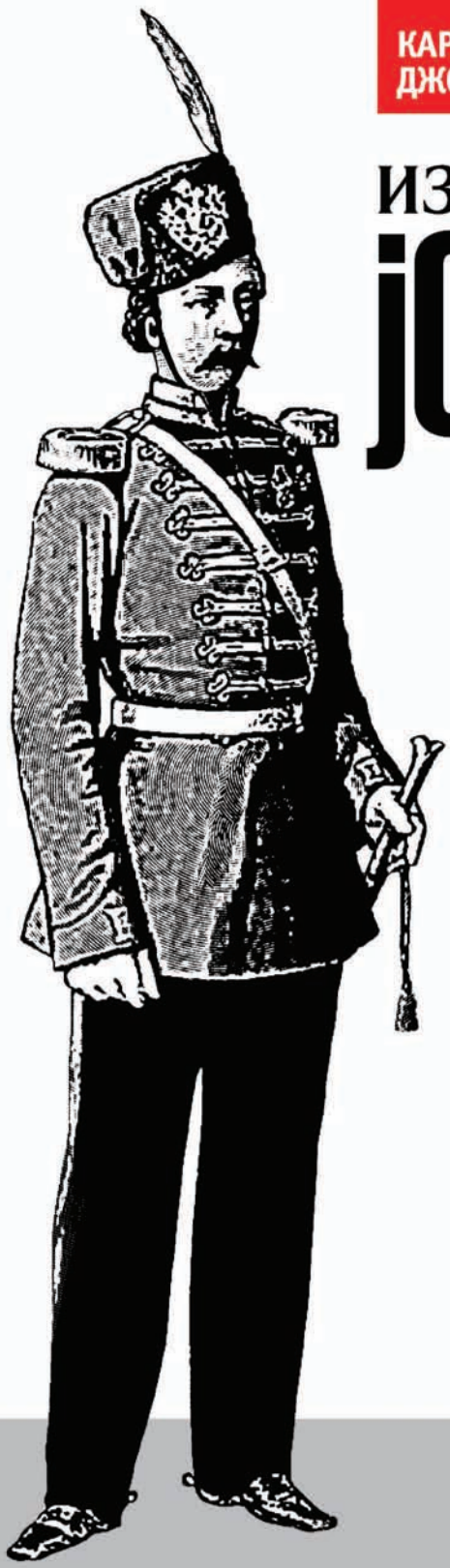
HIGH TECH

КАРЛ ШВЕДБЕРГ
ДЖОНАТАН ЧАФФЕР

ИЗУЧАЕМ jQuery 1.3

ЭФФЕКТИВНАЯ ВЕБ-РАЗРАБОТКА НА JAVASCRIPT



СИМВОЛ®

Learning jQuery 1.3

Better Interaction Design and Web Development with Simple JavaScript Techniques

Jonathan Chaffer, Karl Swedberg

[PACKT] PUBLISHING

HIGH TECH

Изучаем jQuery 1.3

Эффективная веб-разработка на JavaScript

Джонатан Чаффер, Карл Шведберг

СИМВОЛ®

Санкт-Петербург — Москва
2010

Серия «High tech»

Джонатан Чаффер, Карл Шведберг

Изучаем jQuery 1.3
Эффективная веб-разработка на JavaScript

Перевод А. Киселева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*П. Щеголев*
Редактор	*О. Меркулова*
Корректор	*С. Минин*
Верстка	*К. Чубаров*

Чаффер Дж., Шведберг К.

Изучаем jQuery 1.3. Эффективная веб-разработка на JavaScript. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 448 с., ил.

ISBN 978-5-93286-177-6

Издание, посвященное jQuery версии 1.3, знакомит с основами использования этой библиотеки для создания привлекательных интерактивных сайтов. jQuery поможет автоматизировать решение типичных задач и упростить решение более сложных. Опытные веб-дизайнеры, немного знакомые с программированием, смогут быстро приступить к использованию jQuery благодаря тому, что она основана на стандартах технологий HTML и CSS. Опытные программисты при изучении библиотеки оценят ее концептуальную целостность.

В книге рассматриваются методы использования селекторов, приемы организации взаимодействий и воспроизведения анимационных эффектов. Показано, как избежать ошибок, связанных с использованием AJAX, событий и расширенных возможностей языка JavaScript.

Издание предназначено для веб-дизайнеров, желающих использовать интерактивные элементы в своих страницах, и разработчикам, стремящимся создавать веб-приложения с более качественным пользовательским интерфейсом. Опыт работы с jQuery и другими библиотеками JavaScript не требуется, однако приветствуются навыки программирования на языке JavaScript, знание его синтаксических конструкций, а также базовые знания о HTML и CSS.

ISBN 978-5-93286-177-6
ISBN 978-1-847196-70-5 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Подписано в печать 21.12.2009. Формат 70×100 1/16. Печать офсетная.
Объем 28 печ. л. Тираж 1500 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Вступительное слово

Я был горд, когда узнал, что Карл Шведберг (Karl Swedberg) и Джонатан Чаффер (Jonathan Chaffer) взяли на себя труд по созданию книги «Изучаем jQuery 1.3». Эта книга во многом определила стандарт, которому стремятся соответствовать другие книги об этой платформе, а также вообще книги о JavaScript. Издание, безусловно, стало одной из наиболее продаваемых книг о JavaScript в немалой степени благодаря своему качеству и вниманию к деталям.

Я был особенно рад, что за написание книги принялись именно Карл и Джонатан, поскольку уже был знаком с ними и знал, что они прекрасно справятся с поставленной задачей. Являясь членом основной команды разработчиков jQuery, я имел возможность достаточно хорошо узнать Карла за последние пару лет, в частности в ходе работы над его книгой. Глядя на конечный результат, можно без сомнения сказать, что его навыки разработчика и бывшего учителя английского языка идеально подошли для выполнения непростого задания.

У меня также была возможность лично встретиться с обоими авторами, что само по себе большая редкость в мире распределенных проектов с открытыми исходными текстами. Как тогда, так и сейчас Карл и Джонатан являются активными членами сообщества jQuery.

Библиотека jQuery используется различными специалистами. Членами сообщества являются дизайнеры, разработчики, люди, как обладающие опытом программирования, так и не имеющие его. В состав команды jQuery входят люди с разными уровнями подготовки, которые своими отзывами способствуют определению дальнейшего пути развития проекта. Тем не менее всех пользователей jQuery объединяет одна общая цель: постараться максимально упростить разработку веб-приложений на JavaScript.

В настоящее время можно смело утверждать, что любой открытый проект ориентирован на поддержку сообщества или на оказание помощи начинающим пользователям. Но для jQuery это не просто пустые слова, это движущая сила проекта. В действительности, в команде проекта jQuery намного больше людей, занимающихся поддержкой сообщества, написанием документации или расширений, чем тех, кто сопровождает базовый программный код. Хотя библиотека сама по себе чрезвычайно важна, сообщество, окружающее этот программный код, может служить критерием, отличающим еле живой, посредственный проект от проекта, наилучшим образом удовлетворяющего все ваши потребности.

