый объект Marker для каждой точки на карте.

Включаем текущие координаты в объект bounds.

Выбираем масштаб по границам объекта bounds, чтобы на карте были видны все точки наблюдений.

```
$('#ddlTypes').change(function() {
   if($(this).val() != ""){
      clearOverlays();
      getSightingsByType ( $(this).val() );
   }
});
function clearOverlays() {
   if (markersArray) {
      for (i in markersArray) {
         markersArray[i].setMap(null);
      }
      markersArray.length = 0;
      bounds = null;
      bounds = new .google.maps. LatLngBounds();
   }
}
```

Перед добавлением маркеров удаляем с карты все старые маркеры.

Значение раскрывающегося списка передается в параметре функции.

Удаляем объект Marker с карты.

Также сбрасываем переменную bounds.

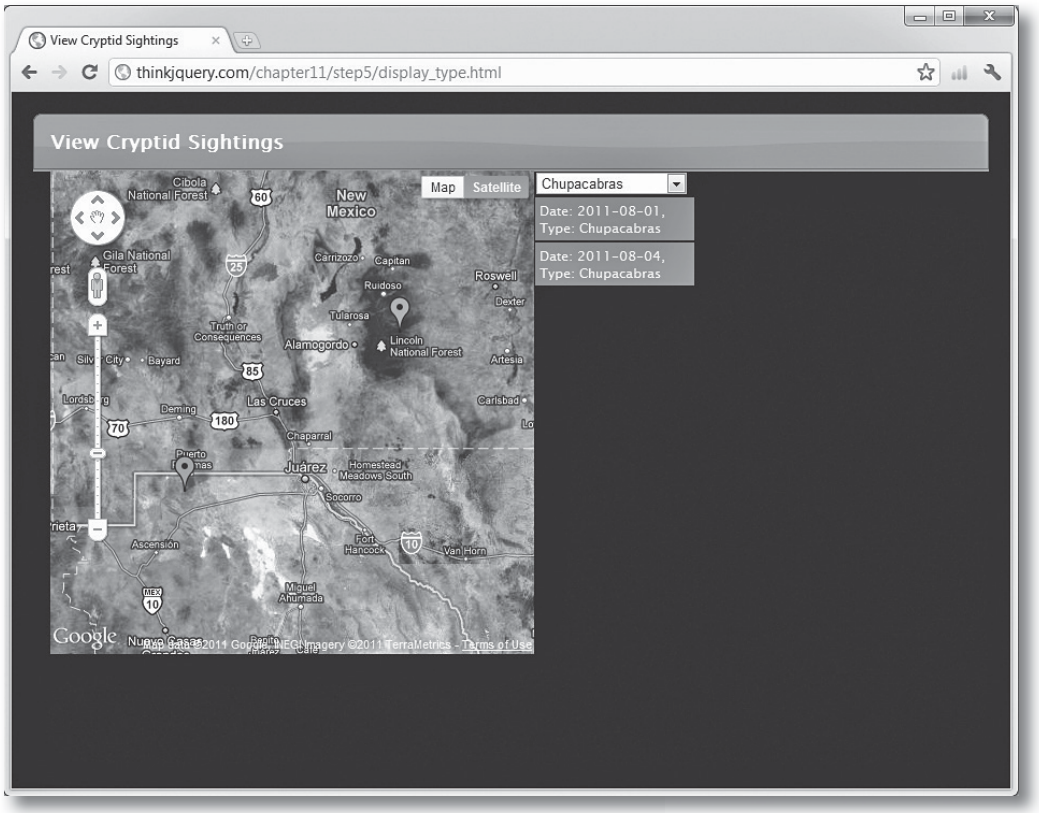
maps.js

Для любознательных

Мы включили функцию `clearOverlays`, которая удаляет ранее добавленные маркеры перед добавлением новых. На картах Google все дополнительные элементы, размещаемые на базовой карте, называются *слоями* (overlays); это могут быть объекты типа **Marker**, **Line**, **Polyline**, **Polygon** и других типов.

ТЕСТ-ДРАЙВ

Включите только что созданный код в файл *maps.js*. Откройте файл *display_type.html* в браузере и выберите один из типов существ в списке.



1. ~~Создать раскрывающийся список типов существ (информация читается из базы данных).~~
2. ~~При изменении состояния раскрывающегося списка получить из базы список существ выбранного типа.~~
3. ~~Вывести информацию обо всех существах, полученную из базы данных, в списке и на карте.~~
4. И список, и маркеры на карте должны реагировать на щелчки, чтобы на карте отображалась дополнительная информация.

Осталось совсем немного...
Всего один пункт!

МОЗГОВОЙ ШТУРМ

Вы уже знаете, как заставить элементы реагировать на щелчки в jQuery. Как это поможет нам с выполнением последнего требования в списке?

Часто Задаваемые Вопросы

В: Значит, я могу бесплатно использовать Maps API на своем сайте?

О: Да! Google бесплатно предоставляет API всем (для использования в личных или коммерческих целях), при условии выполнения пользовательского соглашения.

В: Но я не знаю, выполняю я эти условия или нет. Где можно ознакомиться с пользовательским соглашением?

О: Полный текст пользовательского соглашения находится по адресу *http://www.google.com/apis/maps/terms*.

В: Google Maps API работает с картами всего мира?

О: Осталось совсем немного стран, на которые сервис Google Maps не распространяется. Список можно найти на сайте Google Maps API list.

В: Могу ли я пользоваться этими картами на мобильных устройствах?

О: Да, можете. На момент издания книги вышла 3 версия Google Maps API. Она разрабатывалась с учетом мобильных устройств, для браузеров с поддержкой JavaScript.

В: Но я бы хотел написать специальное приложение. Google Maps API мне подойдет?

О: Если вы пишете на платформе Android или iPhone — подойдет. Google предоставляет для этих платформ библиотеки, которые следует включить в приложение. На других мобильных платформах вам придется использовать API так же, как на сайте.

В: Полная версия Google Maps позволяет находить маршруты. Способен ли на это данный API?

О: Нет, это невозможно. Компания Google разработала другой API — Google Directions API, который позволяет искать маршруты.

В: А как насчет поиска мест на карте по адресу?

О: На профессиональном жаргоне это называется *геокодированием*. У Google для этого есть специальный Geocoding API. Все упоминавшиеся интерфейсы входят в семейство Google Maps API. К счастью, все данные от Google поставляются в формате JSON, с которым вы уже умеете работать.

В: Какие еще API входят в семейство Google Maps API?

О: Directions API и Geocoding API относятся к подразделу, называемому Maps API Web Service. Также в этот подраздел входят интерфейсы Distance API, Elevation API и Places API.

В: А есть и другие?

О: Да. Есть Static Maps API для браузеров, не имеющих полноценной поддержки JavaScript; Maps API для Flash; и даже Earth API, который позволяет загрузить на страницу приложение Google Earth для просмотра трехмерного изображения земного шара, проведения виртуальных экскурсий и рисования фигур. Впрочем, для его использования необходимо установить модуль расширения Google Earth.

В: Существуют ли API для других языков, кроме JavaScript?

О: Да! Доступно бесчисленное множество разных APIs, одни бесплатны, другие требуют приобретения лицензии. Если вы ищете некую функциональность, которую вам не хочется писать самостоятельно, вполне возможно, что для нее уже существует готовый API.

Прослушивание событий карты

Мы уже почти подошли к концу. Вы уже видели разнообразные события, которые jQuery и JavaScript предоставляют для создания интересных интерактивных веб-приложений. Так как Google Maps API состоит из кода JavaScript (хотя и очень хорошо написанного и эффективного), он может использовать способность браузера прослушивать события и действовать соответствующим образом.

И по тем же причинам, по которым в jQuery были добавлены собственные функции создания слушателей, разработчики Google Maps предоставили возможность добавления слушателей событий через API. Не все браузеры одинаково работают со слушателями событий, и такой подход гарантирует, что API сможет управлять тем, как слушатели добавляются к странице. Ниже показано, как добавить слушателя события **click** для создания временного окна Google Maps *InfoWindow*.

Определяем переменную с содержимым, которое нужно отобразить.

```
var contentString = "This is an InfoWindow";

var my_infowindow = new google.maps.InfoWindow({
      content: contentString
});

google.maps.event.addListener(my_marker, 'click', function() {
   my_infowindow.open(map,my_marker);
});
```

Создаем экземпляр объекта Google Maps InfoWindow.

Задаем значение свойства content объекта InfoWindow.

Приказываем карте прослушивать событие click для нашего объекта маркера.

Этот код выполняется при щелчке на маркере (на карте появляется временное окно).

Для любознательных

В Google Maps API почти со всеми типами объектов (**Map**, **Marker**, **Line**, **InfoWindow**, **TrafficOverlay**, **Polygon** и другие) связываются события. И хотя события разных объектов часто имеют одинаковые ***имена***, они могут получать разные ***параметры***! Обязательно просмотрите документацию по объекту, который вы намереваетесь использовать.

