HIGH TECH

БЕР БИБО
ИЕГУДА КАЦ

jQuery

Подробное руководство
по продвинутому
JavaScript

СИМВОЛ®

jQuery in Action

Bear Bibeault, Yehuda Katz

MANNING

HIGH TECH

jQuery

Подробное руководство по продвинутому JavaScript

Бер Бибо, Иегуда Кац

Символ®

Санкт-Петербург — Москва
2009

Серия «High tech»

Бер Бибо, Иегуда Кац

jQuery. Подробное руководство по продвинутому JavaScript

Перевод А. Киселева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*П. Щеголев*
Научный редактор	*Б. Попов*
Редактор	*Т. Темкина*
Корректор	*С. Минин*
Верстка	*Д. Орлова*

Бер Бибо, Иегуда Кац

jQuery. Подробное руководство по продвинутому JavaScript. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 384 с., ил.

ISBN-10: 5-93286-135-5

ISBN-13: 978-5-93286-135-6

Издание представляет собой введение в jQuery – мощную платформу для разработки веб-приложений. Ее уникальная способность составлять «цепочки» из команд позволяет выполнять несколько последовательных операций над элементами страницы. В книге подробно описано как выполнять обход документов HTML, обрабатывать события, воспроизводить анимацию и добавлять поддержку технологии Ajax в свои веб-страницы. Изучение каждой новой концепции закрепляется на практических примерах. Рассмотрены вопросы взаимодействия jQuery с другими инструментами и платформами и методы создания модулей расширения для этой библиотеки.

Книга предназначена для разработчиков, знакомых с языком JavaScript и технологией Ajax и стремящихся создавать краткий и понятный программный код – сократить его в несколько раз позволит грамотное использование библиотеки jQuery.

ISBN-10: 5-93286-135-5
ISBN-13: 978-5-93286-135-6
ISBN: 1-933988-35-5 (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.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 30.10.2008. Формат 70x100 1/16. Печать офсетная.
Объем 24 печ. л. Тираж 2000 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Предисловие

Эта книга о простоте. Зачем веб-разработчику писать сложный программный код объемом с хорошую книгу, если все, что требуется, – это реализовать достаточно простые взаимодействия? Нигде не сказано, что программный код веб-приложений непременно должен быть сложным.

Еще только обдумывая создание jQuery, я решил сосредоточиться на небольшом и простом программном коде, подходящем для всех практических приложений, с которыми веб-разработчики имеют дело каждый день. Прочитав книгу «jQuery in Action», я был приятно удивлен, поскольку увидел в ней яркое проявление принципов, заложенных в библиотеку jQuery.

Благодаря своей практической направленности, выразительности представленного программного кода и удачной структуре книга «jQuery in Action» послужит идеальным источником информации для тех, кто стремится познакомиться с этой библиотекой.

Больше всего в этой книге мне понравилось то внимание, которое проявили Бер (Bear) и Иегуда (Yehuda) к деталям внутренних механизмов библиотеки. Они очень тщательно исследовали и описали jQuery API. Ежедневно я бывал удостоен электронного письма или мгновенного сообщения с просьбой пояснить что-нибудь, с сообщением о новой ошибке, найденной в библиотеке, или с рекомендациями по ее улучшению. Вы можете быть уверены, что эта книга – это один из самых продуманных и проработанных источников знаний о библиотеке jQuery.

Меня приятно удивило то обстоятельство, что в книге описаны подключаемые модули (plugins), а также тактика и теория, лежащие в основе разработки модулей jQuery. Причина неизменной простоты jQuery заключается в ее модульной архитектуре. Библиотека предоставляет множество документированных точек расширения, позволяющих наращивать ее функциональность путем подключения к ним модулей. Добавляемые функциональные возможности, хотя и полезны, но зачастую недостаточно универсальны, чтобы включить их в саму библиотеку jQuery, поэтому и важна модульная архитектура. Некоторые из подключаемых модулей, например Forms, Dimension и LiveQuery, получили очень широкое распространение по вполне очевидным причинам: они грамотно написаны, хорошо документированы и обеспечены технической поддержкой. Обязательно обратите особое внимание

на то, как создаются и как используются модули, поскольку на них базируется jQuery.

С такими источниками информации, как эта книга, проект jQuery будет и дальше успешно развиваться. Раз уж вы решились изучать и применять jQuery, надеюсь, эта книга будет полезна и вам.

Джон Ресиг (John Resig),
создатель библиотеки jQuery

Введение

Один из авторов книги – седой ветеран, начавший программировать, когда FORTRAN стал бомбой, а второй – не по годам сообразительный специалист в данной области, не заставший то время, когда не было Интернета. Что объединило этих двух человек, с таким разным жизненным опытом, для совместной работы над книгой?

Ответ очевиден – библиотека *jQuery*.

Мы пришли к этому, одному из наиболее интересных инструментов создания клиентских приложений путями разными, как день и ночь.

Я (Бер), впервые услышал о jQuery, когда работал над книгой «Ajax in Practice»[1]. Заключительный, и самый лихорадочный, этап работы над книгой называется *редактированием* – кроме всего прочего, технический редактор проверяет грамматическую правильность текста и ясность изложения информации. Это самый напряженный, по крайней мере для меня, период, когда меньше всего мне хотелось бы услышать: «Определенно, здесь нужен еще один раздел».

Одна из глав, написанных мной для книги «Ajax in Practice», рассматривает несколько библиотек поддержки технологии Ajax на стороне клиента, с одной из которых я уже был очень близко знаком (Prototype), а с другими (Dojo Toolkit и DWR) мне пришлось познакомиться очень быстро.

Жонглируя множеством задач (все время оставаясь в курсе дел, руководя своим бизнесом и решая семейные проблемы), технический редактор Валентин Греттаз (Valentin Crettaz) случайно обронил: «А почему у вас нет раздела о jQuery?»

«О джей чём?», – спросил я.

И тут же прослушал лекцию о том, как прекрасна эта совершенно новая библиотека и что ее непременно следует включать в любое современное исследование библиотек, обеспечивающих поддержку технологии Ajax на стороне клиента. Я поспрашивал вокруг – не слыхал ли кто о библиотеке jQuery?

---

[1] Дейв Крейн, Бер Бибо, Джордон Сонневельд «Ajax на практике». – Пер. с англ. – К.: Вильямс, 2008.

Многие ответили положительно, с восторгом подтвердив, что jQuery действительно превосходная библиотека. Дождливым воскресным днем я провел четыре часа на сайте jQuery, изучая документацию и пробуя писать маленькие тестовые программы. Затем я написал новый раздел и отправил его техническому редактору, чтобы убедиться в том, что правильно понял суть библиотеки.

Раздел был принят на ура, и заключительный этап работы над книгой «Ajax in Practice» продолжился. (Добавлю, что раздел о jQuery был опубликован в электронном журнале «Dr. Dobb's Journal».)

Когда пыль улеглась, на задворках сознания пустила ростки моя суматошная попытка понять суть jQuery. Мне понравилось то, что я увидел во время своего стремительного погружения, и захотелось познакомиться с библиотекой поближе. Я стал использовать jQuery в веб-проектах. То, что я увидел, понравилось мне еще больше. Я пробовал заменять старый программный код предыдущих проектов, чтобы увидеть, насколько проще становятся веб-страницы благодаря jQuery. И мне *действительно* понравился полученный результат.