Принцип разработки проекта и порядок использования его программного кода принципиально различаются для большинства открытых проектов и большинства библиотек JavaScript. Участники проекта jQuery и его сообщество обладают широчайшими познаниями; мы понимаем, что разрабатываем jQuery для пользователей с разным опытом программирования, поэтому прилагаем максимальные усилия, чтобы передать эти знания всем пользователям.

Чтобы понять, что собой представляет сообщество jQuery, недостаточно просто прочитать о нем; необходимо личное участие в сообществе, чтобы его прочувствовать. Я надеюсь, у вас будет возможность принять в нем участие. Приходите к нам в форумы и блоги, подписывайтесь на рассылки, и мы поможем вам приобрести более глубокие знания об этой библиотеке.

Для меня jQuery означает гораздо больше, чем блок программного кода. Это совокупность опыта, накопленного на протяжении многих лет в процессе разработки библиотеки. Серьезные взлеты и падения, трудности совершенствования и захватывающее ощущение роста и успеха. Профессиональный рост вместе с пользователями и коллегами по команде, улучшение взаимопонимания между нами, усилия по дальнейшему развитию и адаптации проекта.

Когда я впервые увидел эту книгу, рассказывающую о jQuery как о едином инструменте, в противоположность моим собственным представлениям, я был удивлен и восхищен одновременно. Наблюдение за тем, как другие изучают, осмысливают и применяют jQuery к своим задачам, во многом делает этот проект таким захватывающим.

Я не единственный, кто пользуется этой библиотекой на уровне, существенно отличающемся от уровня взаимодействия обычного пользователя с инструментом. Не уверен, что смогу должным образом описать свои ощущения в тот момент, когда я в очередной раз наблюдал, как загораются глаза пользователей от осознания того, как много может предложить им jQuery.

В какую-то секунду в головах пользователей jQuery раздается щелчок, и они начинают понимать, что эта библиотека представляет собой нечто гораздо большее, чем простой инструмент, и неожиданно их взгляды на разработку динамических веб-приложений полностью меняются. Это самая потрясающая и, безусловно, моя самая любимая особенность проекта jQuery.

Я надеюсь, что у вас тоже появится возможность испытать это поразительное ощущение.

Джон Резиг (John Resig),
создатель библиотеки jQuery

Об авторах

Джонатан Чаффер (Jonathan Chaffer) – технический директор интерактивного подразделения в рекламном агентстве, находящемся в городе Гранд-Рапидс, штат Мичиган. Он руководит разработкой веб-проектов, использующих разнообразные технологии, а также сам продолжает заниматься повседневным программированием.

Чаффер принимал весьма активное участие в разработке открытого проекта Drupal CMS, где в качестве предпочтительной платформы JavaScript предлагалась библиотека jQuery. Он является автором популярного модуля Content Construction Kit, используемого для управления структурированным содержимым сайтов, построенных на основе системы Drupal. Джонатан отвечает за полный пересмотр системы меню Drupal и справочного руководства разработчика по прикладному интерфейсу системы.

Карл Шведберг (Karl Swedberg) – веб-программист в компании Fusionary Media в городе Гранд-Рапидс, штат Мичиган; большую часть времени он занимается веб-дизайном с упором на «веб-стандарты» – применение HTML-кода совместно с продуманными таблицами стилей CSS и «ненавязчивым» JavaScript. Будучи членом команды проектирования jQuery и активным участником обсуждений jQuery, Карл посещает симпозиумы и конференции и проводит корпоративное обучение в Европе и Северной Америке.

Прежде чем Карл занялся разработкой веб-приложений, он был литературным редактором, учителем английского языка в средней школе, владельцем кафе. Его увлечение программированием началось в начале 1990-х, когда он работал в корпорации Microsoft в городе Редмонд, штат Вашингтон, и не угасает по сей день.

Благодарности

Я хотел бы поблагодарить Дженни за ее неугасимый энтузиазм и поддержку, Карла – за стимул продолжать работу над книгой, когда дух ослабевал, а также сообщество Ars Technica – за постоянное вдохновение к техническому совершенствованию.

Джонатан Чаффер

Я благодарен моей супруге Саре за ее неустанную любовь и поддержку. Спасибо также моим милым деткам, Бенджамину и Люсии. Выражаю Джонатану Чафферу свое глубочайшее восхищение его опытом программирования и благодарность за его готовность написать эту книгу вместе со мной.

Большое спасибо Джону Резигу за создание лучшей в мире библиотеки JavaScript и за взращивание вокруг нее такого удивительного сообщества. Спасибо также сотрудникам издательства Packt Publishing, техническим редакторам этой книги, коллективу jQuery Cabal и всем тем, кто помогал и вдохновлял нас на всем протяжении работы над книгой.

Карл Шведберг

О технических редакторах

Акаш Мехта (Akash Mehta) – веб-программист, технический писатель и бизнес-консультант, живущий в городе Брисбен, Австралия. В число его разработок входят веб-сайты брошюрного типа, системы электронного обучения и информационные комплексы. Он пишет статьи о разработке веб-приложений для издательств, выпускающих продукцию как в печатном, так и в электронном виде, регулярно выступает на местных конференциях и оказывает поддержку нескольким видным блогам PHP.

Будучи студентом, Акаш занимался поддержкой веб-приложений на языке PHP и созданием пользовательских интерфейсов с помощью инструментария jQuery. Наряду с деятельностью, нацеленной на получение ученой степени в сферах коммерции и информационных технологий, Акаш продолжает разрабатывать веб-приложения на языках PHP и Python. Во внерабочее время он руководит организацией своей местной группы пользователей PHP.

При создании веб-приложений Акаш использует различные библиотеки, распространяемые с открытыми исходными текстами. Его арсенал содержит несколько платформ разработки приложений, включая Zend Framework, CakePHP и Django; платформы JavaScript, такие как jQuery, Prototype и Mootools; другие платформы, такие как Adobe Flash/Flex, и механизмы баз данных MySQL и SQLite.

В настоящее время Акаш предоставляет услуги по созданию документации и разработке веб-приложений на своем веб-сайте *http://bitmeta.org*.

Дейв Метвин (Dave Methvin) обладает более чем 25-летним опытом разработки программного обеспечения для операционных систем Windows и UNIX. В начале своей карьеры он занимался созданием встроенного программного обеспечения в области робототехники, телекоммуникаций и медицины. Позднее он переориентировался на разработку программного обеспечения для персональных компьютеров с использованием языка C/C++ и веб-технологий.

Дейв имеет также 20-летний опыт работы в компьютерной журналистике. Он был ответственным секретарем в журналах «PC Tech Journal» и «Windows Magazine», где занимался освещением вопросов, связанных с персональными компьютерами и Интернетом; его колонки о JavaScript содержали ценные решения типичных задач

веб-программирования. Помимо этого он является соавтором книги «Networking Windows NT» (John Wiley & Sons, 1997).

В настоящее время Дейв занимает должность главного технического директора веб-сайта PC Pitstop, помогающего пользователям оптимизировать производительность их компьютеров. Он также принимает активное участие в жизни сообщества jQuery.

Майк Алсуп (Mike Alsup) примкнул к проекту jQuery почти с самого начала и передал сообществу множество пользующихся популярностью модулей расширения. Он является активным участником группы jQuery Google Group, в рамках которой часто оказывает помощь начинающим пользователям jQuery.