Возьми в руку карандаш

Заполните пропуски в коде функции `getSightingsByType`, добавляющей функциональность обработки щелчков к маркерам на карте и к пунктам списка. Создайте глобальную переменную `info_window`, которая представляет собой новый экземпляр объекта Google Maps `InfoWindow`; по умолчанию объект инициализируется пустой строкой.

```
var info_window = new google.maps_________________({content: ''});
function_________________(type){
  $.getJSON("_________________ action=getSightingsByType&type="+type, function(json) {
    if (json.sightings.length > 0) {
      $('#sight_list').empty();
      $.each(json.sightings,function() {
        var info = 'Distance: ' + this[_________] + '<br>' + ' Height: ' + this['height'];
        info += ', Weight: ' + this['weight'] + ', Color: ' + this['color'] + '<br>';
        info += 'Latitude: ' + this['lat'] + ', Longitude: ' + this[_________];
        var loc = new _____________________(this['lat'], this['long']);
        var opts = {
          map: map,
          position:_____________
        };
        var marker = new google.maps _____________(opts);
        markersArray.push(marker);
        google.maps.event_____________(marker, 'click', function() {
          info_window.content = info;
          info_window.open(map, marker);
        });
        var $li = $("<li />");
        $li.html('Date: ' + this['date'] + ', Type: ' + this['type']);
        $li.addClass("sightings");
        $li_________(function(){
          info_window.content = info;
          info_window.open(map, _________);
        });
        $li.appendTo("#sight_list");
        _________ extend(loc);
      });
      map _____________ (bounds);
    }
  });
}
```

maps.js

Возьми в руку карандаш Решение

Обновленная функция `getSightingsByType` отображает данные в формате списка и на карте, а также заставляет маркеры карты и пункта списка реагировать на щелчки.

```
var info_window = new google.maps.InfoWindow({content: ''});
function getSightingsByType(type){
  $.getJSON("service.php?action=getSightingsByType&type="+type, function(json) {
    if (json.sightings.length > 0) {
      $('#sight_list').empty();
      $.each(json.sightings,function() {
        var info = 'Distance: ' + this['distance'] + '<br>' + ' Height: ' + this['height'];
        info += ', Weight: ' + this['weight'] + ', Color: ' + this['color'] + '<br>';
        info += 'Latitude: ' + this['lat'] + ', Longitude: ' + this["long" ];
        var loc = new google.maps.LatLng(this['lat'], this['long']);
        var opts = {
          map: map,
          position: loc
        };
        var marker = new google.maps.Marker (opts);
        markersArray.push(marker);
        google.maps.event.addListener (marker, 'click', function() {
          info_window.content = info;
          info_window.open(map, marker);
        });
        var $li = $("<li />");
        $li.html('Date: ' + this['date'] + ', Type: ' + this['type']);
        $li.addClass("sightings");
        $li.click(function(){
          info_window.content = info;
          info_window.open(map, marker);
        });
        $li.appendTo("#sight_list");
        bounds.extend(loc);
      });
      map.fitBounds(bounds);
    }
  });
}
```

Добавляем слушателя события click к объекту Marker на карте.

Добавляем слушателя события к списку, чтобы на карте открывалось окно InfoWindow.

maps.js

ТЕСТ-ДРАЙВ

Включите в файл *maps.js* новый код, созданный нами на предыдущей странице. Затем откройте файл *display_type.html* в браузере и выберите существо в списке. Щелкните на маркерах списка, чтобы просмотреть дополнительную информацию о существе.

1. ~~Создать раскрывающийся список типов существ (информация читается из базы данных).~~
2. ~~При изменении состояния раскрывающегося списка получить из базы список существ выбранного типа.~~
3. ~~Вывести информацию обо всех существах, полученную из базы данных, в списке и на карте.~~
4. ~~И список, и маркеры на карте должны реагировать на щелчки, чтобы на карте отображалась дополнительная информация.~~

Получилось!!!

Всего на нескольких страницах вам удалось создать полностью работоспособный сайт, использующий код на нескольких языках (PHP, SQL, JavaScript и jQuery), а также работающий с Google Maps API средствами Ajax и JSON для отображения довольно сложных данных. Не так уж мало!

Ваш инструментарий jQuery API

Глава 11 осталась позади, а вы научились использовать jQuery с разными API (а также JavaScript, PHP, MySQL, Ajax, JSON и не только!).

API

Прикладные программные интерфейсы — код, предоставляемый другими людьми (или компаниями) для работы с их данными, объектами и различными видами сервиса.

API предоставляют конструкторы, при помощи которых вы создаете свои экземпляры их объектов. Создавая экземпляр, вы получаете возможность использования всех методов и свойств, связанных с этими объектами, в своем коде.

Пара слов на прощание...

Рады были встретиться с вами в стране jQuery!

Жаль, что нам пора расставаться, но теперь вы знаете все необходимое для построения собственных сайтов, использующих jQuery, и вам пора заняться делом. Нам было приятно представить вам мир jQuery. Черкните нам пару строк или расскажите о своих классных разработках Feel на сайте Head First Labs: *http://headfirstlabs.com*.

Десять важных вещей (которые мы не рассмотрели)

Даже после всего сказанного многое осталось «за кадром». Существует масса других полезных возможностей jQuery и JavaScript, которые нам не удалось вместить в книгу. Было бы неправильно даже не упомянуть о них. Мы хотим, чтобы вы были готовы к любому аспекту jQuery, с которым вы можете столкнуться во время самостоятельных исследований.

1. Все, что есть в библиотеке jQuery

Вероятно, вы уже поняли, что библиотека jQuery огромна. Мы постарались представить основной материал, который может понадобиться новичку. Теперь вы знаете все необходимое и можете продолжить самостоятельное изучение библиотеки.

Методы jQuery

```
.add()
.addClass()
.after()
jQuery.ajax()
.ajaxComplete()
.ajaxError()
jQuery.ajaxPrefilter()
.ajaxSend()
jQuery.ajaxSetup()
.ajaxStart()
.ajaxStop()
.ajaxSuccess()
.andSelf()
.animate()
.append()
.appendTo()
.attr()
.before()
.bind()
.blur()
jQuery.browser
.change()
.children()
.clearQueue()
.click()
.clone()
.closest()
jQuery.contains()
.contents()
.context
.css()
jQuery.cssHooks
.data()
jQuery.data()
.dblclick()
deferred.always()
deferred.done()
deferred.fail()
deferred.isRejected()
deferred.isResolved()
deferred.pipe()
deferred.promise()
deferred.reject()
deferred.rejectWith()
deferred.resolve()
deferred.resolveWith()
deferred.then()
.delay()
.delegate()
.dequeue()
jQuery.dequeue()
.detach()
.die()
jQuery.each()
.each()
.empty()
.end()
.eq()
.error()
jQuery.error
event.currentTarget
event.data
event.isDefaultPrevented()
event.
isImmediatePropagationStopped()
event.isPropagationStopped()
event.namespace
event.pageX
event.pageY
event.preventDefault()
event.relatedTarget
event.result
event.stopImmediatePropagation()
event.stopPropagation()
event.target
event.timeStamp
event.type
event.which
jQuery.extend()
.fadeIn()
.fadeOut()
.fadeTo()
.fadeToggle()
.filter()
.find()
.first()
.focus()
.focusin()
.focusout()
jQuery.fx.interval
jQuery.fx.off
jQuery.get()
.get()
jQuery.getJSON()
jQuery.getScript()
jQuery.globalEval()
jQuery.grep()
.has()
.hasClass()
jQuery.hasData()
.height()
.hide()
jQuery.holdReady()
.hover()
.html()
jQuery.inArray()
.index()
.innerHeight()
.innerWidth()
.insertAfter()
.insertBefore()
```

1. Все, что есть в библиотеке jQuery (продолжение)

Методы jQuery (продолжение)

```
.is()
jQuery.isArray()
jQuery.isEmptyObject()
jQuery.isFunction()
jQuery.isPlainObject()
jQuery.isWindow()
jQuery.isXMLDoc()
jQuery()
.jquery
.keydown()
.keypress()
.keyup()
.last()
.length
.live()
.load()
.load()
jQuery.makeArray()
.map()
jQuery.map()
jQuery.merge()
.mousedown()
.mouseenter()
.mouseleave()
.mousemove()
.mouseout()
.mouseover()
.mouseup()
.next()
.nextAll()
.nextUntil()
jQuery.noConflict()
jQuery.noop()
.not()
jQuery.now()
.offset()
.offsetParent()
.one()
.outerHeight()
.outerWidth()
jQuery.param()
.parent()
.parents()
.parentsUntil()
jQuery.parseJSON
jQuery.parseXML()
.position()
jQuery.post()
.prepend()
.prependTo()
.prev()
.prevAll()
.prevUntil()
.promise()
.prop()
jQuery.proxy()
.pushStack()
.queue()
jQuery.queue()
.ready()
.remove()
.removeAttr()
.removeClass()
.removeData()
jQuery.removeData()
.removeProp()
.replaceAll()
.replaceWith()
.resize()
.scroll()
.scrollLeft()
.scrollTop()
.select()
.serialize()
.serializeArray()
.show()
.siblings()
.size()
.slice()
.slideDown()
.slideToggle()
.slideUp()
.stop()
jQuery.sub()
.submit()
jQuery.support
.text()
.toArray()
.toggle()
.toggle()
.toggleClass()
.trigger()
.triggerHandler()
jQuery.trim()
jQuery.type()
.unbind()
.undelegate()
jQuery.unique()
.unload()
.unwrap()
.val()
jQuery.when()
.width()
.wrap()
.wrapAll()
.wrapInner()
```

1. Все, что есть в библиотеке jQuery

Селекторы jQuery

```
Селектор "Все" ("*")
Селектор "Атрибут содержит префикс"
[name|="value"]
Селектор "Атрибут содержит" [name*="value"]
Селектор "Атрибут содержит слово" [name~="value"]
Селектор "Атрибут завершается" [name$="value"]
Селектор "Атрибут равен" [name="value"]
Селектор "Атрибут не равен" [name!="value"]
Селектор "Атрибут начинается" [name^="value"]
:animated
:button
:checkbox
:checked
Селектор дочерних элементов ("parent > child")
Селектор классов (".class")
:contains()
Селектор потомков ("ancestor descendant")
:disabled
Селектор элементов ("element")
:empty
:enabled
:eq()
:even
:file
:first-child
:first
:focus
:gt()
Селектор "Имеет атрибут" [name]
:has()
:header
:hidden
Селектор идентификаторов ("#id")
:image
:input
:last-child
:last
:lt()
Селектор множественных атрибутов [name="value"]
[name2="value2"]
Множественный селектор ("selector1, selector2,
selectorN")
Селектор "Следующий смежный" ("prev + next")
Селектор следующих одноуровневых элементов ("prev
~ siblings")
:not()
:nth-child()
:odd
:only-child
:parent
:password
:radio
:reset
:selected
:submit
:text
:visible Selector
```

2. jQuery CDN

CDN (C*ontent Distribution Networks*, «сети доставки контента») — большие сети серверов, предназначенные для хранения и распространения информации (данных, программных продуктов, кода API, мультимедийных файлов и видеороликов и т. д.), обеспечивающие доступность информации в Интернете. Каждый узловой сервер содержит копию предоставляемых данных. Рациональное размещение узлов в Сети позволяет ускорить доступ к информации для многих пользователей. Примеры традиционных CDN — Windows Azure и Amazon CloudFront.