Восхищенный своим новым открытием и гонимый желанием поделиться им с другими, я послал в издательство Manning предложение написать книгу «jQuery in Action». Разумеется, я должен был их убедить. (Из чувства мести за такой переполох, я предложил своему техническому редактору, с которого все началось, стать техническим редактором и *этой* книги. Клянусь, *это* послужило ему хорошим уроком!)

И тут редактор Майк Стефенс (Mike Stephens) спросил: «А не хотели бы вы работать над книгой вместе с Иегудой Кацем (Yehuda Katz)?»

«С Иегентой кем?», – спросил я...

Иегуда пришел в этот проект совершенно другим путем; он был знаком с библиотекой jQuery еще в те дни, когда она даже не имела номера версии.[1] Наткнувшись на модуль Selectables, он заинтересовался основной библиотекой jQuery. Несколько разочарованный недостатком (в то время) электронной документации, он обыскал различные wiki-ресурсы[2] и основал сайт Visual jQuery (*visualjquery.com*).

---

[1] В оригинале – «имела номер версии», без частицы «не», однако после знакомства с разделом history на сайте jquery.com, я убедился, что библиотека получила первый номер версии почти год спустя после начала разработки. – *Примеч. пер.*

[2] Те, кого интересует происхождение и история wiki-проектов, могут заглянуть, например, сюда: *wiki.org/wiki.cgi?WhatIsWiki* – что такое Wiki, *c2.com/cgi/wiki?WikiWikiWeb* – распространенный термин, *c2.com/cgi/wiki?InformalHistoryOfProgrammingIdeas* – «ты помнишь, как все начиналось...» – *Примеч. науч. ред.*

В скором времени он дал толчок появлению отличной документации, стал оказывать помощь проекту jQuery и следить за модульной архитектурой и экосистемой, попутно пропагандируя jQuery в сообществе пользователей Ruby.

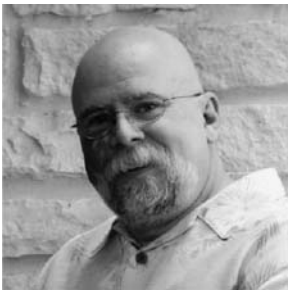
И вот настал день, когда ему позвонили из издательства Manning (издателю его порекомендовал друг), предложив поработать над книгой о jQuery вместе с парнем по имени Бер...

Несмотря на разницу в возрасте, опыте и путях, которыми мы пришли в этот проект, из нас получилась отличная команда и мы прекрасно провели время, работая над книгой. Даже географическая разобщенность (я живу в сердце Техаса, а Иегуда – на побережье Калифорнии) не стала для нас препятствием благодаря электронной почте и системе обмена мгновенными сообщениями!

Думаем, что комбинация наших знаний и талантов позволила сделать добротную и информативную книгу для вас. Надеемся, что, читая ее, вы получите столько же удовольствия, сколько и мы, работая над ней.

Рекомендуем только выбрать наиболее подходящее для чтения время.

Об авторах



Бер Бибо разрабатывает программное обеспечение уже тридцать лет, начав с создания программы Tic-Tac-Toe на суперкомпьютере Control Data Cyber, где в качестве устройства ввода применялся телетайп со скоростью обмена 100 бод. Поскольку у Бера два электротехнических диплома, он должен был бы проектировать антенны или что-то вроде того, но начав работать в Digital Equipment Corporation, он все больше тянулся к программированию. Бер успел поработать на Lightbridge Inc., BMC Software, Dragon Systems и даже служил в вооруженных силах США, где обучал солдат-пехотинцев взрывать танки. (Угадайте, чем он больше всего увлекался!) Теперь Бер – разработчик архитектур программных комплексов и технический менеджер в компании, которая занимается разработкой и сопровождением крупных финансовых веб-приложений, обслуживающих бухгалтерию многих компаний из списка Fortune 500.

Кроме повседневной работы, Бер пишет книги, руководит небольшой собственной фирмой, которая создает веб-приложения и предлагает другие электронные услуги (только не видеосъемку свадеб, никакой свадебной видеосъемки!), а также помогает поддерживать порядок на сайте *JavaRanch.com* под псевдонимом sheriff. Когда не сидит за компьютером, Бер любит приготовить *много* еды (чем и объясняется размер его джинсов), увлекается монтажом фото- и видеофильмов, катается на своем мотоцикле Yamaha V-Star и носит футболки с тропическими картинками.

Живет и работает он в городе Остин (Austin), штат Техас, который нежно любит, за исключением аболютно ненормальных водителей.

Иегуда Кац за последние несколько лет поучаствовал в разработке нескольких проектов с открытым исходным кодом. Он не только член основной команды проекта jQuery, но также участник проекта Merb, альтернативы Ruby on Rails (также реализованной на языке Ruby).

Иегуда родился в штате Миннесота, рос в Нью Йорке, а теперь живет в солнечной Калифорнии, в городе Санта-Барбаре. Он занимался разработкой веб-сайтов для New York Times, Allure Magazine, Architectural Digest, Yoga Journal и других известных клиентов. Программирует на таких языках, как Java, Ruby, PHP и JavaScript.

В свободное от работы время помогает поддерживать *VisualjQuery.com* и отвечает на вопросы начинающих пользователей jQuery на канале IRC и в официальной почтовой рассылке jQuery.

Благодарности

Вас не удивляет длинный список имен, пробегающих по экрану в конце фильма? Неужели для создания фильма действительно требуется столько людей?

В работе над книгой тоже участвует столько людей, что большинство из вас удивится. Многие должны вложить в книгу свой труд и талант, чтобы у вас в руках оказался этот томик (или электронная книга, которую вы читаете с экрана).

Специалисты издательства Manning усердно трудились вместе с нами, чтобы книга получилась хорошей, и мы благодарны им за это. Без них эта книга не состоялась бы. В «заключительные титры» этой книги попали имена не только нашего издателя Марьян Бас (Marjan Bace) и редактора Майка Стефенса, но и следующих сотрудников: Дуглас Падник (Douglas Pudnick), Андреа Кучер (Andrea Kaucher), Карен Тегтмайер (Karen Tegtmayer), Кэти Тенант (Katie Tenant), Меган Йоки (Megan Yockey), Дотти Марсико (Dottie Marsico), Мэри Пиргис (Mary Piergies), Тиффани Тейлор (Tiffany Taylor), Денис Далинник (Denis Dalinnik), Габриель Добреску (Gabriel Dobrescu) и Рон Томич (Ron Tomich).

Трудно переоценить труд наших рецензентов, которые помогали довести книгу до совершенства, отыскивали опечатки, исправляли ошибки в терминологии и в программном коде и даже помогли организовать структуру глав в книге. Каждый цикл рецензирования существенно улучшал книгу. За время, потраченное на рецензирование книги, мы хотим поблагодарить Джонатана Блумера (Jonathan Bloomer), Валентина Греттаза, Дениса Куриленко (Denis Kurilenko), Раму Кришна Вавилала (Rama Krishna Vavilala), Филипа Халлсторма (Philip Hallstrom), Джея Бланчарда (Jay Blanchard), Джеффа Каннингхема (Jeff Cunningham), Эрика Паскарело (Eric Pascarello), Джоша Хейера (Josh Heyer), Грегга Болингера (Gregg Bolinger), Эндрю Симера (Andrew Siemer), Джона Тайлера (John Tyler) и Теда Годдарда (Ted Goddard).