Майк живет в штате Нью-Йорк, работает ведущим программистом в компании Click Commerce, Inc., где занимается разработкой веб-приложений и основное внимание уделяет языку Java и библиотеке Swing.

Созданные им модули расширения для jQuery можно найти по адресу: *http://jquery.malsup.com/*.

Предисловие

Проект был запущен в 2005 году Джоном Резигом, вундеркиндом JavaScript, который в настоящее время работает в корпорации Mozilla. Вдохновленный достижениями пионеров в этой области, таких как Дин Эдвардс (Dean Edwards) и Саймон Уиллисон (Simon Willison), Резиг создал библиотеку функций, упрощающих поиск элементов веб-страниц и реализацию их поведения. К тому моменту, когда он во всеуслышание объявил о своем проекте в январе 2006 года, им были добавлены в библиотеку функции, позволяющие манипулировать деревом DOM и воспроизводить простые анимационные эффекты. Резиг дал своему проекту название jQuery, чтобы подчеркнуть главенствующее положение функций поиска, или «запросов» (querying), элементов веб-страниц для последующего воздействия на них из программного кода на JavaScript. Спустя несколько лет были существенно расширены возможности библиотеки jQuery, улучшена ее производительность, и она нашла применение на некоторых наиболее популярных сайтах в Интернете. Хотя Резиг и остается ведущим разработчиком, jQuery продолжает развиваться как настоящий свободный проект и может сегодня похвастаться командой первоклассных разработчиков на JavaScript, а также энергичным сообществом, насчитывающим тысячи программистов.

Библиотека jQuery способна повысить качество ваших веб-сайтов независимо от вашего уровня подготовки. Она предоставляет широкий круг возможностей, имеет простой синтаксис и обеспечивает устойчивую совместимость с различными платформами, размещаясь при этом в одном компактном файле. Более того, существуют сотни модулей, расширяющих функциональность библиотеки jQuery, что делает ее весьма ценным инструментом для решения практически любых задач на стороне клиента.

Книга «Изучаем jQuery 1.3» представляет собой постепенное и пошаговое введение в концепции jQuery, позволяющее вам повысить интерактивность ваших страниц и добавить в них анимационные эффекты, даже если ваши предыдущие попытки использовать JavaScript были неудачными. Эта книга поможет вам избежать ошибок, связанных с использованием технологии AJAX, событиями, эффектами и дополнительными особенностями языка JavaScript. Кроме того, она содержит краткий справочник по библиотеке jQuery, к которому можно будет возвращаться снова и снова.

О чем рассказывается в этой книге

В *главе 1* вы начнете свое знакомство с библиотекой jQuery. Здесь дается описание jQuery и ее возможностей, рассказывается, как загрузить и установить библиотеку, а также демонстрируется первый сценарий.

В *главе 2* вы познакомитесь с выражениями-селекторами jQuery и методами обхода дерева DOM, выполняющими поиск элементов страницы независимо от их местоположения. В этой главе вы узнаете, как с помощью jQuery можно оказывать влияние на стили отображения различных типов элементов, причем в некоторых случаях способом, недоступным при использовании каскадных таблиц стилей CSS.

В *главе 3* вы научитесь использовать механизм jQuery обработки событий для реализации реакции элементов в ответ на события, возникающие в броузере. Вы увидите, насколько просто с помощью jQuery производится подключение обработчиков событий к элементам, даже когда страница еще не загружена полностью. Вашему вниманию будут представлены такие сложные темы, как всплытие событий, делегирование и пространства имен.

В *главе 4* вы познакомитесь с приемами воспроизведения анимационных эффектов средствами jQuery и увидите, как скрывать, отображать и перемещать элементы страниц с применением эффектов, полезных и приятных для глаз.

В *главе 5* вы узнаете, как изменять страницу по команде. Эта глава научит вас «на лету» изменять структуру документа HTML и его содержимое.

В *главе 6* вы познакомитесь с разнообразными предоставляемыми библиотекой jQuery способами, упрощающими доступ к функциональным возможностям на стороне сервера и не прибегающими при этом к полному обновлению страницы.

В следующих трех главах (7, 8 и 9) будет представлено несколько реальных примеров, использующих все, о чем рассказывалось в предыдущих главах, и обеспечивающих надежные решения часто встречающихся проблем.

В *главе 7* «Манипулирование таблицами» вы научитесь выполнять сортировку, отбор и стилизацию информации для создания красивого и функционального представления данных.

В *главе 8* «Интерактивные формы» вы овладеете тонкостями проверки данных на стороне клиента, создания адаптивных схем размещения элементов форм и приемов реализации интерактивных особенностей поведения клиент-серверных форм, таких как функция автодополнения.

В *главе 9* «Прокрутка и перемещение» вы узнаете, как дополнить страницу возможностью управления отображением ее элементов. Вы на-

учитесь перемещать информацию за и в пределы области видимости исходя из логики работы программы или по требованию пользователя.

В главах 10 и 11 вы познакомитесь с модулями расширений к библиотеке, написанными сторонними разработчиками, и узнаете о различных способах создания своих собственных расширений.

В *главе 10* «Использование модулей расширения» вы займетесь исследованием модуля Form и официальной коллекции расширений элементов пользовательского интерфейса, известной как jQuery UI. Кроме того, здесь вы узнаете, где можно найти массу других популярных расширений для jQuery и посмотреть, какие особенности они могут предложить.

В *главе 11* «Разработка модулей расширения» вы узнаете, как использовать внушительные возможности библиотеки jQuery для создания с нуля собственных модулей расширения. Здесь вы научитесь создавать собственные вспомогательные функции, добавлять методы к объектам jQuery, определять свои селекторы и многое другое.

Приложение A «Ресурсы в Интернете» содержит перечень информационных веб-сайтов, где обсуждается широкий круг тем, имеющих отношение к jQuery, JavaScript и к разработке веб-приложений вообще.

В *приложении B* «Инструменты разработчика» вы познакомитесь с дополнительными программными продуктами, предназначенными для редактирования и отладки программного кода, использующего библиотеку jQuery, в вашей личной среде разработки.

В *приложении C* «Замыкания в JavaScript» вы получите четкое представление о замыканиях – что это такое и как их использовать.

В *приложении D* «Краткий справочник» кратко представлена вся библиотека jQuery со всеми ее методами и селекторами. Справочник имеет удобный формат, который позволяет легко отыскивать необходимую информацию о методе или селекторе, когда вы точно знаете, что он существует, но не уверены, как пишется его имя.

Что необходимо для работы с этой книгой

Для самостоятельной проработки примеров программного кода, демонстрируемых в этой книге, необходимо следующее:

- Простой текстовый редактор.
- Современный веб-броузер, такой как Mozilla Firefox, Apple Safari или Microsoft Internet Explorer.
- Исходный файл библиотеки jQuery версии 1.3.1 или выше, который можно загрузить с сайта *http://jquery.com/*.

Кроме того, для запуска примеров, использующих технологию AJAX, из главы 6 потребуется сервер с поддержкой PHP.