Многие крупные фирмы предоставляют копии jQuery в общедоступных сетях CDN. Ниже приводятся ссылки на копии jQuery, которыми вы можете пользоваться.

- **Google Ajax API CDN**
  - *http://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js*
- **Microsoft CDN**
  - *http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.6.2.min.js*
- **jQuery CDN (через Media Temple)**
  - *http://code.jquery.com/jquery-1.6.2.min.js* (компактная версия)
  - *http://code.jquery.com/jquery-1.6.2.js* (исходная версия)

Вместо того чтобы каждый раз загружать код jQuery, вы можете включать в свои приложения эти ссылки.

3. Пространство имен jQuery: метод noConflict

Многие библиотеки JavaScript используют символ `$` в качестве имени функции или переменной, как это делает jQuery. В случае jQuery `$` представляет собой синоним для имени jQuery, поэтому вся функциональность работает и без использования `$`. Если вам понадобится включить другую библиотеку JavaScript, кроме jQuery, передайте контроль за именем `$` другой библиотеке, используя вызов `$.noConflict`:

```
<script type="text/javascript" src="other_lib.js"></script>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
      $.noConflict();
      //Код, использующий $ из других библиотек
</script>
```

Данный прием особенно эффективен в сочетании с возможностью метода `.ready` определять синоним для объекта jQuery, так как в функции обратного вызова, передаваемой `.ready`, можно использовать `$` без риска возникновения конфликтов:

```
<script type="text/javascript" src="other_lib.js"></script>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
      $.noConflict();
      jQuery(document).ready(function($) {
            // Код, использующий $ из jQuery
      });
      // Код, использующий $ из других библиотек
</script>
```

Используйте этот прием только, если вы собираетесь работать с другими библиотеками JavaScript, использующими ссылку `$`. Если в вашей странице используется только библиотека jQuery, все это вам не понадобится, даже в том случае, если страница включает несколько модулей расширения.

4. Отладка кода jQuery

Отладка кода всегда полезна, особенно если вы работаете над крупномасштабным проектом, в котором используется множество разнообразных объектов, включаемых файлов или API. Нередко бывает нужно узнать содержимое полученного объекта или переменной, но вы не хотите выводить его в окне сообщения или вообще разбираться с получением свойств объекта.

На помощь приходят отладочные модули расширения. Они помогут заглянуть внутрь объектов, чтобы вы могли проследить за интересующими вас свойствами или понаблюдать за изменением состояния переменной с течением времени, например узнать о непреднамеренном присваивании null. Все эти возможности чрезвычайно полезны при отладке кода JavaScript и jQuery.

При программировании на JavaScript и jQuery нам показались особенно полезными два отладочных модуля расширения — Dump и VariableDebugger.

> *http://plugins.jquery.com/project/Dump* (для просмотра содержимого объектов). *http://plugins.jquery.com/project/VariableDebugger* (аналогично, но с выводом информации во временном окне).

Существуют и другие отладочные модули расширения, а со временем наверняка их будет все больше — как и усовершенствований уже существующих модулей. Эти модули показались нам полезными, но если вам захочется поискать что-нибудь получше, посетите сайт jQuery Plug-ins (*http://plugins.jquery.com/*) и проведите поиск по слову «debug».

Конечно, в процессе отладки можно использовать вспомогательные средства браузера, неоднократно упоминавшиеся в книге.

5. Расширенная анимация и очереди

Очереди в jQuery чаще всего используются для анимации, но вы можете использовать их так, как считаете нужным.

Очередь представляет собой массив функций, существующий на уровне элемента и хранящийся в `jQuery.data`. Очереди работают по принципу FIFO (First In, First Out, т. е. «первым вошел, первым вышел». Функции включаются в очередь вызовом `.queue`, а исключаются вызовом `.dequeue`.

Каждый элемент может иметь одну или несколько очередей функций, присоединенных к нему из jQuery. В большинстве приложений используется только одна очередь (**fx**). Очереди позволяют выполнить с элементом последовательность операций **асинхронно**, т. е. без прерывания программы. Типичный пример — последовательный вызов нескольких методов анимации для элемента:

```
$('#my_element').slideUp().fadeIn();
```

При выполнении этой команды анимация скольжения начинается немедленно, а проявление ставится в очередь fx и вызывается только при завершении перехода скольжения.

Метод `.queue` напрямую работает с содержимым очереди функций. Особенно удобен вызов `.queue` с передачей функции обратного вызова; он позволяет поместить новую функцию в конец очереди.

Данная возможность сходна с передачей функции методу анимации, но она не требует передачи функции на момент запуска анимации.

```
$('#my_element').slideUp();
$('#my_element').queue(function() {
      alert('Animation complete.');
      $(this).dequeue();
});
```

Эта запись эквивалентна следующей:

```
$('#my_element').slideUp(function() {
      alert('Animation complete.');
});
```

Учтите, что при добавлении функции вызовом `.queue` для выполнения следующей функции в очереди необходимо вызвать функцию `.dequeue`.

В jQuery 1.4 вызываемая функция передается в первом аргументе другой функции. При ее вызове следующий элемент автоматически выводится из очереди, а очередь продолжает двигаться:

```
$("#test").queue(function(next) {
      // Здесь что-то происходит...
      next();
});
```

По умолчанию в jQuery используется очередь fx.

ПРИМЕЧАНИЕ. Если вы используете пользовательскую очередь, то должны вручную исключать функции из очереди вызовом `.dequeue`. В отличие от очереди по умолчанию fx, пользовательские очереди не поддерживают автозапуск!

6. Проверка форм

К сожалению, нам не хватило места для описания такой важной возможности, как использование jQuery для проверки данных форм на стороне клиента/браузера. В главах 9, 10 и 11 был приведен небольшой пример проверки данных на стороне сервера средствами PHP перед вставкой в базу. Проверка на стороне сервера очень важна, и даже можно сказать, необходима. Некорректно построенная команда `insert` или `select` может выдать намного больше информации о ваших данных, чем вы предполагали изначально.

Но вернемся к проверке данных на стороне клиента.

Существует много модулей расширения jQuery, предназначенных для этой цели. Одно из наших любимых решений — модуль «validation», доступный по адресу *http://docs.jquery.com/Plugins/validation*.

Этот модуль расширения позволяет определить набор правил для каждого элемента формы. В этих правилах вы настраиваете процедуру проверки и уточняете критерии правильности данных. Они могут быть любыми: минимальная или максимальная длина поля, проверка обязательных полей, проверка действительного формата адреса электронной почты и т. д. Далее приводятся некоторые примеры с сайта jQuery.

Элемент *name* определяется как обязательный, а элемент *email* – как обязательный и содержащий действительный адрес электронной почты.

```
$(".selector").validate({
      rules: {
            // Простое правило, преобразуемое в {required:true}
            name: "required",
            // Составное правило
            email: {
                  required: true,
                  email: true
            }
      }
});
```

Элемент определяется как обязательный (`required`*) с минимальной длиной (*`minlength`*) 2, причем для обоих правил определяются пользовательские сообщения.*

```
$("#myinput").rules("add", {
      required: true,
      minlength: 2,
      messages: {
            required: "Required input",
            minlength: jQuery.format("Please, at least {0} characters are
necessary")
      }
});
```

7. Эффекты jQuery UI

Библиотека jQuery UI Effects содержит ряд дополнительных анимаций, отсутствующих в базовой библиотеке jQuery. Эти эффекты можно разбить на три категории.

1. **Цветовые анимации**
   Цветовые анимации расширяют функцию `animate` так, чтобы она могла анимировать изменения цветов. Поддерживается цветовая анимация следующих свойств:

```
backgroundColor
borderBottomColor
borderLeftColor
borderRightColor
borderTopColor
color
outlineColor
```

2. **Переходы классов**
   Переходы классов расширяют базовый API классов так, чтобы он мог анимировать переход между двумя классами. jQuery UI изменяет следующие методы для передачи трех дополнительных параметров: `speed`, `easing` (необязательно) и `callback`.

`addClass(class)`
Добавляет класс с анимированным переходом между состояниями.

`removeClass(class)`
Исключает класс с анимированным переходом между состояниями.

`toggleClass(class)`
Добавляет класс, если он отсутствует, или исключает его, если он входит в группу.

`switchClass(currentClass, newClass)`
Заменяет один класс другим с анимацией перехода.

3. **Расширенное сглаживание**
   Расширенное сглаживание (easing) включено в базовую функциональность Effects. Это jQuery-версия функций сглаживания, созданных Робертом Пеннерсом на ActionScript для Flash. Это математические формулы, позволяющие сделать анимацию объектов более плавной и точной. Полный список всех функций сглаживания:

`linear`	`easeInQuart`	`easeInExpo`	`easeInBack`
`swing`	`easeOutQuart`	`easeOutExpo`	`easeOutBack`
`jswing`	`easeInOutQuart`	`easeInOutExpo`	`easeInOutBack`
`easeInQuad`	`easeInQuint`	`easeInCirc`	`easeInBounce`
`easeOutQuad`	`easeOutQuint`	`easeOutCirc`	`easeOutBounce`
`easeInOutQuad`	`easeInOutQuint`	`easeInOutCirc`	`easeInOutBounce`
`easeInCubic`	`easeInSine`	`easeInElastic`	
`easeOutCubic`	`easeOutSine`	`easeOutElastic`	
`easeInOutCubic`	`easeInOutSine`	`easeInOutElastic`	

8. Создание собственных модулей расширения jQuery

Написание модулей расширения jQuery — исключительно полезная возможность, которая экономит массу времени и усилий в ходе разработки. Самые сложные и ответственные функции абстрагируются в модули, предназначенные для повторного использования.