Отдельное спасибо Валентину Греттазу, техническому редактору книги. Помимо проверки всех примеров программного кода в разных окружениях он предложил массу ценных рекомендаций по повышению технической точности текста.

Бер Бибо (Bear Bibeault)

Это моя третья публикация, и список тех, кого я хотел бы поблагодарить, включая всех членов и персонал *javaranch.com*, довольно велик. Без моей причастности к проекту JavaRanch я никогда не смог бы начать писать и поэтому я искренне благодарен Полу Уитону (Pol Wheaton) и Кэти Сиерра (Kathy Sierra), с кого все это началось, а также другим штатным сотрудникам, которые подбадривали и поддерживали меня, включая (как минимум) Эрика Паскарелло (Eric Pascarello), Бена Соузера (Ben Souther), Эрнеста Фридмана Хилла (Ernest Friedman Hill), Марка Хершберга (Mark Herschberg) и Макса Хаббиби (Max Habbibi).

Выражаю свою благодарность Валентину Греттазу – он не только был техническим редактором, но и первым ввел меня в мир jQuery, а также моему коллеге Дэниелу Хедрику (Daniel Hedrick), который добровольно предлагал мне примеры программного кода на языке PHP для последней части книги.

Мой партнер Джей (Jay) и собаки – Литл Бер (не могли же мы назвать его просто *Бером*?) и Козмо, – спасибо вам за ваше незаметное присутствие и за то, что делили со мной крышу над головой, почти не задевая клавиатуру MacBook Pro в течение всех месяцев, пока я работал над этой книгой.

Наконец, я хочу поблагодарить моего соавтора Иегуду Каца, без которого этот проект не смог бы воплотиться.

Иегуда Кац (Yehuda Katz)

Для начала я хотел бы выразить свою благодарность моему соавтору Беру Бибо за его огромный опыт в работе над книгами. Рождению этой книги в огромной мере способствовали его писательский талант и удивительная способность преодолевать препятствия, возникающие на пути профессиональной публикации.

Заговорив о тех, благодаря кому все стало возможным, считаю своим долгом поблагодарить мою любимую жену Леа (Leah), которая проводила без меня столько долгих ночей и рабочих выходных, что заранее я не решился бы попросить об этом. Ее вклад в эту книгу сравним с моим собственным; как всегда, благодаря ей я смог перенести самый сложный этап проекта. Я люблю тебя, Леа.

Совершенно очевидно, что без библиотеки jQuery не было бы и книги «jQuery in Action». Я хочу выразить свою благодарность создателю jQuery Джону Ресигу за то, что он изменил процесс разработки клиентских сценариев, сняв гору с плеч веб-разработчиков всей планеты (видите ли, у нас есть довольно большие группы разработчиков в Китае, Японии, Франции и многих других странах). Я также считаю его своим другом и талантливым автором, помогавшим мне подготовиться к выполнению моих тяжелых обязательств.

Никакой библиотеки jQuery не было бы без огромного сообщества пользователей и членов основной команды, включая разработчиков Брендона Аарона (Brandon Aaron) и Йерна Зафферера (Jörn Zaefferer), популяризаторов Рея Банго (Rey Bango) и Карла Шведберга (Karl Swedberg), Пауля Бакауса (Paul Bakaus), возглавляющего проект jQuery UI, а также Клауса Хартла (Klaus Hartl) и Майка Алсупа (Mike Alsup), работающих в команде разработки модулей вместе со мной. Эта большая группа программистов помогала продвигать платформу jQuery от простого набора базовых операций до библиотеки JavaScript мирового уровня с поддержкой (модульной) практически всего, что только можно пожелать, полностью реализованной усилиями пользователей. Я не привожу полный список тех, кто участвовал в работе, – вас так много! Достаточно сказать, что меня не было бы здесь без уникального сообщества, сплотившегося вокруг этой библиотеки, и у меня не хватает слов, чтобы выразить свою благодарность.

Наконец, я хотел бы поблагодарить свою семью, которую я почти не вижу с тех пор, как начались мои поездки по стране. По мере взросления я и мои братья впитали дух товарищества, а вера семьи в меня всегда придавала мне сил и уверенности. Мама, Никки (Nikki), Эби (Abie) и Иаков (Yaakov): спасибо вам, я люблю вас.

Об этой книге

Делай больше меньшими усилиями.

Эти слова просто и понятно выражают цель этой книги: помочь вам узнать, как наполнить веб-страницы большей функциональностью при меньшем объеме сценариев. Авторы книги, один из которых сотрудник и проповедник (evangelist[1]) jQuery, а другой – увлеченно-восторженный пользователь, полагают, что jQuery – лучшая библиотека из доступных на сегодняшний день, которая позволит вам сделать это.

Эта книга поможет вам быстро и эффективно овладеть jQuery и, надеемся, сделает это занятие увлекательным. Здесь рассматривается собственно ядро jQuery API, каждый метод прикладного программного интерфейса описан в удобном для восприятия формате, включающем описание входных параметров и возвращаемых значений. В книге присутствуют небольшие примеры использования библиотеки, а для так называемых *важных концепций* мы предусмотрели так называемые *лабораторные страницы* (*lab pages*). Эти забавные проверочные страницы представляют собой прекрасный способ увидеть нюансы использования методов jQuery без необходимости писать много своего программного кода.

Все исходные тексты примеров и лабораторных страниц можно загрузить с сайта *www.manning.com/bibeault*.

Мы могли бы еще долго рассказывать, как хороша эта книга, подпуская при этом рекламные словечки, но вряд ли кому приятно тратить время на чтение этой чепухи, правда? Чего вам на самом деле хочется – погрузиться в биты и байты, ведь так?

Так что же вас держит? Вперед!

Для кого эта книга

Книга адресована как начинающим, так и опытным веб-разработчикам, желающим взять на себя управление сценариями JavaScript на

---

[1] Наличие таких людей важно для успешного проекта. К сожалению, не всегда «верхний менеджмент» организации понимает это. Но, например, в компании Sun Microsystems специально введен почетный и поощряемый статус Евангелиста – проповедника и проводника философии Java. – *Примеч. науч. ред.*

своих веб-страницах и создавать настоящие, интерактивные, богатые возможностями интернет-приложения без необходимости писать весь клиентский программный код, что обычно требуется при создании приложений на пустом месте.

Пользу из этой книги извлекут все веб-разработчики, стремящиеся создавать с помощью библиотеки jQuery удобные веб-приложения, которые бы восхищали, а не раздражали пользователей.

При чтении некоторых разделов начинающий веб-разработчик может почувствовать себя несмышленышем, но это не должно его останавливать. Мы включили в книгу приложение с описанием основных понятий JavaScript, помогающих освоить весь потенциал jQuery. Поняв основные концепции, такой читатель обнаружит, что сама библиотека jQuery очень дружелюбна к новичкам и при этом достаточно мощна для более опытных веб-разработчиков.