Для кого предназначена эта книга

Эта книга предназначена для веб-дизайнеров, желающих использовать интерактивные элементы в своих страницах, и для разработчиков, стремящихся создавать веб-приложения с более качественным пользовательским интерфейсом. Наличие базовых навыков программирования на языке JavaScript является обязательным условием. Вы должны обладать базовыми знаниями о языке разметки HTML и таблицах CSS и уверенно разбираться в синтаксических конструкциях JavaScript. Знание библиотеки jQuery и опыт работы с любыми другими библиотеками на JavaScript не требуются.

Типографские соглашения

В этой книге вам встретятся различные способы оформления текста, используемые для выделения информации различного типа. Ниже даются несколько примеров таких способов оформления и пояснения к ним.

Элементы программного кода в тексте оформляются следующим образом: «Подключение других контекстов может осуществляться с помощью директивы `include`».

Блоки программного кода оформляются, как показано ниже:

```
<html>
  <head>
    <title>the title</title>
  </head>
  <body>
    <div>
      <p>This is a paragraph.</p>
      <p>This is another paragraph.</p>
      <p>This is yet another paragraph.</p>
    </div>
  </body>
</html>
```

Чтобы привлечь внимание к некоторому фрагменту в блоке программного кода, соответствующие строки или элементы будут выделяться жирным шрифтом:

```
$(document).ready(function() {
  $('a[href^=mailto:]').addClass('mailto');
  $('a[href$=.pdf]').addClass('pdflink');
  $('a[href^=http][href*=henry]')
    .addClass('henrylink');
});
```

Данные ввода-вывода в командной строке оформляются, как показано ниже:

```
outerFn():
Outer function
Inner function
```

Новые термины и **важные слова** выделены жирным шрифтом. Элементы интерфейса, которые отображаются на экране, например в меню или в диалогах, выделяются в тексте рубленым шрифтом.

Примечание

Так выделяются предупреждения или важные примечания.

Совет

Так выделяются советы и рекомендации.

Обратная связь с читателями

Мы всегда приветствуем любые отзывы наших читателей. Дайте нам знать, что вы думаете об этой книге, что вам понравилось или, наоборот, не понравилось. Мнения читателей очень важны для нас, так как они помогают нам создавать книги, от которых вы сможете получать максимальную отдачу.

Чтобы передать нам свой отзыв, просто отправьте его на адрес электронной почты *feedback@packtpub.com*, не забудьте при этом указать название книги.

Если вы хотите, чтобы мы опубликовали какую-либо книгу, заполните форму SUGGEST A TITLE на странице *www.packtpub.com* или отправьте электронное письмо по адресу: *suggest@packtpub.com*.

Если вы обладаете серьезными познаниями в какой-либо области и хотели бы написать или предложить нам уже написанную вами книгу, обращайтесь на страницу с правилами оформления для авторов по адресу: *www.packtpub.com/authors*.

Поддержка покупателей

Теперь, когда вы являетесь владельцем книги, выпущенной издательством Packt, мы можем предложить еще кое-что, что позволит извлечь максимум из вашей покупки.

Программный код примеров из книги

По адресу *http://www.packtpub.com/files/code/6705_Code.zip* вы можете загрузить программный код примеров.

Загружаемые файлы содержат инструкции по их использованию.

Опечатки

Хотя мы приняли все меры для обеспечения точности содержимого книги, ошибки не исключаются полностью. Если вы обнаружите ошибку в одной из наших книг, будь то ошибка в тексте или в программном коде, мы будем весьма признательны, если вы сообщите нам об этом. Тем самым вы сможете уберечь других читателей от разочарования и помочь улучшить последующие издания этой книги. В случае обнаружения опечаток сообщите о них, посетив страницу *http://www.packtpub.com/support*, где следует выбрать нужную книгу, щелкнуть на ссылке let us know и подробно описать найденную опечатку. После проверки вашего сообщения, оно будет принято, и замеченная вами опечатка будет добавлена в список существующих опечаток. Список существующих опечаток можно просмотреть, выбрав название книги на странице *http://www.packtpub.com/support*.

Пиратство

Пиратское распространение авторских материалов в Интернете независимо от типа носителя по-прежнему остается насущной проблемой. Издательство Packt очень серьезно относится к защите своих авторских прав и лицензий. Если в Интернете вам встретятся нелегальные копии наших произведений в любом виде, пожалуйста, сразу же сообщите местоположение копии или название веб-сайта, чтобы мы могли предпринять необходимые меры.

Сообщения о подозрительных копиях вы можете направлять по адресу *copyright@packtpub.com*.

Мы будем вам благодарны за помощь в защите наших авторов и нашей возможности издавать для вас ценные книги.

Вопросы

В случае появления проблем, связанных с какими-либо аспектами книги, вы можете направлять свои вопросы по адресу *questions@packtpub.com*, и мы приложим все усилия, чтобы разобраться с ними.

1

Введение в jQuery

В наши дни Всемирная паутина представляет собой динамичную среду, и ее пользователи предъявляют высокие требования как к оформлению, так и к функциональности сайтов. Для создания интересных интерактивных сайтов разработчики используют библиотеки JavaScript, такие как jQuery, чтобы автоматизировать решение наиболее типичных задач и упростить решение более сложных. Одной из причин высокой популярности библиотеки jQuery является ее способность помогать при решении весьма широкого круга задач.

На первый взгляд кажется, что сложно выбрать, с чего начать, потому что библиотека jQuery реализует весьма широкие функциональные возможности. Тем не менее она имеет согласованную и симметричную архитектуру; большая часть концепций заимствована из **HTML** и **каскадных таблиц стилей (Cascading Style Sheets, CSS)**. Архитектура библиотеки быстро осваивается дизайнерами даже с малым опытом программирования, поскольку многие веб-разработчики, как правило, имеют бóльший опыт работы с указанными технологиями, чем с JavaScript. Так, в первой главе мы напишем действующую программу, использующую jQuery и содержащую всего три строчки кода. В свою очередь опытным программистам будет помогать и концептуальная целостность библиотеки, в чем мы убедимся в последующих более сложных главах.

Итак, посмотрим, что нам может предложить библиотека jQuery.

Что делает библиотека jQuery

Библиотека jQuery предоставляет многоцелевой уровень абстракции для решения типичных задач разработки веб-приложений и потому может применяться практически в любых ситуациях. Библиотека имеет расширяемую архитектуру; так как постоянно появляются новые расширения и добавляются новые возможности, в одной книге просто не

охватить все функции и допустимые случаи использования jQuery. Однако базовые возможности позволяют решать следующие задачи:

- **Доступ к элементам документа**. Чтобы выполнить обход дерева **объектной модели документа (Document Object Model, DOM)** и отыскать определенные фрагменты HTML без применения библиотеки JavaScript, пришлось бы написать уйму строк программного кода. Библиотека jQuery предлагает надежный и эффективный механизм селекторов, который позволяет извлекать требуемые фрагменты документа для последующего анализа и модификации.
- **Изменение внешнего вида страницы**. Каскадные таблицы стилей (CSS) предлагают мощный механизм определения внешнего вида документа, но он оказывается бесполезным в случае веб-броузеров, не поддерживающих единые стандарты. С помощью jQuery разработчики могут восполнить этот недостаток, опираясь на стандарты, поддерживаемые всеми броузерами. Кроме того, библиотека jQuery позволяет изменять классы или отдельные стилевые свойства, применяемые к фрагменту документа, даже после того, как он будет отображен.
- **Изменение содержимого документа**. Библиотека jQuery позволяет не только выполнять простые косметические изменения документа, но и дает возможность модифицировать его содержимое. С помощью одного удобного в использовании **прикладного программного интерфейса (Application Programming Interface, API)** можно изменять текст, вставлять или изменять изображения, переупорядочивать списки и даже полностью изменять и расширять структуру документа HTML.
- **Отклик на действия пользователя**. Даже самые тщательно разработанные и мощные реализации поведения становятся бесполезными, если отсутствует возможность управления моментом, когда они должны запускаться. Библиотека jQuery предлагает элегантный способ, дающий возможность перехватывать самые разнообразные события, такие как щелчок мышью на ссылке, и не загромождать при этом код разметки HTML обработчиками событий. Кроме того, прикладной интерфейс механизма обработки событий ликвидирует существующие между броузерами противоречия, которые часто вызывают чувство досады у веб-разработчиков.
- **Воспроизведение анимационных эффектов в документе**. Для эффективного взаимодействия пользователя с документом дизайнер должен обеспечить обратную визуальную связь. Библиотека jQuery содействует решению этой задачи, предоставляя множество анимационных эффектов, таких как растворение и стирание элементов, а также удобные инструментальные средства для реализации новых эффектов.
- **Извлечение информации со стороны сервера без полного обновления страницы**. Этот шаблон программирования известен как **асинхронный JavaScript и XML (Asynchronous JavaScript And XML,**

AJAX) и помогает веб-разработчикам создавать полнофункциональные и быстро реагирующие сайты. Библиотека jQuery скрывает сложности, связанные с несовместимостью броузеров, позволяя разработчикам сконцентрироваться на реализации функциональности на стороне сервера.

- **Упрощение решения типичных задач программирования на JavaScript**. В дополнение ко всем возможностям, связанным с документами, библиотека jQuery предоставляет расширения к базовым конструкциям JavaScript, таким как обход массивов в цикле и манипулирование ими.

Чем обусловлен успех jQuery

Новый всплеск интереса к динамическому HTML привел к возникновению большого числа платформ JavaScript. Одни из них являются узкоспециализированными инструментами, уделяющими внимание только одной или двум из указанных выше задач. Другие пытаются предложить полный спектр возможностей и анимационных эффектов и представляют собой комплекты всего, что только может потребоваться. Чтобы обеспечить широкий круг возможностей, обозначенных выше, и остаться при этом достаточно компактной, библиотека jQuery использует несколько стратегий:

- **Использование знаний о CSS**. Благодаря тому, что в основу механизма поиска элементов страницы были положены **селекторы CSS**, библиотека jQuery унаследовала краткий и понятный способ выражения структуры документа. Для дизайнеров, стремящихся обеспечить интерактивность своих страниц, библиотека jQuery стала отправной точкой, поскольку знание синтаксиса CSS является необходимым и обязательным условием профессиональной веб-разработки.
- **Поддержка расширений**. Во избежание «расползания» функциональных возможностей библиотека jQuery оставляет реализацию особых случаев за **модулями расширения**. Порядок создания новых модулей прост и подробно описан в документации, что способствует разработке разнообразных изобретательных и полезных модулей. Даже многие базовые функциональные возможности библиотеки jQuery реализованы с использованием механизма расширений и могут удаляться из библиотеки по мере необходимости, что позволяет еще сильнее уменьшать ее размер.
- **Абстрактный способ обхода несовместимостей броузеров**. Суровая действительность веб-разработки такова, что каждый броузер имеет свои отклонения от общепринятых стандартов. Значительную часть любого веб-приложения составляет реализация функциональных особенностей для каждой из платформ. Вследствие невозможности реализовывать некоторые особенности одинаково для всех броузеров, библиотека jQuery добавляет **уровень абстракции**, позволяю-

щий унифицировать решение типичных задач, уменьшить объем программного кода и существенно упростить его.

- **Всегда работает с наборами**. Когда мы предписываем библиотеке jQuery *отыскать все элементы с классом* `collapsable` *и скрыть их*, нам не требуется производить обход всех элементов в цикле, потому что методы, такие как `.hide()`, автоматически работают не с отдельными объектами, а с их наборами. Благодаря этому приему, называемому **неявной итерацией**, многие конструкции цикла становятся ненужными, что приводит к существенному сокращению объема программного кода.
- **Позволяет выполнять множество операций в одной строке**. Во избежание чрезмерного использования временных переменных или ненужных повторов библиотека jQuery для основных своих методов использует шаблон программирования, называемый **сцеплением**. Благодаря этому результатом большинства операций над объектом является сам объект, готовый к выполнению следующей операции.

Применение этих стратегий позволило сохранить малый объем библиотеки (сжатый файл занимает менее 20 Кбайт) и в то же время обеспечить компактность специализированного программного кода, использующего эту библиотеку.

Элегантность библиотеки обеспечивается частично архитектурными решениями, а частично эволюционным процессом ее развития, поддерживаемым энергичным сообществом, возникшим вокруг этого проекта. Пользователи jQuery обсуждают не только разработку модулей расширения, но и улучшения самого ядра библиотеки. В приложении А приводится множество ссылок на ресурсы сообщества, доступные разработчикам jQuery.

Несмотря на значительные усилия, необходимые для разработки столь гибкой и надежной системы, конечный продукт остается бесплатным для любых видов использования. Этот открытый проект распространяется под двойной лицензией: **GNU Public License** (как и многие другие открытые проекты) и **MIT License** (в целях содействия использованию jQuery в проприетарном программном обеспечении).

Хронология развития проекта jQuery

В данной книге рассматриваются функциональные возможности и синтаксис версии **jQuery 1.3.x**, последней на момент написания этих строк. Главная цель библиотеки – обеспечить простой способ поиска элементов веб-страницы и манипулирования ими – остается неизменной в ходе разработки, но некоторые особенности синтаксиса и возможности изменяются от версии к версии. В приведенном далее кратком обзоре хронологии развития проекта описываются наиболее важные изменения, происходившие от версии к версии.