Вместо того чтобы долго объяснять тонкости создания модулей расширения jQuery, мы лучше предоставим это дело экспертам. Чрезвычайно содержательный и подробный учебник доступен по адресу *http://docs.jquery.com/Plugins/Authoring*.

Несколько рекомендаций для тех, кто только начинает заниматься разработкой собственных модулей расширения jQuery.

- Всегда заключайте свой модуль расширения в конструкцию `(function($){ // код модуля })(jQuery);`.
- Избегайте лишнего использования ключевого слова `this` в непосредственной области видимости функции вашего модуля расширения.
- Если только ваш модуль расширения не возвращает конкретное значение, всегда возвращайте из функции модуля расширения ключевое слово `this` (чтобы обеспечить возможность сцепления вызовов).
- Вместо длинного списка аргументов передавайте конфигурацию вашего модуля расширения в виде объекта, поля которого могут заменять настройки по умолчанию.
- Не загромождайте объект `jQuery.fn` более чем одним пространством имен на каждый модуль расширения.
- Всегда указывайте пространство имен для своих методов, событий и данных.

9. Замыкания

Замыкания (closures) — чрезвычайно сложный аспект JavaScript. Хотя замыкания не вошли в книгу, мы твердо убеждены в том, что они заслуживают хотя бы краткого упоминания.

Разобраться в замыканиях не так уж сложно. Необходимо понять основной принцип их работы. Но если ваше знакомство с темой начнется с чтения подробных технических описаний, это лишь собьет вас с толку.

Для начала пару определений...

- **Замыканием называется локальная переменная функции, которая продолжает существовать после возвращения управления.**

- **Каждый раз, когда ключевое слово `function` встречается внутри другой функции, внутренняя функция имеет доступ к переменным внешней функции.**

Странно, не правда ли?

Замыкания полностью зависят от области видимости переменных и объектов, той области, в которой объекты, переменные и функции создаются и остаются доступными и которая определяет контекст обращения к ним. Переменные, объекты и функции могут определяться либо в локальной, либо в глобальной области видимости.

Локальная область видимости: определение чего-либо в ограниченной части кода (например, внутри функции).

Глобальная область видимости: глобальные переменные, объекты и функции, в отличие от локальных, доступны в любой точке вашего кода.

Рассмотрим следующий пример:

```
function func1(x) {
      var tmp = 3;
      function func2(y) {
            alert(x + y + (++tmp));
      }
      func2(10);
}
func1(2);
```

9. Замыкания (продолжение)

Переменная `tmp` объявлена с локальной областью видимости внутри функции `func1`. В сообщениях будет выводиться значение 16, потому что `func2` может обратиться к переменной `x` (определенной как аргумент `func1`), а также к значению `tmp` из `func1`.

Но это не замыкание. Замыкание возникает при возвращении внутренней функции, и эта внутренняя функция «замыкается» над переменными `func1` перед выходом.

А теперь рассмотрим другой пример:

```
function func1(x) {
      var tmp = 3;
      return function (y) {
            alert(x + y + (++tmp));
      }
}
var func2 = func1(2);  // func2 is now a closure.
func2(10);
```

И снова `tmp` определяется в локальной области видимости, но функция `func2` имеет глобальную область видимости. Приведенная выше функция снова выдаст 16, потому что для `func2` значения `x` и `tmp` продолжают оставаться доступными, хотя они и не находятся непосредственно в ее области видимости.

Но поскольку переменная `tmp` находится в замыкании `func2`, она тоже будет увеличиваться при каждом вызове `func2`.

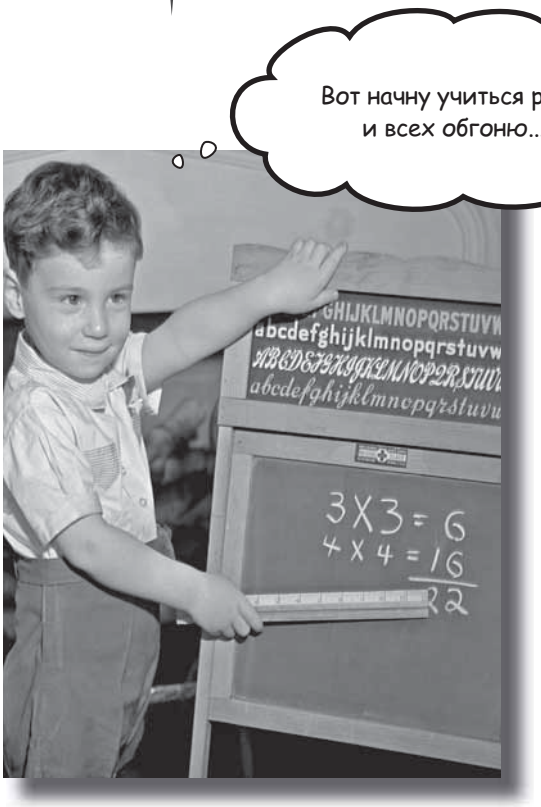
Существует техническая возможность создания нескольких функций замыкания, скажем, возвращением списка функций или посредством присваивания глобальным переменным. Все они будут обращаться к одним и тем же экземплярам `x` и `tmp`; они не будут создавать собственные копии.

10. Шаблоны

Шаблоны jQuery все еще находятся в фазе бета-тестирования, но уже сейчас можно сказать, что это полезный механизм, который поможет создавать более гибкие сайты без значительного объема кода HTML или jQuery. Шаблоны получают данные и связывают их с готовой разметкой, чтобы одна и та же разметка могла использоваться для отображения взаимосвязанных и похожих данных.

За дополнительной информацией обращайтесь по адресу: *http://api.jquery.com/category/plugins/templates/*.

Готовимся к великим свершениям



Вам понадобится среда, в которой вы сможете потренировать свои навыки PHP, но так, чтобы ваши данные не стали уязвимыми для внешних атак. Разработку приложений PHP всегда желательно начинать с безопасной среды и только потом открывать доступ для внешнего мира. В этом приложении приведены инструкции по установке веб-сервера, MySQL и PHP. После их выполнения в вашем распоряжении появится безопасная среда для работы и экспериментов.

Создание среды разработки PHP

Прежде чем вы сможете разместить готовое приложение в Web, необходимо разработать его с использованием ваших новых навыков jQuery и Ajax. Однако веб-приложение никогда не следует разрабатывать в Web, где любой желающий сможет получить доступ к нему. Установите программное обеспечение **локально, чтобы построить и полностью протестировать свое приложение до того, как переводить его в сетевой доступ.**

Для построения и тестирования приложений PHP и MySQL необходимо установить на локальном компьютере три программных продукта:

1. Веб-сервер
2. PHP
3. Сервер баз данных MySQL

PHP — не сервер, а набор правил, которые «понимает» веб-сервер (они нужны, чтобы интерпретировать код PHP). Веб-сервер и сервер MySQL — исполняемые программы, запускаемые на компьютере.

Помните, что речь идет о настройке **локального компьютера для выполнения функций** веб-сервера при разработке PHP. Вам также потребуется полноценный **сетевой веб-сервер, чтобы размещать готовые приложения**.

Что у вас уже есть?

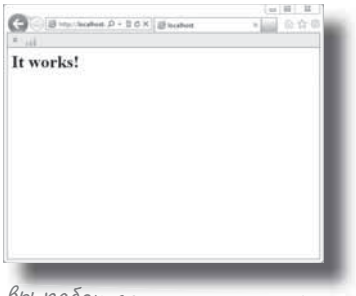
Прежде чем устанавливать компоненты среды разработки PHP, для начала проверьте, какие программы уже установлены на компьютере. Рассмотрим три компонента и разберемся, как определить, какие из них уже установлены в системе.

Состав установленного программного обеспечения сильно зависит от платформы компьютера. В Mac OS X веб-сервер установлен по умолчанию, а на большинстве компьютеров Windows его нет.

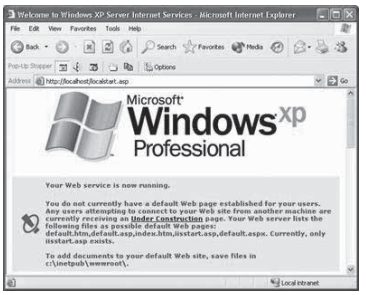
Примечание: материал приложения относится к Windows XP, Vista, Windows 7 и Windows Server 2003/2008, а на платформе Mac — к Mac OS X 10.3.x и более новых версий.

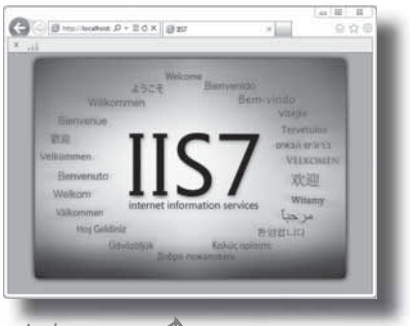
У вас установлен веб-сервер?

Если вы работаете на новом PC или Mac, возможно, веб-сервер уже установлен. Откройте окно браузера и введите в адресной строке команду адрес http://localhost. Если в браузере откроется заставка, это означает, что веб-сервер установлен и нормально работает.



Если вы работаете на компьютере Mac или Windows с установленным веб-сервером Apache, заставка выглядит примерно так.





На компьютере Windows с IIS заставка выглядит примерно так.

У вас установлена поддержка PHP? Какая версия?