Как новичкам, так и ветеранам разработки веб-приложений полезно включить jQuery в свой арсенал. Мы ручаемся, что эта книга поможет вам быстро изучить все, что для этого нужно.

Структура книги

Книга организована так, чтобы вы смогли овладеть jQuery максимально быстро и эффективно. Она начинается с введения в основы jQuery и сразу переходит к фундаментальным концепциям прикладного программного интерфейса jQuery. После этого мы продемонстрируем вам различные области применения, где jQuery позволяет писать невероятно продуктивный клиентский программный код, от обработки событий до выполнения запросов к серверу с применением технологии Ajax. Затем вашему вниманию будет представлен обзор наиболее популярных расширений jQuery.

В главе 1 мы рассмотрим философию jQuery и то, как она согласуется с такими современными принципами программирования, как «ненавязчивый JavaScript». Мы исследуем причины, по которым было бы желательно использовать jQuery, вкратце рассмотрим принцип ее действия и такие базовые концепции, как обработчики события готовности документа, вспомогательные функции, создание элементов объектной модели документа (Document Object Model, DOM) и расширений для jQuery.

В главе 2 представлена концепция обернутых наборов элементов – базовая концепция jQuery, определяющая принцип ее действия. Вы узнаете, как создавать обернутый набор элементов – коллекцию элементов DOM, участвующих в операции как единое целое, – за счет выбора элементов документа страницы с помощью богатой коллекции мощных *селекторов* jQuery. Вы увидите, как эти селекторы наращивают возможности, предлагающие нам стандартные способы применения каскадных таблиц стилей (CSS).

В главе 3 мы узнаем, как можно использовать обернутые наборы jQuery для управления структурой DOM страницы. Мы рассмотрим вопросы изменения стилей и атрибутов элементов, содержимого элементов, а также перемещения элементов в пределах страницы и изучим принципы работы с элементами формы.

В главе 4 показано, как с помощью jQuery существенно упростить обработку событий в веб-страницах. В конце концов, именно обработка пользовательских событий делает интернет-приложения полнофункциональными, и все, кому уже приходилось иметь дело с запутанными механизмами обработки событий в разных броузерах, порадуются простоте данной задачи с jQuery.

Мир анимации и визуальных эффектов станет предметом обсуждения главы 5. Здесь мы увидим, что jQuery позволяет создавать анимационные эффекты не только просто, но и эффективно, и даже забавно.

В главе 6 мы познакомимся со вспомогательными функциями и флагами, которые jQuery предоставляет в распоряжение не только авторов страниц, но и всех, кто пишет расширения и модули для jQuery.

О том, как пишутся расширения и модули, будет рассказано в главе 7. Мы увидим, насколько просто пишутся расширения для jQuery, – для этого не потребуется писать хитроумный программный код JavaScript или знать устройство jQuery. Поэтому есть смысл всякий программный код многократного использования оформлять в виде расширения для jQuery.

Глава 8 заинтересует вас одной из наиболее важных областей в разработке полнофункциональных интернет-приложений: выполнение запросов Ajax. Здесь мы увидим, насколько проще применять технологию Ajax с помощью jQuery, и как эта библиотека ограждает нас от ловушек, которыми чревато введение поддержки технологии Ajax в страницы, существенно упрощая реализацию наиболее распространенных видов взаимодействий Ajax (таких, как возврат данных в формате JSON[1]).

Наконец, в главе 9 мы познакомимся с наиболее популярными и мощными модулями jQuery и узнаем, где отыскать информацию о множестве подобных модулей расширения. Мы исследуем модули, позволяющие обрабатывать формы, и более мощные возможности Ajax, чем предполагает jQuery, а также модули, позволяющие реализовать механизм перетаскивания (drag-and-drop) на страницах.

В приложении описаны такие ключевые концепции JavaScript, как *контексты функции* и *замыкания* (*closers*), составляющие основу для максимально эффективного использования библиотеки jQuery в наших

---

[1] JSON (JavaScript Object Notation) – формат обмена данными; основан на подмножестве языка программирования JavaScript, определенного в стандарте ECMA-262 3rd Edition, December 1999 (*json.org/json-ru.html*). – *Примеч. науч. ред.*

страницах. Это приложение предназначено для тех, кто хочет освежить свои знания об этих концепциях.

Соглашения по оформлению примеров программного кода

Программный код в листингах примеров или в тексте выполнен моноширинным шрифтом, как, например, `этот текст`, чтобы отделить его от обычного текста. Имена методов и функций, свойств, элементов XML и атрибутов также выделяются этим шрифтом.

В некоторых случаях первоначальный программный код может быть отформатирован с целью уместить его на книжной странице. Вообще, программный код изначально был написан с учетом ограниченной ширины книжной страницы, но иногда вы можете заметить незначительные различия между оформлением листингов и исходных программных кодов примеров, доступных для загрузки. В нескольких редких случаях, когда длинная строка не может быть отформатирована без изменения ее смысла, в листинге появится маркер продолжения строки. Большая часть листингов сопровождается описанием, в котором подчеркиваются наиболее важные концепции. Часто в листингах присутствуют нумерованные маркеры для ссылок из пояснительного текста.

Загрузка программного кода примеров

Исходный программный код всех приведенных в книге примеров (а также ряд дополнительных примеров, не попавших в книгу) доступен для загрузки на странице *http://www.manning.com/jQueryinAction* или *http://www.manning.com/bibeault*.

Примеры программного кода организованы так, чтобы его проще было использовать с локальным веб-сервером. Распакуйте загруженный архив с примерами в любую папку и назначьте ее корневой папкой документов веб-сервера. Начальная страница, откуда можно запускать примеры, находится в корневой папке, в файле *index.html*.

За исключением примеров для главы 8 и части примеров для главы 9, примерам не требуется локальный веб-сервер, и можно загружать их непосредственно в броузер. Инструкции по установке программного продукта Tomcat – веб-сервера для примеров из главы 8 – приведены в файле *chapter8/tomcat.pdf*.

Все примеры были опробованы в разных типах броузеров, включая Internet Explorer 7, Firefox 2, Opera 9, Safari 2 и Camino 1.5. Примеры также вполне успешно работают в Internet Explorer 6, но при этом могут наблюдаться некоторые проблемы с расположением элементов на странице. Обратите внимание: программный код библиотеки jQuery работает в IE6 безупречно – все проблемы с расположением элементов обусловлены применением стилей CSS. Поскольку большинство читателей этой книги – профессиональные веб-разработчики, предполага-

ется, что у каждого читателя есть несколько разных броузеров, в которых будут опробоваться примеры.[1]

Форум Author Online

Покупка книги «jQuery в действии» дает право свободного доступа к частному форуму, который поддерживается издательством Manning Publications, где можно оставлять свои комментарии о книге, задавать вопросы технического характера и получать помощь от авторов книги и от других пользователей. Чтобы получить доступ к форуму, откройте в броузере страницу *http://www.manning.com/jQueryinAction* или *http://www.manning.com/bibeault*. Здесь вы найдете информацию о том, как попасть на форум после регистрации, какого рода помощь будет доступна, и о правилах поведения в форуме.

Издательство Manning обязуется предоставить своим читателям место встречи, где может завязаться диалог между читателями и авторами книги. Но издательство не гарантирует присутствие всех или части авторов на форуме – их содействие форуму книги остается добровольным (и неоплачиваемым). Рекомендуем задавать такие вопросы, которые могли бы вызвать интерес у авторов!