- **Этап общественной разработки**: в августе 2005 года Джон Резиг (John Resig) впервые объявил об усовершенствованной библиотеке «Behaviour», основанной на библиотеке Prototype. Официально новая платформа была выпущена 14 января 2006 года под названием **jQuery**.
- **jQuery 1.0** (август 2006): эта первая стабильная версия библиотеки уже имела надежную поддержку селекторов CSS, технологии AJAX и механизма событий.
- **jQuery 1.1** (январь 2007): в этой версии была произведена существенная модернизация прикладного программного интерфейса (API). Многие редко используемые методы были объединены, что привело к уменьшению числа методов, которые необходимо изучать и документировать.
- **jQuery 1.1.3** (июль 2007): в этой версии была существенно повышена производительность механизма селекторов jQuery. Начиная с этой версии, производительность библиотеки jQuery стала сопоставимой с производительностью родственных библиотек JavaScript, таких как Prototype, Mootools и Dojo.
- **jQuery 1.2** (сентябрь 2007): из этой версии был убран синтаксис **XPath** выбора элементов, так как он стал избыточным при наличии синтаксиса CSS. В этой версии стала более гибкой настройка эффектов, а благодаря добавлению **логики управления событиями в пространствах имен** упростилась разработка расширений.
- **jQuery UI** (сентябрь 2007): этот новый набор расширений был выпущен, чтобы заменить популярный, но устаревающий модуль Interface. В него была включена богатая коллекция готовых виджетов, а также ряд дополнительных инструментов для создания сложных элементов, таких как интерфейсы буксирования элементов мышью (drag-and-drop).
- **jQuery 1.2.6** (май 2008): в состав основной библиотеки были включены функциональные возможности популярного модуля расширения **Dimensions**, созданного Брендоном Аароном (Brandon Aaron).
- **jQuery 1.3** (январь 2009): значительная модернизация механизма селекторов (**Sizzle**) обеспечила гигантский прирост производительности библиотеки. Официально было объявлено о поддержке **делегирования событий**.

Примечание

Примечания к выпуску для более старых версий jQuery можно найти на веб-сайте проекта по адресу: *http://docs.jquery.com/History_of_jQuery*.

Наша первая веб-страница, использующая библиотеку jQuery

Теперь, когда мы рассмотрели круг возможностей, доступных в библиотеке jQuery, можно попробовать задействовать ее.

Загрузка jQuery

На **официальном веб-сайте проекта jQuery** (*http://jquery.com/*) всегда можно найти самую последнюю версию библиотеки и самые свежие новости. Для начала нам необходимо получить копию библиотеки jQuery, которую можно загрузить непосредственно на главной странице сайта. Доступными могут быть несколько версий jQuery, но нам как разработчикам сайтов лучше всего подходит самая последняя несжатая версия. При вводе готового сайта в эксплуатацию ее можно заменить сжатой версией.

Никаких особенных действий при установке производить не требуется, достаточно просто поместить библиотеку в доступное место на сервере. Так как JavaScript является интерпретируемым языком программирования, можно не беспокоиться о сборке или компиляции. Чтобы обеспечить доступ к библиотеке внутри страницы, необходимо просто добавить ссылку на нее в документ HTML.

Подготовка документа HTML

Большинство примеров использования jQuery состоит из трех частей: собственно документ HTML, файлы CSS, где содержатся определения стилей, и файлы JavaScript, выполняющие операции над документом. В первом примере представлена страница с выдержкой из книги, к некоторым частям которой применяются классы CSS.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en" lang="en">
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8"/>

    <title>Through the Looking-Glass</title>

    <link rel="stylesheet" href="alice.css"
      type="text/css" media="screen" />

    <script src="jquery.js" type="text/javascript"></script>
    <script src="alice.js" type="text/javascript"></script>
  </head>
```

```
  <body>
    <h1>Through the Looking-Glass</h1>
    <div class="author">by Lewis Carroll</div>

    <div class="chapter" id="chapter-1">
      <h2 class="chapter-title">1. Looking-Glass House</h2>
      <p>There was a book lying near Alice on the table,
        and while she sat watching the White King (for she
        was still a little anxious about him, and had the
        ink all ready to throw over him, in case he fainted
        again), she turned over the leaves, to find some
        part that she could read, <span class="spoken">
        "&mdash;for it's all in some language I don't know,"
        </span> she said to herself.</p>
      <p>It was like this.</p>
      <div class="poem">
        <h3 class="poem-title">YKCOWREBBAJ</h3>
        <div class="poem-stanza">
          <div>sevot yhtils eht dna ,gillirb sawT'</div>
          <div>;ebaw eht ni elbmig dna eryg diD</div>
          <div>,sevogorob eht erew ysmim llA</div>
          <div>.ebargtuo shtar emom eht dnA</div>
        </div>
      </div>
      <p>She puzzled over this for some time, but at last
        a bright thought struck her. <span class="spoken">
        "Why, it's a Looking-glass book, of course! And if
        I hold it up to a glass, the words will all go the
        right way again."</span></p>
      <p>This was the poem that Alice read.</p>
      <div class="poem">
        <h3 class="poem-title">JABBERWOCKY</h3>
        <div class="poem-stanza">
          <div>'Twas brillig, and the slithy toves</div>
          <div>Did gyre and gimble in the wabe;</div>
          <div>All mimsy were the borogoves,</div>
          <div>And the mome raths outgrabe.</div>
        </div>
      </div>
    </div>
  </body>
</html>
```

Примечание

Фактическое местоположение файлов на сервере не имеет значения. Просто ссылки из одного файла на другой должны производиться в соответствии с выбранной схемой их размещения. В большинстве примеров в этой книге используются не абсолютные пути к файлам (`/images/foo.png`), а относительные (`../images/foo.png`). Это позволяет опробовать примеры на локальном компьютере без использования веб-сервера.

После обычной преамбулы HTML загружается таблица стилей. В данном примере используется следующая очень простая таблица стилей.

```
body {
  font: 62.5% Arial, Verdana, sans-serif;
}
h1 {
  font-size: 2.5em;
  margin-bottom: 0;
}
h2 {
  font-size: 1.3em;
  margin-bottom: .5em;
}
h3 {
  font-size: 1.1em;
  margin-bottom: 0;
}
.poem {
  margin: 0 2em;
}
.highlight {
  font-style: italic;
  border: 1px solid #888;
  padding: 0.5em;
  margin: 0.5em 0;
  background-color: #ffc;
}
```

Вслед за таблицей стилей подключаются файлы JavaScript. Очень важно, чтобы тег `script`, подключающий библиотеку jQuery, находился *перед* тегами, подключающими программный код наших сценариев, в противном случае платформа jQuery окажется недоступной, когда наш программный код попытается использовать ее.

Примечание

В оставшейся части книги будут приводиться только те фрагменты HTML и файлов CSS, которые важны для обсуждения. Полные версии файлов доступны на вспомогательном веб-сайте книги *http://book.learningjquery.com* и на веб-сайте издательства *http://www.packtpub.com/support*.

Теперь у нас имеется страница, которая выглядит так, как показано на рис. 1.1.

Примечание

В этом примере демонстрируется простейший случай использования jQuery. На практике такого рода оформление текста реализуется исключительно средствами CSS.

Through the Looking-Glass

by Lewis Carroll

1. Looking-Glass House

There was a book lying near Alice on the table, and while she sat watching the White King (for she was still a little anxious about him, and had the ink all ready to throw over him, in case he fainted again), she turned over the leaves, to find some part that she could read, "—for it's all in some language I don't know," she said to herself.

It was like this.

YKCOWREBBAJ
sevof yhlla eht dna ,gillrb sawT'
;ebaw eht ni olbmig dna eryg elD
,sevogorob eht erew ysmim lA
.eslgulp shlar emom eht dnA

She puzzled over this for some time, but at last a bright thought struck her. "Why, it's a Looking-glass book, of course! And if I hold it up to a glass, the words will all go the right way again."

This was the poem that Alice read.

JABBERWOCKY
'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

Рис. 1.1. *Первоначальный документ HTML*

Мы воспользуемся библиотекой jQuery, чтобы изменить оформление текста поэмы.

Подключение jQuery

Наш собственный программный код будет находиться во втором, пока еще пустом файле JavaScript, который подключается к документу HTML с помощью тега `<script src="alice.js" type="text/javascript"></script>`. Для данного примера нам потребуется написать всего три строки программного кода:

```
$(document).ready(function() {
  $('.poem-stanza').addClass('highlight');
});
```

Поиск текста поэмы

Фундаментальной операцией в библиотеке jQuery является выборка части документа. Ее выполняет конструкция `$()`. Обычно она принимает строковый параметр, который может содержать любое выражение селектора CSS. В данном случае нам требуется найти все части документа, к которым применяется класс `poem-staza`, поэтому селектор выглядит очень просто. Однако далее в книге мы будем рассматривать более сложные варианты. Подробнее о поиске различных частей документа будет рассказываться в главе 2.

В действительности функция `$()` является фабричной функцией, создающей **объект jQuery**, представляющий собой основной строительный блок, с которым нам предстоит работать. Объект jQuery объединяет ноль или более элементов DOM и позволяет воздействовать на них различными способами. В данном случае мы изменяем внешнее пред-

ставление выбранных частей страницы, причем выполняем это за счет изменения списка классов, применяемых к тексту поэмы.

Добавление нового класса

Метод `.addClass()` подобно большинству других методов jQuery имеет имя, говорящее само за себя, – он применяет класс CSS к выбранной части страницы. Метод принимает единственный параметр – имя добавляемого класса. Этот метод, как и его противоположность `.removeClass()`, позволяет исследовать работу библиотеки jQuery путем применения различных селекторов, доступных в библиотеке. Рассматриваемый пример просто добавляет класс `highlight`, который в нашей таблице стилей определен как курсивный текст с рамкой.

Обратите внимание, что для добавления класса нам не требуется выполнять обход всех строф поэмы в цикле. Как уже говорилось, в библиотеке jQuery используется прием **неявной итерации** в таких методах, как `.addClass()`, поэтому для изменения всех выбранных частей документа необходимо произвести единственный вызов функции.

Выполнение программного кода

Комбинации `$()` и `.addClass()` оказалось вполне достаточно, чтобы добиться поставленной цели, заключающейся в изменении внешнего вида текста поэмы. Однако если просто вставить эту строку программного кода в заголовок документа, она не даст никакого эффекта. Программный код JavaScript обычно запускается сразу же, как только обнаруживается броузером, то есть во время обработки заголовка, когда обрабатываемая этим кодом разметка HTML еще отсутствует. Нам необходимо отложить выполнение программного кода до момента, когда дерево DOM окажется доступным для использования.

Традиционным механизмом управления моментом запуска программного кода JavaScript являются **обработчики событий**. Многие обработчики служат для реализации реакции на события, инициируемые пользователем, такие как щелчок мышью или нажатие клавиши. Если бы в нашем распоряжении не было jQuery, нам пришлось бы использовать обработчик `onload`, который запускается после того, как отображена вся страница (вместе со всеми изображениями в ней). Чтобы запустить свой программный код по событию `onload`, необходимо было бы поместить его внутрь функции:

```
function highlightPoemStanzas() {
  $('.poem-stanza').addClass('highlight');
}
```

и затем подключить эту функцию как обработчик события, изменив тег `<body>`:

```
<body onload="highlightPoemStanzas();">
```

В этом случае программный код запустился бы по окончании загрузки страницы.

Такой подход имеет свои недостатки. Чтобы добиться требуемого эффекта, мы производим непосредственное изменение кода разметки HTML. Столь тесная связь между структурой и функциональностью приводит к загромождению кода, особенно при необходимости использовать одну и ту же функцию в нескольких страницах или в случае обработки других событий, таких щелчок мышью, в каждом экземпляре некоторого элемента страницы. Добавление новых эффектов может потребовать внесения изменений во многих местах, что увеличивает вероятность ошибки и осложняет параллельную работу дизайнеров и программистов.

Чтобы избежать подобных трудностей, библиотека jQuery предоставляет возможность запланировать однократный вызов функции после загрузки всего дерева DOM документа без ожидания загрузки изображений с помощью конструкции `$(document).ready()`. Для функции, определенной выше, эта конструкция приобретает вид:

```
$(document).ready(highlightPoemStanzas);
```

Данный прием не требует внесения изменений в разметку HTML; подключение функции производится непосредственно в файле JavaScript. Подробнее о том, как реализовывать обработку других **событий**, не влияя на структуру HTML, будет рассказываться в главе 3.

Однако описанная реализация все еще остается немного избыточной, так как функция `highlightPoemStanzas()` определяется лишь для однократного использования. Последнее означает, что для получения незначительной выгоды нам приходится использовать идентификатор в глобальном пространстве имен функций, который мы должны помнить, чтобы не использовать его повторно. В JavaScript, как и в некоторых других языках программирования, имеется возможность обойти такое неэффективное использование пространства имен, которая заключается в применении **анонимных функций** (иногда их называют **лямбда-функциями**). Благодаря анонимным функциям мы можем записать программный код в том виде, в каком он был представлен первоначально:

```
$(document).ready(function() {
  $('.poem-stanza').addClass('highlight');
});
```

С помощью ключевого слова `function`, за которым не следует имя функции, мы определяем функцию именно там, где она необходима, но не раньше. Тем самым мы ликвидируем лишний программный код JavaScript и сокращаем его общий объем до трех строк. Этот способ чрезвычайно удобен при работе с библиотекой jQuery, поскольку многие методы в качестве аргументов принимают функции, редко используемые более одного раза.

Если анонимные функции таким способом определяются в теле других функций, можно создать **замыкание**. Это продвинутая и мощная концепция, однако следует понимать, что широкое использование вложенных определений функций может приводить к непредсказуемым последствиям и чрезмерному расходованию памяти. Тема замыканий подробно рассматривается в Приложении C.

Конечный результат

Теперь, когда весь программный код JavaScript находится на своем месте, страница выглядит, как показано на рис. 1.2.

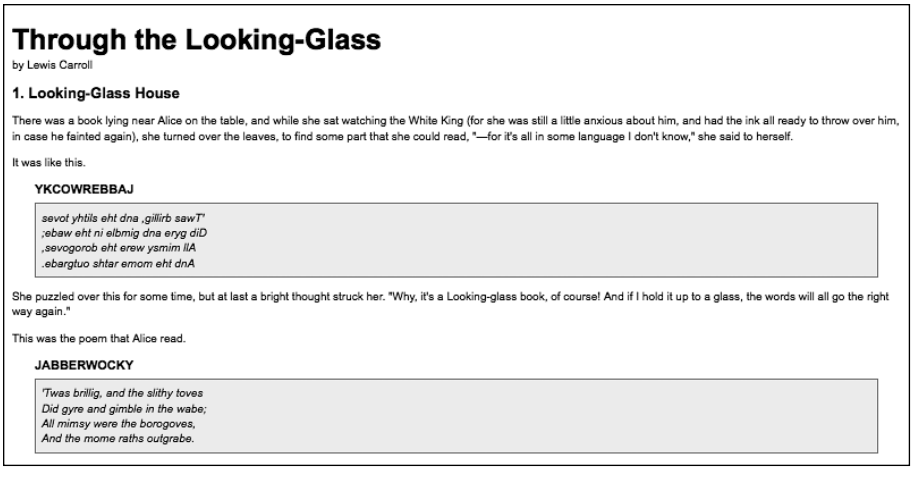


Рис. 1.2. *Страница после применения стилей*

Благодаря тому что программный код JavaScript добавляет класс `highlight`, строфы поэмы выводятся курсивом и заключены в рамки, как определено таблицей стилей `alice.css`.

В заключение

Итак, мы получили представление о том, почему даже для решения простых задач разработчики предпочитают использовать платформу JavaScript, вместо того чтобы писать весь программный код с самого начала. Кроме того, мы познакомились с некоторыми из преимуществ платформы jQuery, с учетом которых мы могли бы выбрать именно ее среди других вариантов. Мы обсудили также в общих чертах, решение каких задач упрощает jQuery.

В этой главе мы узнали, как обеспечить доступ к библиотеке jQuery из программного кода JavaScript в нашей веб-странице, как задействовать фабричную функцию `$()` для поиска элементов страницы, имеющих заданный класс CSS, как использовать метод `.addClass()` для применения

оформления к найденным элементам страницы и как вызывать метод `$(document).ready()` для запуска кода после загрузки страницы.

Для демонстрации работы библиотеки jQuery мы использовали простой, но мало полезный в реальных ситуациях пример. В следующей главе в ходе изучения сложного языка селекторов jQuery мы подробнее остановимся на программном коде и рассмотрим примеры его практического применения.

2

Селекторы

Для получения простого и быстрого доступа к элементам или группам элементов в объектной модели документа (Document Object Model, DOM) библиотека jQuery предоставляет мощные селекторы каскадных таблиц стилей (Cascading Style Sheets, CSS). В этой главе мы исследуем некоторые из этих селекторов, а также **дополнительные селекторы** библиотеки jQuery. Кроме того, мы рассмотрим **методы обхода дерева DOM**, обеспечивающие еще большую гибкость в достижении поставленных целей.

Объектная модель документа

Одной из наиболее сильных сторон jQuery является возможность легко производить выборку элементов дерева DOM. Объектная модель документа – своего рода древовидная структура. В языке HTML, как и в других языках разметки, эта модель применяется для описания взаимоотношений элементов страницы. Говоря о такого рода взаимоотношениях, мы будем использовать ту же терминологию, что и для описания взаимоотношений внутри семьи, – родители, потомки и так далее. Чтобы понять, как метафора генеалогического дерева применяется к документу, рассмотрим простой пример:

```
<html>
  <head>
    <title>the title</title>
  </head>
  <body>
    <div>
      <p>This is a paragraph.</p>
      <p>This is another paragraph.</p>
      <p>This is yet another paragraph.</p>
    </div>
  </body>
</html>
```

Здесь элемент <html> является **предком** для всех остальных элементов; другими словами, все остальные элементы являются **потомками** элемента <html>. Элементы <head> и <body> являются не только потомками, но еще и **дочерними** элементами по отношению к элементу <html>. Точно так же элемент <html> является не только предком для элементов <head> и <body>, но и их **родителем**. Элементы <p> являются дочерними (и потомками) по отношению к элементу , потомками по отношению к элементам <body> и <html> и **братскими** (то есть одного уровня) по отношению друг к другу. В Приложении B рассказывается, как можно визуализировать древовидную структуру DOM с использованием дополнительного программного обеспечения.

Прежде чем продолжать, следует заметить, что множества элементов, отбираемых методами с помощью селекторов, всегда обертываются в объекты jQuery. Эти объекты jQuery очень просты в обращении, когда требуется выполнить какие-то операции над элементами страницы. Мы легко можем привязывать к этим объектам обработчики **событий**, добавлять **эффекты**, а также объединять в **цепочки** несколько операций или эффектов. Однако объекты jQuery отличаются от обычных элементов DOM или узлов списков и не обязательно предоставляют те же методы и свойства. По этой причине в заключительной части этой главы мы рассмотрим способы доступа к элементам DOM, обернутым в объект jQuery.

Фабричная функция $()

Независимо от того, какого типа селектор будет использоваться, вызов любых операций из библиотеки jQuery всегда начинается со знака доллара и круглых скобок: $(). Почти все, что может использоваться в таблицах стилей, точно так же может обертываться в кавычки и помещаться между круглыми скобками, благодаря чему появляется возможность применять методы jQuery для поиска соответствующих элементов.

Примечание

Использование jQuery совместно с другими библиотеками JavaScript

Знак доллара $ в библиотеке jQuery представляет собой обычный «псевдоним» для идентификатора jQuery. При использовании в странице нескольких подобных библиотек может возникнуть конфликт имен, потому что функция $() очень часто встречается в библиотеках JavaScript. Этих конфликтов можно избежать, заменив в нашем программном коде, использующем библиотеку jQuery, каждый экземпляр $ на jQuery. Дополнительные решения этой проблемы рассматриваются в главе 10.

Селекторы составляются из трех основных строительных блоков: **имени тега, идентификатора (ID)** и **класса**. Они могут использоваться са-

мостоятельно или в комбинации с другими селекторами. В табл. 2.1 показано, как выглядит каждый из этих трех селекторов.

Таблица 2.1. Селекторы CSS и jQuery

Селектор	CSS	jQuery	Описание
Имя тега	`p`	`$('p')`	Выбирает все параграфы в документе
Идентификатор (ID)	`#some-id`	`$('#some-id')`	Выбирает единственный элемент документа с идентификатором `some-id`
Класс	`.some-class`	`$('.some-class')`	Выбирает все элементы документа, имеющие класс `some-class`

Как уже говорилось в главе 1, когда к фабричной функции `$()` присоединяются методы, они автоматически выполняют неявные итерации по элементам, заключенным в объект jQuery. Благодаря этому мы можем избежать **явного выполнения итераций**, например конструкции цикла `for`, которая часто бывает необходима при работе с деревом DOM.

Теперь, когда мы рассмотрели основы, можно приступать к исследованию более сложных случаев использования селекторов.

Селекторы CSS

Библиотека jQuery поддерживает практически все селекторы, включенные в спецификации CSS с 1 по 3, с которыми можно ознакомиться на сайте консорциума *World Wide Web Consortium*: *http://www.w3.org/Style/CSS/#specs*. Этот факт позволяет разработчикам развивать свои веб-сайты, не беспокоясь о том, что какие-то броузеры (в частности, Internet Explorer 6) могут не понимать некоторые селекторы при условии, что в броузеры включена поддержка JavaScript.

Примечание

Опытные разработчики, использующие jQuery, всегда должны применять в своем программном коде концепции **прогрессивного улучшения** (progressive enhancement) и **постепенной деградации** (graceful degradation), чтобы обеспечить точное отображение страницы, даже если с отключенной поддержкой JavaScript она будет выглядеть не так красиво, как с включенной. Мы будем рассматривать эти концепции на протяжении всей книги.

В ходе изучения принципов работы библиотеки jQuery с селекторами CSS мы будем использовать структуру, присутствующую на многих