Если веб-сервер установлен, вы сможете легко проверить, установлена ли у вас поддержка PHP, и какая именно версия. Создайте сценарий с именем *info.php* и включите в него следующую команду:

```
<?php phpinfo(); ?>
```

Сохраните файл в каталоге, используемом веб-сервером. В Windows типичная команда выглядит так:

C:\inetpub\wwwroot (для IIS)

или:

C:\Program Files (x86)\Apache Software Foundation\Apache2.2\htdocs (для Apache)

А вот как она обычно выглядит на Mac:

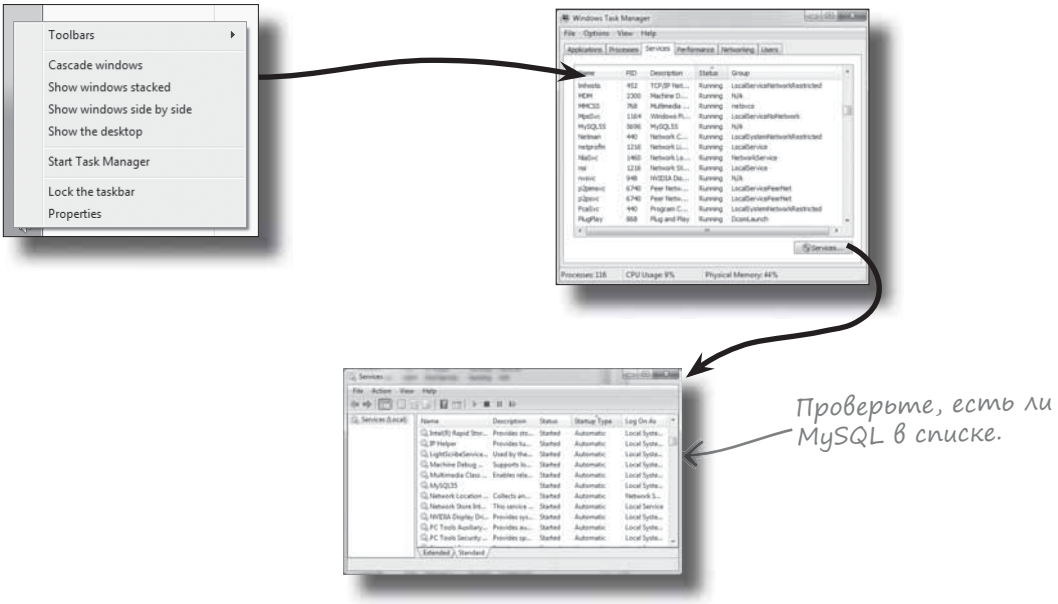
/Users/yourname/sites/

Если вы захотите открыть этот файл в своем браузере по адресу http://localhost/info.php, то при установленной поддержке PHP результат будет примерно таким:

Установленная версия PHP

У Вас установлен MySQL? Какая версия?

В системе Windows щелкните правой кнопкой мыши на панели задач Windows, выберите Диспетчер задач и перейдите на вкладку Службы.



Чтобы проверить наличие MySQL на Mac, откройте терминал и введите:

```
cd /user/local/mysql
```

Если команда работает, значит сервер MySQL установлен. Для проверки версии введите команду:

```
mysql
```

Терминал MySQL также называется «монитором» MySQL.

Если команда выполнена успешно, значит сервер MySQL установлен.

Версия MySQL, установленная в вашей системе

```
File Edit Window Help IHeartPHP
$ cd /usr/local/mysql
$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.0.51b MySQL Community Server (GPL)
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

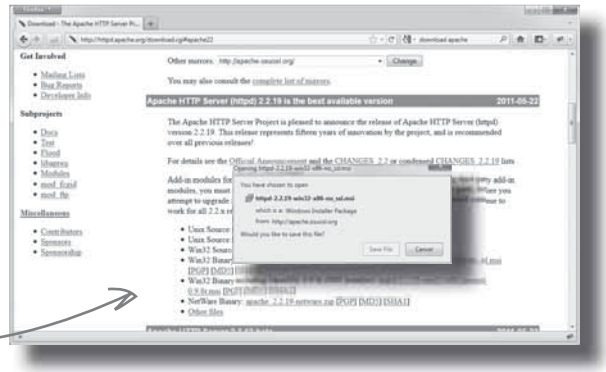
Начнем с веб-сервера

В зависимости от того, какая версия Windows установлена на вашем компьютере, вы можете загрузить Microsoft IIS (Internet Information Server) или Apache, веб-сервер с открытым кодом. Если вам нужен сервер для Mac, вероятно, стоит остановиться на веб-сервере Apache, потому что он уже установлен на вашем компьютере.

Далее приводится краткое описание установки Apache в Windows.

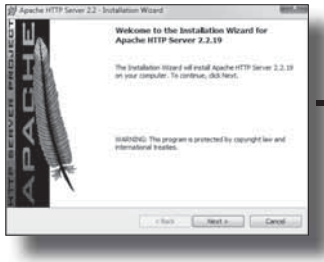
Откройте страницу *http://httpd.apache.org/download.cgi*.

Если вы используете Windows, мы рекомендуем загрузить файл *apache_2.2.19-win32-x86-no_ssl.msi*. Когда файл будет загружен, сделайте на нем двойной щелчок, чтобы начать автоматическую установку.

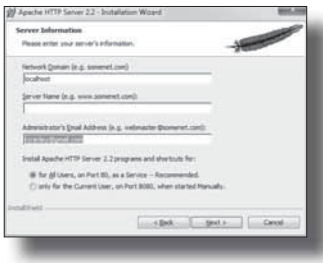


Загрузите эту версию и сделайте на ней двойной щелчок после загрузки.

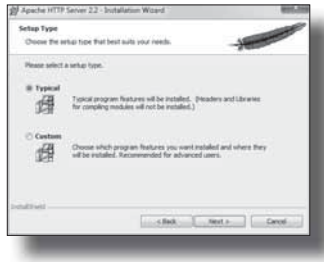
Открывается окно мастера установки. Инструкции в основном понятны без пояснений, и в большинстве случаев вы можете подтвердить выбор по умолчанию.



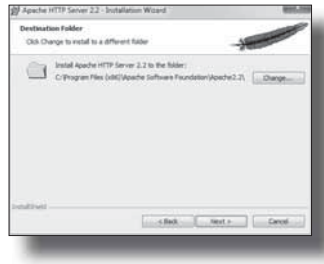
Выберите домен, к которому принадлежит ваш компьютер. Если домена нет, введите **`localhost`**.



Проще всего выбрать типичный вариант установки.

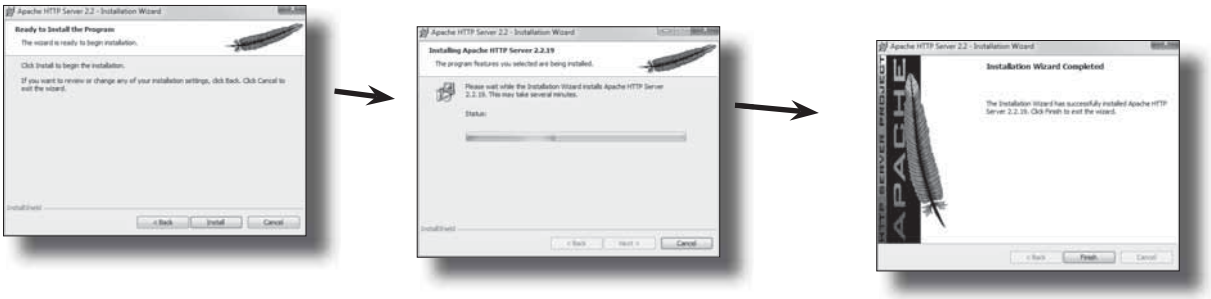


Обычно для установки программного продукта можно выбрать каталог по умолчанию.



Установка Apache... завершение

Работа почти закончена. Щелкните на кнопке **Install** и подождите пару минут. Вот и все!



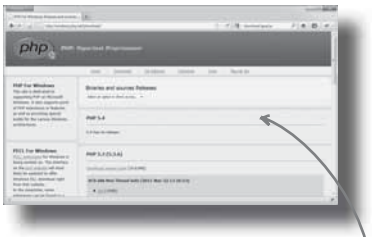
Веб-сервер настраивается на автоматический запуск при загрузке системы. Впрочем, вы можете останавливать и запускать его по своему усмотрению в диалоговом окне **Панель управления ▸ Администрирование ▸ Службы**. Веб-сервер находится в списке под именем **Apache2.2**.

Если эти инструкции вам не подошли, повторите попытку или введите строку «установка Apache в Windows» в поисковой системе.

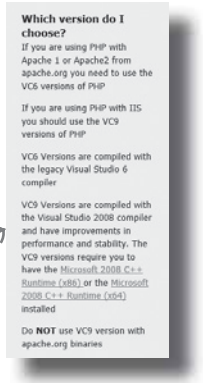
Установка PHP

Если вы используете Windows, откройте страницу *http://www.php.net/downloads.php* или *http://windows.php.net/download*.

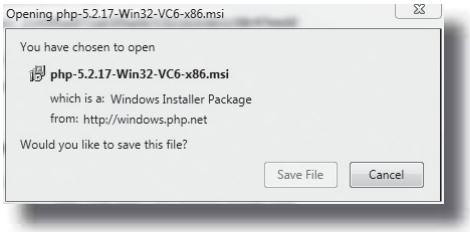
Для Windows мы также рекомендуем загрузить соответствующую версию с автоматической установкой. Пользователи Apache загружают файл *php-5.2.17-Win32-VC6-x86.msi*, а пользователи IIS — файл *php-5.3.6-Win32-VC9-x86.msi*. Загрузите файл и двойным щелчком начните автоматическую установку PHP.



Раздел загрузки версий для Windows в формате .msi



Прочитайте описание и выберите версию.

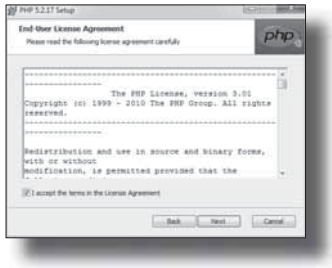


Загрузите файл и сделайте на нем двойной щелчок. Кнопка **Run** начинает установку.

Действия по установке PHP

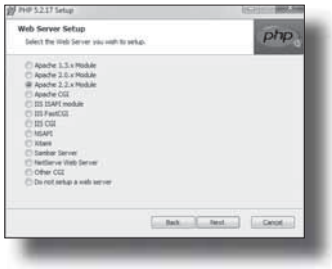
Для начала необходимо принять несколько тривиальных решений.

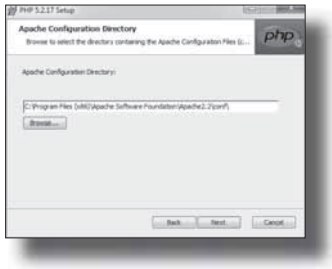
Примите условия лицензионного соглашения, чтобы продолжить установку.



Обычно PHP рекомендуется устанавливать в папку по умолчанию, но вы можете указать другую папку.

Будьте внимательны. Если вы используете Apache, выберите правильную версию. Если вы используете ISAPI, то, скорее всего, следует выбрать модуль IISAPI. За дополнительной информацией обращайтесь к документации. В данном примере был выбран веб-сервер Apache 2.2, а на следующем экране вводится путь к установленному экземпляру Apache.



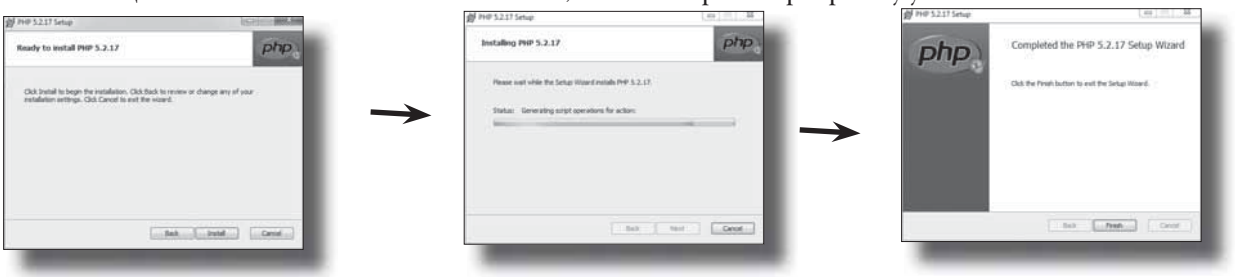


Прокрутите список **Extensions** и выберите **MySQL**. Это позволит использовать встроенные функции PHP по работе с MySQL, использованные в книге!

Прокрутите список в категории Extensions и щелкните на пункте MySQL. Выберите вариант «Entire feature».

Действия по установке PHP... завершение

Вот и все! Щелкните на кнопках **Install** и **Done**, чтобы закрыть программу установки.



Создайте новый сценарий с именем *info.php* и включите в него команду:

```
<?php phpinfo(); ?>
```

Сохраните файл в каталоге своего веб-сервера. В системе Windows это обычно папка

C:\inetpub\wwwroot (для IIS) или

C:\Program Files (x86)\Apache Software Foundation\Apache2.2\htdocs (для Apache). На Mac обычно используется */Users/yourname/sites/*.

Откройте файл в браузере: http://localhost/info.php,

Если поддержка PHP была установлена успешно, результат должен выглядеть примерно так:

Если инструкции вам не подошли, повторите попытку или введите «установка PHP для Apache [или IIS] в Windows» в поисковой системе.

Установка MySQL

Инструкции и решение проблем

Также вам потребуется сервер баз данных MySQL; давайте разберемся, как загрузить и установить его. Бесплатная версия РСУБД MySQL сейчас официально называется **MySQL Community Server**.

Ниже приведено описание последовательности действий по установке MySQL в Windows и Mac OS X. Эта краткая сводка не заменит превосходных инструкций с сайта MySQL; мы **настоятельно рекомендуем посетить сайт и прочитать их!** За более подробным описанием и рекомендациями по решению проблем обращайтесь по адресу:

Загрузите версию 5.5 и выше.

http://dev.mysql.com/doc/refman/5.5/en/windows-installation.html

Также вам пригодится программа просмотра запросов MySQL, которая позволяет ввести запрос и ознакомиться с результатами в интерфейсе приложения (вместо консольного окна).

Установка MySQL в системе Windows

1. **Откройте страницу:**
   http://dev.mysql.com/downloads/

   И щелкните на кнопке «**Download the Beta**» в разделе **MySQL Installer for Windows**. (Примечание: на момент написания книги существовала только бета-версия.)

2. Выберите в списке **Microsoft Windows**.

Немного прокрутите вниз.

Загрузка программы установки

3. Выберите в списке **Windows(x86, 32-bit), MSI Installer**.

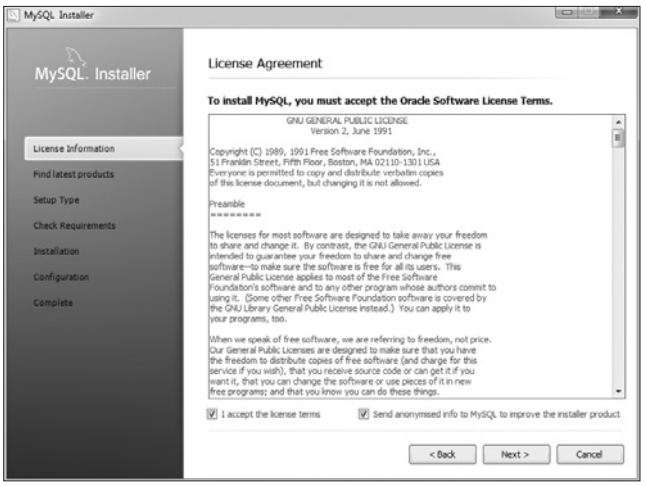
Щелкните на ссылке **No thanks, just take me to the downloads!** (если только вы не хотите зарегистрироваться на сайте или у вас уже имеется учетная запись).

4. На странице выводится список серверов, на которых доступна нужная версия продукта. Выберите ближайший к вам сервер.

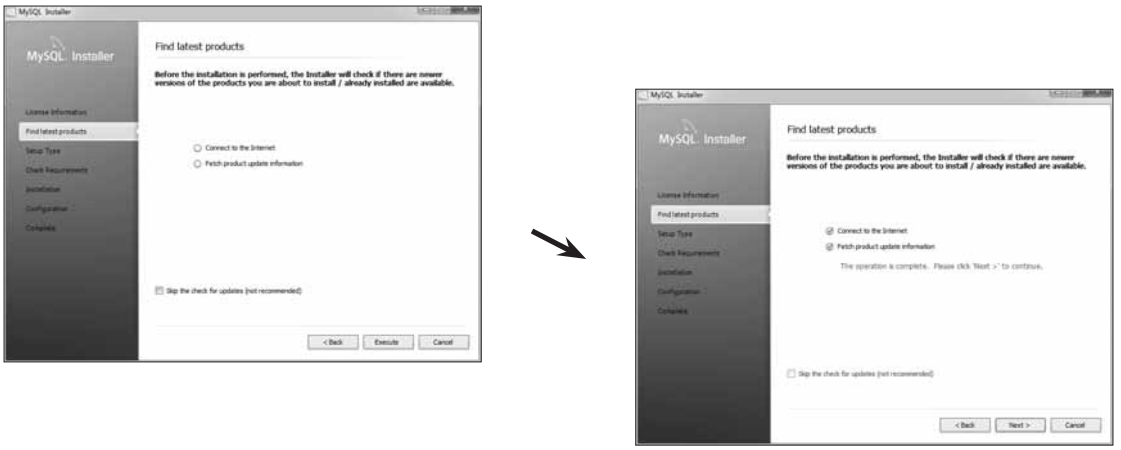
5. Когда файл будет загружен, щелкните на нем правой кнопкой мыши и выберите команду «Запуск от имени администратора» (если у вас включен механизм Windows UAC). Дальнейшая установка будет осуществляться под руководством мастера установки. Щелкните на кнопке **Next**.

Выбор папки для установки

6 Прочитайте текст лицензии, согласитесь на ее условия и щелкните на кнопке **Next**.

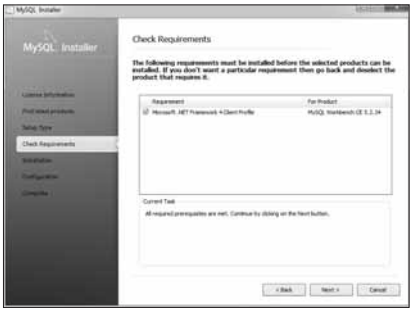


7 На следующем шаге выполняется автоматическая проверка на наличие новых версий. Если вы хотите пропустить ее, установите флажок Skip Check. Лучше, чтобы приложение устанавливалось в самой последней версии. После обновления нажмите **Next**, чтобы продолжить установку.

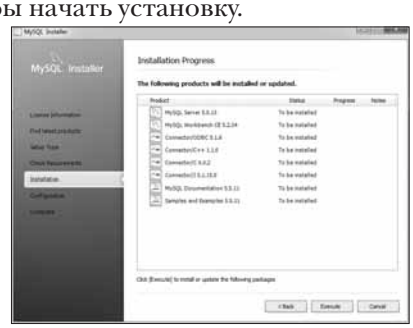


8. Вам будет предложено выбрать тип установки. Для наших целей идеально подойдет стандартный вариант для разработчиков (**Developer Default**). Оставьте пути установки, предложенные по умолчанию, и нажмите **Next**.

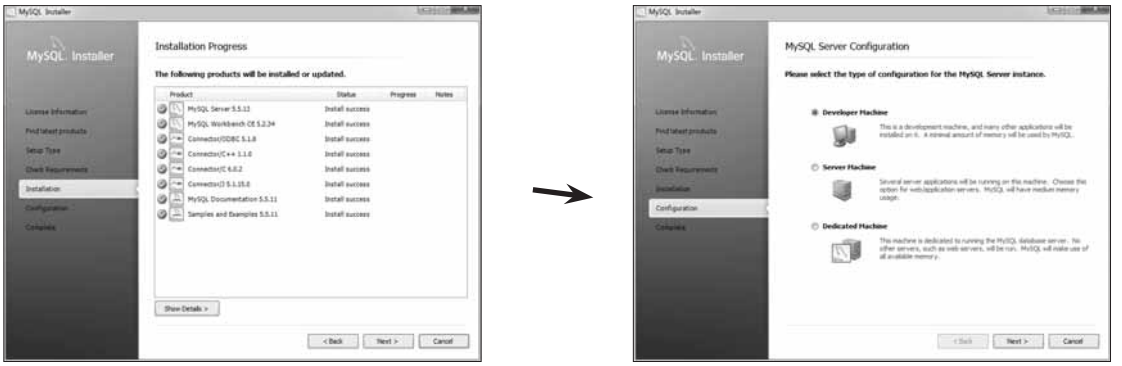
9. Программа установки проверяет совместимость с Microsoft .NET Framework 4 Client Profile. Это необходимо для приложения MySQL Workbench. Если проверка не проходит, обновите Windows на сайте *http://update.microsoft.com/*.



10. В следующем окне перечислены все устанавливаемые компоненты. Щелкните на кнопке **Execute**, чтобы начать установку.



11 После подтверждения успешной установки всех компонентов нажмите **Next**, чтобы перейти к настройке конфигурации службы MySQL. Установите переключатель **Developer Machine** и нажмите **Next**.



12 Убедитесь в том, что флажки **Enable TCP/IP Networking и Create Windows Service** установлены. Оставьте в других полях значения по умолчанию. Введите пароль привилегированного пользователя MySQL, нажмите **Next**.

13 Установка завершается. Если приложение MySQL Workbench не запустилось автоматически, запустите его командой меню **Пуск ▸ Все программы ▸ MySQL**.

Включение поддержки PHP в Mac OS X

Поддержка PHP входит в Mac OS X версии 10.5+ (Leopard), но по умолчанию она заблокирована. Чтобы включить ее, необходимо открыть основной файл конфигурации Apache *http.conf* и убрать комментарий в одной строке кода. Этот файл находится в папке установки Apache.

Найдите строку кода, которая начинается с «решетки» (#) — признака комментария:

```
#LoadModule php5_module        libexec/apache2/libphp5.so
```

Удалите «решетку» и перезапустите сервер, чтобы активизировать поддержку PHP. Владельцем файла *http.conf* является пользователь «root», а это означает, что для его изменения вам придется ввести свой пароль. Вероятно, вам также придется внести изменения в файл *php.ini*. За подробной информацией о том, как выполнить эти действия и включить поддержку PHP, обращайтесь по адресу *http://foundationphp.com/tutorials/php_leopard.php*.

Установка MySQL в Mac OS X

Если вы используете Mac OS X Server, в вашей системе уже должна быть установлена версия MySQL.

Прежде чем продолжать, проверьте, нет ли в вашей системе установленного экземпляра MySQL (выберите команду *Applications/Server/MySQL Manager*).

1. **Откройте страницу:**
   http://dev.mysql.com/downloads/

   и щелкните на ссылке **MySQL Community Server**.

   В этом описании используется 32-разрядная версия. Убедитесь в том, что загружаемая версия соответствует вашей операционной системе.

2 Щелкните на кнопке Download для версии **Mac OS X v10.6 (x86, 32-bit), DMG Archive**.

Чтобы найти ее, список придется прокрутить!

3 Щелкните на ссылке **No thanks, just take me to the downloads!** (если только вы не хотите зарегистрироваться на сайте или у вас уже имеется учетная запись).

Продолжить без регистрации.

Для ускорения загрузки выберите сервер, расположенный ближе всего к вам.

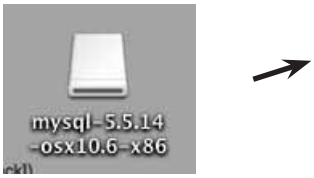
Загрузка установочного пакета

4 **Вернитесь на страницу:**
http://dev.mysql.com/downloads/

и щелкните на ссылке **MySQL Workbench**.

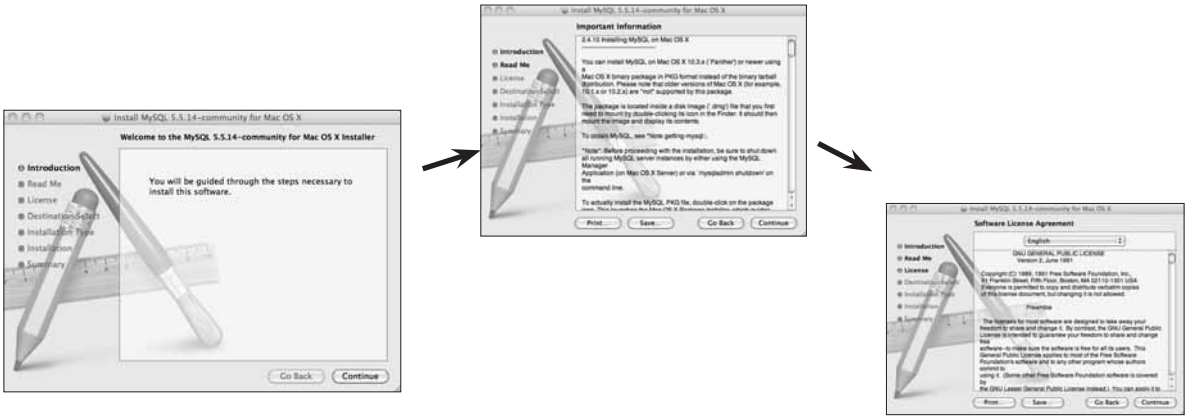
Щелкните на ссылке **No thanks, just take me to the downloads!** (если только вы не хотите зарегистрироваться на сайте или у вас уже имеется учетная запись) и снова выберите зеркальный сервер для загрузки.

5 Когда загрузка обоих файлов будет завершена, сделайте двойной щелчок на файле *mysql-5.5.14-osx10.6-x86.dmg*, а затем на файле *mysql-5.5.14-osx10.6-x86.pkg*, чтобы запустить программу установки пакета.

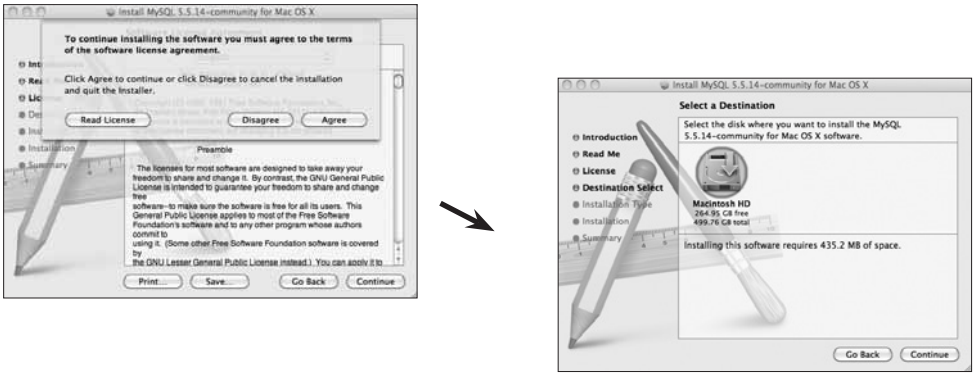


Запуск установки пакета

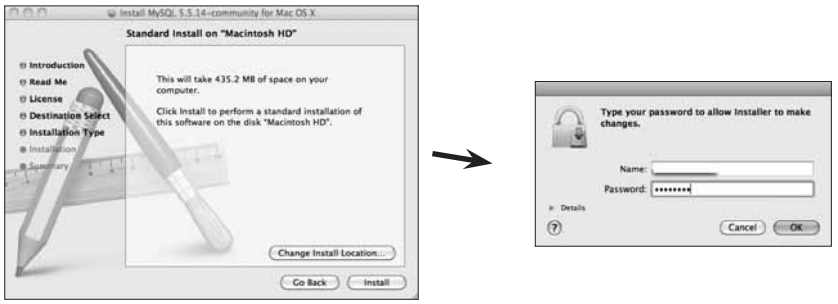
6 Начинается установка. Нажмите **Next**, чтобы перейти к странице **Read Me**, а затем нажмите **Continue**, чтобы перейти к странице с лицензионным соглашением.



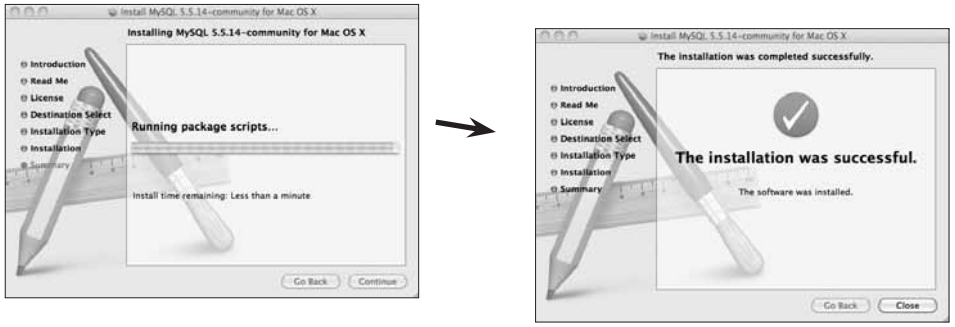