Форум Author Online и архивы предыдущих обсуждений доступны на веб-сайте издателя.

Об этой серии

Введения, обзоры и примеры в книгах серии «...in Action» комбинируются так, чтобы способствовать изучению *и* запоминанию. Согласно исследованиям в когнитивистике лучше запоминается информация, полученная при наличии заинтересованности в ее получении.

В издательстве Manning нет ни одного специалиста в области когнитивистики, тем не менее, мы убеждены, что для лучшего запоминания процесс обучения должен пройти через стадии исследования, игры и, что довольно интересно, пересказа изученного материала. Люди склонны запоминать новые сведения, которые дает им преподаватель, только активно исследовав их. Мы учимся *в действии*. Квинтэссенцией книг «...in Action» являются примеры. Они побуждают читателя опробовать новый программный код, поиграть с ним и исследовать новые идеи.

Есть и другая, более прозаическая причина такого названия книг: наши читатели – занятые люди. Они читают книги, чтобы выполнить

---

[1] Авторы подготовили примеры с помощью jQuery 1.2.1. Ко времени подготовки данного перевода для загрузки появилась версия 1.2.6 (*http://docs.jquery.com/Downloading_jQuery*). Отличия версий описаны в разделах Release Notes сайта проекта: *http://docs.jquery.com/Release:jQuery 1.2.6*. – *Примеч. науч. ред.*

свою работу или решить проблему. Им требуются книги, позволяющие легко находить информацию и изучать только то, что нужно, и только когда это нужно. Им нужны книги, которые помогут им *в действии*. Книги этой серии предназначены именно для таких читателей.

Об иллюстрации на обложке

Иллюстрация на обложке книги «jQuery in Action» называется «The Watchman» (Дозорный). Она взята из французской книги «Encyclopedie des Voyages» Дж. Г. С. Саво (J. G. St. Saveur), опубликованной в 1796 году. Путешествия для удовольствия были в то время относительно новым явлением, и большой популярностью пользовались разные путеводители, такие как эта книга. Обычного путешественника или того, кто любит путешествовать сидя дома, эта книга знакомила с жителями других стран мира, а также с национальными костюмами и униформой французских солдат, государственных служащих, торговцев и крестьян.

Разнообразные рисунки в «Encyclopedie des Voyages» – яркое свидетельство того, как уникальны и неповторимы были города мира всего 200 лет тому назад. Это было время, когда особенности одежды жителей двух областей, разделенных всего несколькими десятками миль, позволяли безошибочно определить, в какой из двух областей проживает человек. Путеводитель позволяет почувствовать разобщенность людей того времени и любого другого исторического периода, за исключением нынешнего.

С тех пор стиль одежды сильно изменился, и разнообразие, обусловленное разным происхождением, исчезло. Теперь по одежде сложно определить, на каком континенте проживает тот ли иной человек. Пожалуй, с оптимистичной точки зрения, мы пожертвовали культурным и внешним разнообразием в угоду разнообразию личной жизни. Или для более разнообразной и интересной интеллектуальной и технической деятельности.

Мы, издательство Manning, прославляем изобретательность, инициативу и радость работы с компьютерами, воскрешая на обложках наших книг иллюстрации из путеводителя двухвековой давности, отражающие богатое многообразие тогдашней жизни.

В этой главе:

- Для чего нужна библиотека jQuery
- Что такое «ненавязчивый JavaScript»
- Основы jQuery
- jQuery и другие библиотеки JavaScript

1

Введение в jQuery

Большую часть своей жизни считавшийся среди серьезных веб-разработчиков «игрушечным», язык JavaScript обрел былой авторитет на волне интереса к полнофункциональным интернет-приложениям и технологиям Ajax. Языку пришлось очень быстро повзрослеть, так как разработчики клиентских сценариев отказались от приема копирования и вставки программного кода JavaScript в пользу полноценных библиотек, которые позволяют решать сложные проблемы, связанные с различиями между броузерами разных типов, и реализуют новые и улучшенные парадигмы разработки веб-приложений.

Несколько запоздало появившаяся в мире библиотек JavaScript jQuery покорила сообщество веб-разработчиков, быстро завоевав поддержку крупных сайтов, таких как MSNBC, и была высоко оценена такими открытыми проектами, как SourceForge, Trac и Drupal.

В отличие от других инструментов, сконцентрированных на применении сложных методик JavaScript, jQuery стремится изменить представление веб-разработчиков о принципах создания полнофункциональных интернет-приложений. Вместо того чтобы тратить время на жонглирование непростыми возможностями языка JavaScript, разработчики могли бы, применив знание каскадных таблиц стилей (Cascading Style Sheets, CSS), расширенного языка разметки гипертекста (eXtensible Hypertext Markup Language, XHTML) и старого доброго JavaScript, напрямую манипулировать элементами страницы, воплотив мечту о быстрой разработке веб-приложений.

В этой книге мы во всех подробностях рассмотрим, что может предложить jQuery нам как авторам полнофункциональных интернет-приложений для страниц. Для начала узнаем, что в действительности может дать jQuery при разработке веб-страниц.

1.1. Почему jQuery?

Потратив некоторое время на попытки привнести динамическую функциональность в ваши страницы, вы обнаружите, что постоянно следуете одному и тому же шаблону: сначала отбирается элемент или группа элементов, а затем над ними выполняются некоторые действия: вы могли бы скрывать или показывать элементы, добавлять к ним класс CSS, создавать анимационные эффекты или изменять атрибуты.

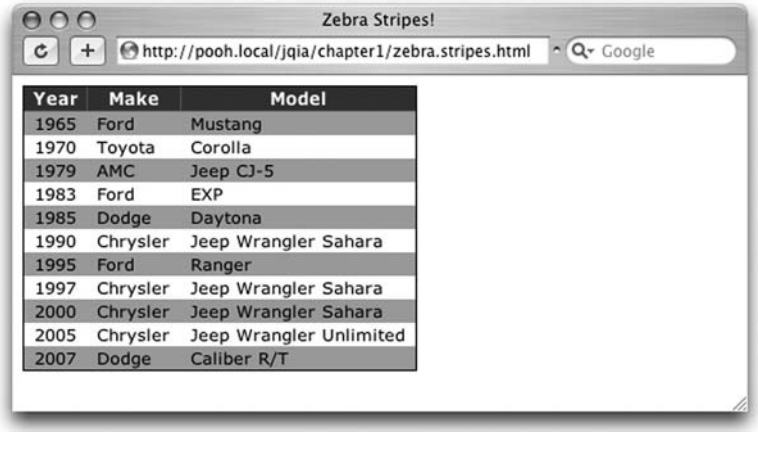
С обычным JavaScript для решения каждой из задач потребуются десятки строк программного кода. Создатель jQuery разработал свою библиотеку именно для того, чтобы сделать наиболее общие задачи тривиальными. Например, чтобы создать таблицу с разным цветом фона для четных и нечетных строк, дизайнеру потребуется написать до 10 строк кода на языке JavaScript. А вот как тот же эффект достигается с помощью jQuery:

```
$("table tr:nth-child(even)").addClass("striped");
```

Не волнуйтесь, если сейчас эта строка кажется вам странной. Вскоре вы поймете, как она работает, и сами будете использовать короткие, но мощные инструкции jQuery, чтобы добавить живости своим страницам. Давайте коротко рассмотрим, как работает этот фрагмент кода.

Эта инструкция отыскивает все четные строки (элемент `<tr>`) во всех таблицах на странице и добавляет к ним класс CSS `striped`. Применяя желаемый цвет фона к этим строкам через правило CSS класса `striped`, единственная строка JavaScript может улучшить эстетику всей страницы.

В результате таблицы на странице будут выглядеть, как показано на рис. 1.1.



Year	Make	Model
1965	Ford	Mustang
1970	Toyota	Corolla
1979	AMC	Jeep CJ-5
1983	Ford	EXP
1985	Dodge	Daytona
1990	Chrysler	Jeep Wrangler Sahara
1995	Ford	Ranger
1997	Chrysler	Jeep Wrangler Sahara
2000	Chrysler	Jeep Wrangler Sahara
2005	Chrysler	Jeep Wrangler Unlimited
2007	Dodge	Caliber R/T

Рис. 1.1. *Библиотека jQuery позволяет добавить расцветку строк таблицы с помощью единственной инструкции!*

Истинная мощь этой инструкции jQuery заложена в *селекторе* – выражении идентификации элементов, позволяющем идентифицировать и отбирать нужные элементы страницы. В данном случае – каждый четный элемент `<tr>` во всех таблицах. Полный исходный код этой страницы вы найдете среди загружаемых примеров этой книги, в файле *chapter1/zebra.stripes.html*.

Позднее мы узнаем, как легко создавать эти селекторы, но сначала посмотрим, как изобретатели jQuery представляют себе эффективное использование JavaScript в наших страницах.

1.2. Ненавязчивый JavaScript

Помните суровые времена до появления CSS, когда мы были вынуждены смешивать в HTML-страницах стилевую разметку со структурой документа?

Эти воспоминания почти наверняка заставят содрогнуться любого, кто создавал тогда страницы. Добавление CSS в арсенал инструментов веб-разработчика позволяет отделить стилистическую информацию от структуры документа и проводить на заслуженный отдых такие теги, как `<font>`. Отделение стиля от структуры не только упрощает управление документами, но и придает им гибкость полного изменения стиля отображения страницы простой заменой таблицы стилей.

Немногие из нас захотели бы вернуться в прошлое, когда стили отображения применялись непосредственно к HTML-элементам. И все же до сих пор очень популярна разметка вроде этой:

```
<button
  type="button"
  onclick="document.getElementById('xyz').style.color='red';">
    Click Me
</button>
```

Здесь видно, что стиль элемента-кнопки, включая шрифт надписи на ней, определяется не тегом `<font>` или другой нежелательной разметкой стиля, а задается правилами CSS, действующими в пределах страницы. В этом объявлении не смешивается разметка стиля и структура, однако здесь налицо смешение *поведения* и структуры за счет включения кода JavaScript, выполняемого по щелчку на кнопке, в код разметки элемента `<button>` (в данном случае щелчок на кнопке вызывает окрашивание в красный цвет некоторого элемента объектной модели документа (Document Object Model, DOM) с именем `xyz`).

По тем же причинам, по которым желательно отделять стиль от структуры HTML-документа, точно так же (если не более) желательно было бы отделить *поведение* элементов от структуры.

Такого рода отделение известно под названием *ненавязчивый JavaScript*, и разработчики jQuery сосредоточили свои усилия на том, чтобы

помочь авторам страниц следовать ему. С точки зрения концепции ненавязчивого JavaScript и библиотеки jQuery наличие выражений или инструкций на языке JavaScript в теге `<body>` HTML-страниц, в атрибутах HTML-элементов (например, `onclick`) либо в блоках сценариев в теле страницы считается неправильным.

Вы можете спросить: «Как же пользоваться кнопкой без атрибута `onclick`?» Рассмотрим такое определение кнопки:

```
<button type="button" id="testButton">Click Me</button>
```

Оно гораздо проще! Но теперь кнопка просто *ничего не делает*.

Вместо того чтобы встраивать определение поведения кнопки в ее разметку, мы поместим его в блок сценария в разделе `<head>` страницы, за пределами тела документа, как показано ниже:

```
<script type="text/javascript">
  window.onload = function() {
    document.getElementById('testButton').onclick = makeItRed;
  };

  function makeItRed() {
    document.getElementById('xyz').style.color = 'red';
  }
</script>
```

В обработчике события страницы `onload` мы связываем функцию `makeItRed()` с атрибутом `onclick` элемента-кнопки. Обработчик события `onload` (а не встроенный код) используется потому, что нам нужно, чтобы элемент-кнопка уже *существовал* к тому моменту, как мы попытаемся манипулировать им. (В разд. 1.3.3 будет показано, что jQuery предоставляет лучший способ размещения такого программного кода.)

Если в этом примере что-то покажется вам непонятным, не пугайтесь! В приложении приведен обзор понятий JavaScript, которые помогут вам эффективно использовать jQuery. Кроме того, в оставшейся части главы мы узнаем, как jQuery позволяет писать код, аналогичный предыдущему, более короткий, более простой и одновременно более гибкий.

Ненавязчивый JavaScript – это мощная методика, позволяющая разделить обязанности в веб-приложениях, но за это приходится платить. Возможно, вы уже обратили внимание, что для достижения поставленной цели нам потребовалось добавить немного больше строк кода, чем в случае, когда код JavaScript помещался непосредственно в код разметки. Ненавязчивый JavaScript не только может привести к увеличению объема программного кода, но также требует определенной дисциплины и применения хорошо зарекомендовавших себя шаблонов программирования клиентских сценариев.

Но в этом нет ничего плохого – все, что побуждает нас с заботой и уважением писать свой клиентский код, обычно размещаемый на сервере, нам только на пользу. Но без jQuery для этого пришлось бы *выполнить* лишнюю работу.

Как уже говорилось, разработчики jQuery сосредоточили свои усилия на том, чтобы упростить нам применение методик ненавязчивого JavaScript при программировании страниц, не платя за это лищним программным кодом. Мы считаем, что эффективное использование jQuery позволяет нам добиться больших возможностей за счет меньшего объема сценариев.

А теперь спокойно рассмотрим, как библиотека jQuery позволяет расширять функциональность страниц без лишних усилий и головной боли.

1.3. Основы jQuery

Суть jQuery в том, чтобы отбирать элементы HTML-страниц и выполнять над ними некоторые операции. Если вы знакомы с CSS, то хорошо понимаете, насколько удобны селекторы, которые описывают группы элементов, объединенные по каким-либо атрибутам или по местоположению в документе. Благодаря jQuery вы сможете, используя свои знания, значительно упростить код JavaScript.

Библиотека jQuery в первую очередь обеспечивает непротиворечивую работу программного кода во всех основных типах броузеров, решая такие сложные проблемы JavaScript, как ожидание загрузки страницы перед тем, как выполнить какие-либо операции.

На тот случай, если в библиотеке обнаружится недостаток функциональности, разработчики предусмотрели простой, но весьма действенный способ ее расширения. Многие начинающие программисты jQuery обнаруживают эту гибкость на практике, расширяя возможности jQuery в первый же день.