7 В следующем окне выводится текст лицензионного соглашения MySQL. Если вы согласны на его условия, щелкните на кнопке **Continue**, а затем на кнопке **Agree**. Снова щелкните на кнопке **Continue**, чтобы установить MySQL в каталог по умолчанию.



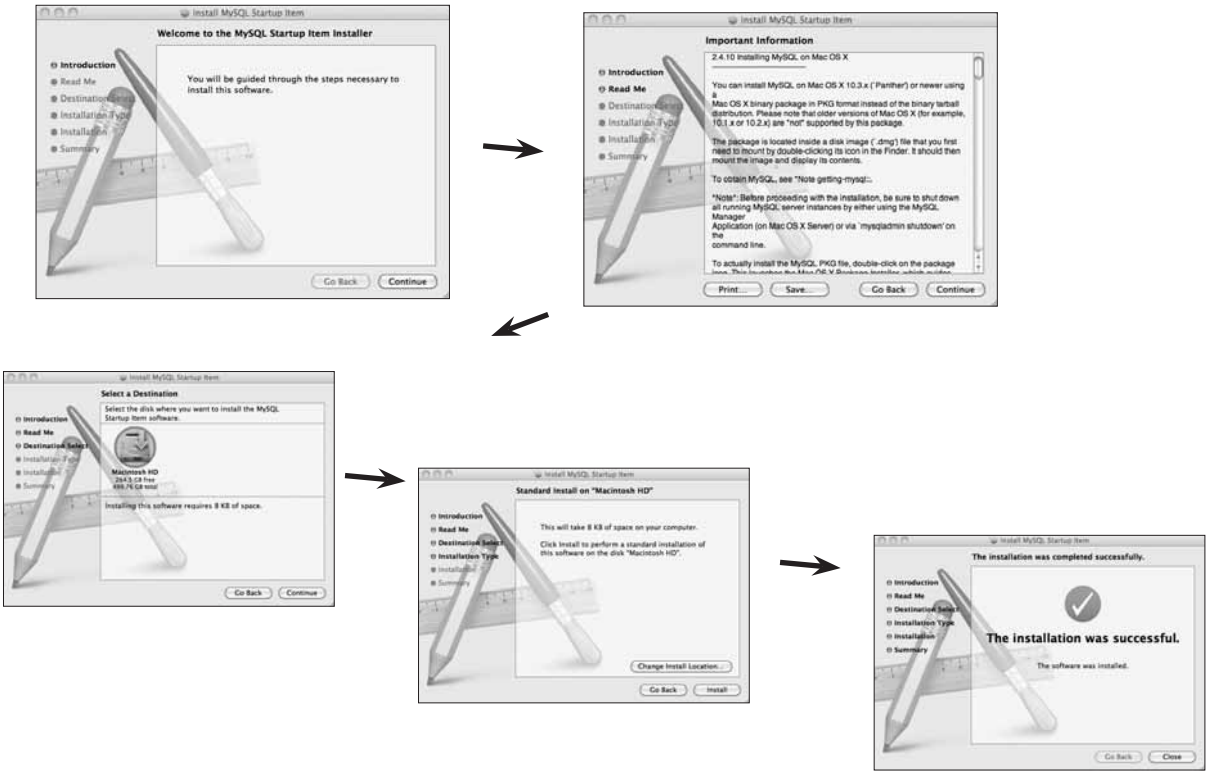
8 Щелкните на кнопке **Install**, введите имя и пароль администратора и щелкните на кнопке **OK**, чтобы начать установку.



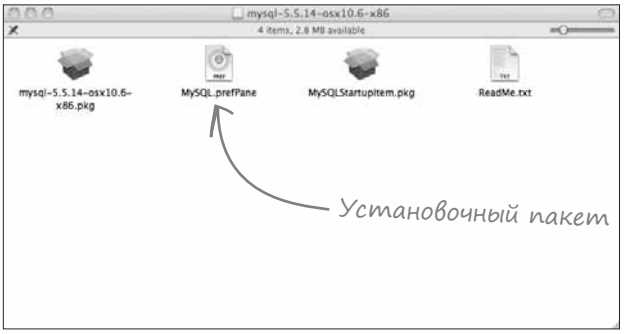
Через некоторое время появляется сообщение об успешном завершении установки.



9 Повторите те же действия для файла *MySQLStartupItem.pkg*.



10 Сделайте двойной щелчок на файле **MySQL.prefPane**, также входящем в образ *mysql-5.5.14-osx10.6-x86.dmg*, для установки приложения MySQL Preference Pane; затем щелкните на кнопке **Start MySQL Server**.

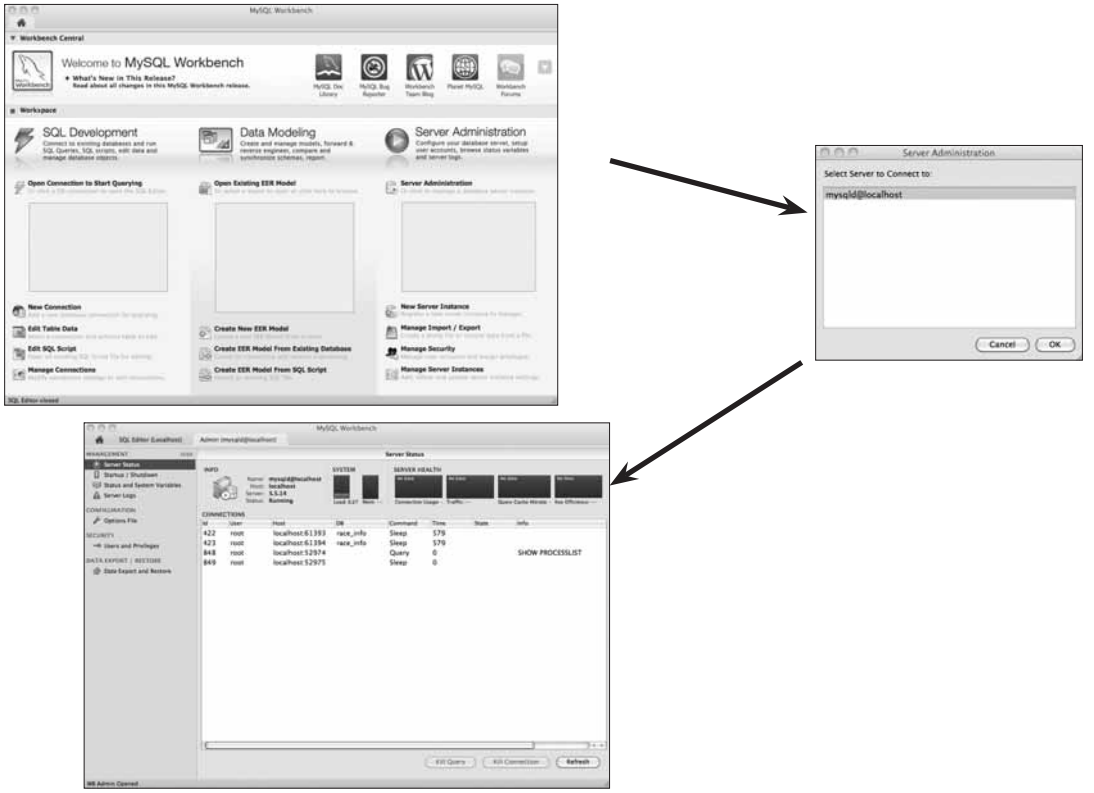


11 Сделайте двойной щелчок на файле *mysql-workbench-gpl-5.2.34.osx-i686.dmg* (который также был загружен ранее), чтобы запустить процесс установки программы MySQL Workbench.

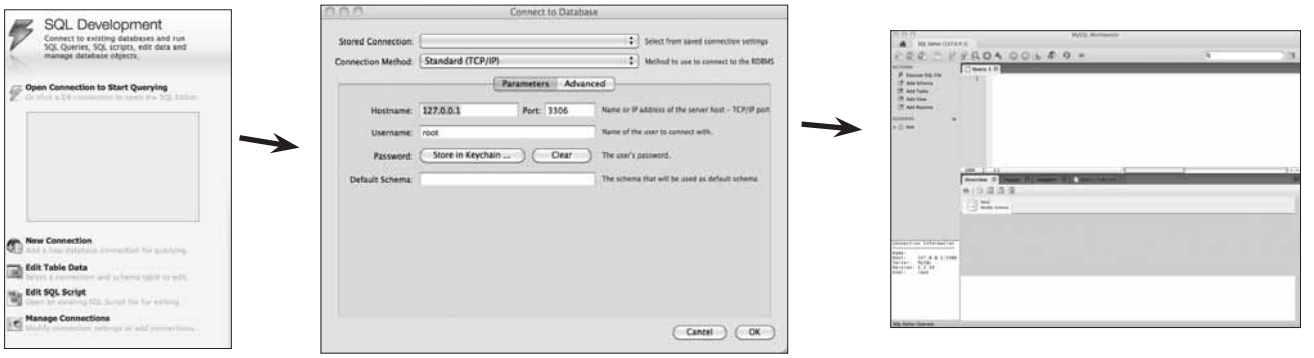
Перетащите значок *MySQLWorkbench.app* в папку *Applications*.

Запустите программу Workbench из списка приложений.

12 Убедитесь в том, что сервер работает (панель **Server Administration**). Если панель пуста, щелкните на ссылке **New Server Instance** и выберите сервер.



13 Создайте новое подключение в разделе SQL Development: щелкните на ссылке **New Connection** и введите данные в открывшемся окне. Затем сделайте двойной щелчок на новом подключении, чтобы открыть его.



За дополнительной информацией о MySQL и MySQL Workbench обращайтесь по адресу *http://dev.mysql.com/doc/*.

Райан Бенедетти, Ронан Крэнли

Изучаем работу с jQuery

Перевел с английского Е. Матвеев

Заведующий редакцией	*А. Кривцов*
Руководитель проекта	*А. Юрченко*
Ведущий редактор	*Ю. Сергиенко*
Литературный редактор	*Б. Файзуллин*
Художественный редактор	*Л. Адуевская*
Корректор	*В. Ганчурина*
Верстка	*Е. Леля*

ООО «Мир книг», 198206, Санкт-Петербург, Петергофское шоссе, 73, лит. А29.
Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2;
95 3005 — литература учебная.
Подписано в печать 16.05.12. Формат 84x108/16. Усл. п. л. 55,440. Тираж 2000. Заказ
Отпечатано в ОАО «Первая Образцовая типография», филиал «Дом печати - ВЯТКА»
в полном соответствии с качеством предоставленного оригинал-макета.
610033, г. Киров, ул. Московская, 122
Факс: (8332) 53-53-80, 62-10-36