Но для начала давайте посмотрим, как знание CSS помогает создать краткий, но мощный программный код.

1.3.1. Обертка jQuery

С введением CSS в веб-технологии для отделения представления от содержимого понадобился способ, позволяющий ссылаться на группы элементов страницы из внешних таблиц стилей. В результате был разработан метод, основанный на использовании *селекторов*, которые представляют элементы на основе их атрибутов или местоположения в HTML-документе.

Например, селектор

```
p a
```

ссылается на все ссылки (элементы `<a>`), вложенные в элементы `<p>`. Библиотека jQuery использует те же самые селекторы и поддерживает не только обычные селекторы, применяемые сегодня в CSS, но и другие, еще не полностью реализованные в большинстве броузеров. Селектор

`nth-child` из примера с полосатой таблицей, приведенного выше, – отличный пример применения селектора, определенного в спецификации CSS3.

Синтаксис отбора группы элементов прост:

```
$(selector)
```

или

```
jQuery(selector)
```

Функция `$()`, на первый взгляд, необычна, но большинство пользователей быстро начинают применять ее благодаря ее краткости.

Например, чтобы получить группу ссылок, вложенных в элементы `<p>`, можно использовать следующий код:

```
$("p a")
```

Функция `$()` (псевдоним функции `jQuery()`) возвращает специальный объект JavaScript, который содержит массив элементов DOM, соответствующих указанному селектору. У этого объекта много удобных предопределенных методов, способных воздействовать на группу элементов.

На языке программирования такого рода конструкция называется *оберткой* (*wrapper*), потому что она «обертывает» отобранные элементы дополнительной функциональностью. Мы будем использовать термин *обертка jQuery*, или *обернутый набор*, ссылаясь на группы элементов, управлять которыми позволяют методы, определенные в jQuery.

Предположим, нам требуется реализовать постепенное исчезновение всех элементов `<div>` с классом CSS `notLongForThisWorld`. jQuery позволяет сделать это так:

```
$("div.notLongForThisWorld").fadeOut();
```

Особенность многих из этих методов, часто называемых *командами* jQuery, состоит в том, что по завершении своих действий (например, действия, обеспечивающего постепенное исчезновение) они возвращают ту же самую группу элементов, готовую к выполнению другой операции. Предположим, что после того, как элементы исчезнут, к ним нужно добавить класс CSS `removed`. Записать это можно так:

```
$("div.notLongForThisWorld").fadeOut().addClass("removed");
```

Такую *цепочку* (*chain*) команд jQuery можно продолжать до бесконечности. Вы без труда сможете отыскать в Интернете примеры цепочек jQuery, состоящих из десятков команд. А поскольку каждая функция работает сразу со всеми элементами, соответствующими указанному селектору, вам не потребуется выполнять обход массива элементов в цикле. Все нужное происходит за кулисами!

Но даже при том, что группа отобранных элементов представлена довольно сложным объектом JavaScript, в случае необходимости мы мо-

жем обращаться к ней как к обычному массиву элементов. В итоге следующие две инструкции дают идентичные результаты:

```
$("#someElement").html("Добавим немного текста");
```

или

```
$("#someElement")[0].innerHTML =
  "Добавим немного текста";
```

Поскольку здесь мы использовали селектор по атрибуту ID, ему будет соответствовать один элемент. В первом случае используется метод библиотеки jQuery – `html()`, который замещает содержимое элемента DOM некоторой HTML-разметкой. Во втором случае с помощью jQuery извлекается массив элементов, выбирается первый элемент массива с индексом `0` и затем его содержимое замещается с помощью самого обычного метода JavaScript.

Если мы захотим получить тот же результат с помощью селектора, который может отобрать несколько элементов, то можно использовать любой из двух следующих способов, дающих идентичные результаты:

```
$("div.fillMeIn")
  .html("Добавим немного текста в группу элементов");
```

или

```
var elements = $("div.fillMeIn");
for(i=0;i<elements.length;i++)
  elements[i].innerHTML =
    "Добавим немного текста в группу элементов";
```

С увеличением сложности поставленных задач способность jQuery создавать цепочки команд по-прежнему способствует уменьшению числа строк программного кода, необходимого для получения желаемых результатов. Библиотека jQuery поддерживает не только селекторы, которые вы уже знаете и любите, но и более сложные селекторы, определенные как часть спецификации CSS, и даже некоторые нестандартные селекторы.

Вот несколько примеров.

```
$("p:even");
```

Этот селектор отбирает все четные элементы `<p>`.

```
$("tr:nth-child(1)");
```

Этот селектор отбирает первые строки во всех таблицах.

```
$("body > div");
```

Этот селектор отбирает элементы `<div>`, являющиеся прямыми потомками элемента `<body>`.

```
$("a[href$=pdf]");
```

Этот селектор отбирает ссылки на файлы PDF.

```
$("body > div:has(a)")
```

Этот селектор отбирает элементы `<div>`, которые являются прямыми потомками элемента `<body>` и содержат ссылки.

Мощная штука!

Для начала вы можете использовать свое знание CSS, а затем изучите более сложные селекторы, поддерживаемые библиотекой. Очень подробно селекторы будут рассматриваться в разд. 2.1, а их полный перечень вы найдете по адресу *http://docs.jquery.com/Selectors*.

Выбор элементов DOM для выполнения операций – это самое обычное дело для наших страниц, но в некоторых случаях требуется выполнить какие-либо действия, никак не связанные с элементами DOM. Давайте коротко рассмотрим, что еще может предложить jQuery помимо манипулирования элементами.

1.3.2. Вспомогательные функции

Даже при том, что обертывание элементов для выполнения операций над ними – основное применение функции `$()` в библиотеке jQuery, это не единственная ее возможность. Одна из ее дополнительных возможностей – выступать в качестве префикса *пространства имен* (*namespace*) для вспомогательных функций общего назначения. Обертка jQuery, получаемая в результате вызова `$()` с селектором, предоставляет авторам страниц максимум возможностей, поэтому большинство авторов страниц использует другие возможности, предоставляемые вспомогательными функциями, сравнительно редко. Мы не будем подробно рассматривать эти функции до главы 6, где описаны подготовительные действия для создания подключаемых модулей jQuery. Но некоторые из этих функций *встретятся* вам в следующих разделах, поэтому мы опишем их здесь.

Поначалу способ обращения к этим функциям может казаться немного странным. Давайте рассмотрим пример использования вспомогательной функции, которая усекает строки. Вызов этой функции выглядит так:

```
$.trim(someString);
```

Если префикс `$.` кажется вам странным, запомните, что `$` – это обычный идентификатор JavaScript, такой же, как и любой другой. Вызов той же самой функции с использованием идентификатора `jQuery` выглядит более знакомым:

```
jQuery.trim(someString);
```

Здесь совершенно очевидно, что функция `trim()` принадлежит пространству имен `jQuery` с псевдонимом `$`.

Примечание

В документации эти элементы называются вспомогательными *функциями*, но совершенно очевидно, что в действительности они являются *методами* функции `$()`. Не углубляясь в технические детали, мы также будем называть эти методы *вспомогательными функциями*, чтобы терминология соответствовала электронной документации.

Одну такую вспомогательную функцию, позволяющую расширять jQuery, мы рассмотрим в разд. 1.3.5, а другую, позволяющую jQuery мирно сосуществовать с другими клиентскими библиотеками, – в разд. 1.3.6. Но для начала рассмотрим еще один важный аспект функции `$`.

1.3.3. Обработчик готовности документа

Методика ненавязчивого JavaScript позволяет отделить поведение элементов от структуры документа, поэтому мы будем манипулировать с элементами страницы за пределами разметки документа, где создаются эти элементы. Для этого нам нужен механизм, позволяющий дождаться окончания загрузки элементов DOM страницы и только после этого выполнить необходимые операции. В примере с полосатой таблицей мы должны дождаться момента, когда будет загружена вся таблица, и только потом изменить цвет фона ее строк.

Традиционно для этих целей используется обработчик `onload` объекта `window`, который выполняет инструкции после того, как страница будет загружена целиком. Обычно он вызывается так:

```
window.onload = function() {
  $("table tr:nth-child(even)").addClass("even");
};
```

Тем самым программный код раскрашивания таблицы вызывается только после того, как документ полностью загружен. К сожалению, броузер задерживает выполнение обработчика `onload` не только до момента создания полного дерева DOM, но также ждет, пока будут загружены все изображения и другие внешние ресурсы и страница отобразится в окне броузера. В результате посетитель может заметить задержку между тем моментом, когда он впервые увидит страницу, и тем, когда будет выполнен сценарий `onload`.

Хуже того, если изображение или другой ресурс загружается достаточно долго, посетитель вынужден ждать окончания его загрузки, прежде чем дополнительные особенности поведения элементов станут доступными. Во многих реальных применениях это могло бы обречь идею ненавязчивого JavaScript на неудачу.

Намного лучший подход заключается в том, чтобы задерживать запуск сценариев, обеспечивающих дополнительные особенности поведения, *только* до момента окончания загрузки структуры документа,

когда HTML-код страницы будет преобразован броузером в дерево DOM. Добиться этого независимым от типа броузера способом достаточно сложно, но библиотека jQuery предоставляет простой способ запуска программного кода сразу после загрузки дерева DOM, но до загрузки внешних изображений. Формальный синтаксис определения такого кода (из примера с таблицей):

```
$(document).ready(function() {
  $("table tr:nth-child(even)").addClass("even");
});
```

Сначала производится обертывание экземпляра документа в функцию `jQuery()`, а затем применяется метод `ready()`, которому функция передается для исполнения после того, как документ станет доступным для манипуляций.

Этот синтаксис мы назвали *формальным* потому, что гораздо чаще используется сокращенная форма его записи:

```
$(function() {
  $("table tr:nth-child(even)").addClass("even");
});
```

Вызывая функцию `$()`, мы тем самым предписываем броузеру дождаться, пока дерево DOM (и только дерево DOM) будет полностью загружено, прежде чем выполнить этот код. Более того, в одном и том же HTML-документе мы можем применять этот прием многократно, и броузер выполнит все указанные функции в порядке следования объявлений. Напротив, методика на базе обработчика `onload` объекта `window` позволяет указать единственную функцию. Это ограничение чревато трудноуловимыми ошибками, если какой-либо сторонний сценарий использует механизм `onload` для собственных нужд (что никак нельзя признать хорошей практикой).

Мы познакомились со вторым назначением функции `$()`. Давайте посмотрим, что еще она может нам предложить.

1.3.4. Создание элементов DOM

Совершенно очевидно, что авторы jQuery избегали вводить глобальные идентификаторы в пространство имен JavaScript, сделав функцию `$()` (которая является обычным псевдонимом функции `jQuery()`) достаточно универсальной для выполнения множества операций. Итак, у этой функции есть еще одна особенность, которую мы хотим исследовать.

Передавая функции `$()` строку с кодом разметки HTML-элементов дерева DOM, мы можем создавать эти элементы на лету. Например, так можно создать новый элемент-абзац:

```
$("<p>Hi there!</p>")
```

Но в создании ни с чем не связанных элементов DOM (или иерархии элементов) нет никакого смысла. Обычно после создания иерархии элементов задействуются другие функции jQuery для манипулирования деревом DOM.

Давайте исследуем в качестве примера листинг 1.1.

Листинг 1.1. Создание элементов HTML на лету

```
<html>
  <head>
    <title>Follow me!</title>
    <script type="text/javascript" src="../scripts/jquery-1.2.js">
    </script>
    <script type="text/javascript">
      $(function(){
        $("<p>Hi there!</p>").insertAfter("#followMe");
      });
    </script>
  </head>

  <body>
    <p id="followMe">Follow me!</p>
  </body>
</html>
```

❶ Обработчик готовности документа, создающий элементы HTML

❷ Существующий элемент, за которым будет вставлен новый

В этом примере в теле документа определяется существующий HTML-элемент абзаца с именем `followMe` ❷. В сценарии, который находится в разделе `<head>`, определяется сценарий обработчика события готовности документа ❶. Этот сценарий добавляет вновь создаваемый абзац в дерево DOM сразу вслед за существующим элементом:

```
$("<p>Hi there!</p>").insertAfter("#followMe");
```

Результат показан на рис. 1.2.

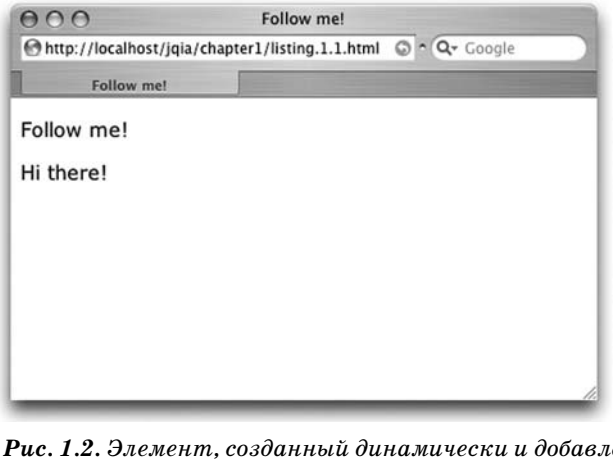


Рис. 1.2. *Элемент, созданный динамически и добавленный в документ*

Полный комплект функций манипулирования деревом DOM мы рассмотрим в главе 2, где вы увидите, что jQuery предоставляет много функций манипулирования DOM, позволяющих придать документу желаемую структуру.

Теперь, когда вы познакомились с базовым синтаксисом jQuery, давайте взглянем на одну из самых мощных особенностей этой библиотеки.

1.3.5. Расширение jQuery

Функция-обертка jQuery обеспечивает доступ к многим полезным функциям, которые мы постоянно будем использовать в страницах. Но ни одна библиотека не в состоянии удовлетворить все потребности без исключения. Можно даже утверждать, что ни одна библиотека не должна стремиться удовлетворить все возможные потребности, иначе появится огромная неуклюжая масса программного кода, содержащая редко используемые функции, которые только мешают работе!

Понимая это, авторы библиотеки jQuery прилагают усилия для выявления функциональных особенностей, востребованных большинством авторов страниц, чтобы включать в библиотеку только такие особенности. Ввиду того что у каждого автора страниц могут быть собственные неповторимые потребности, библиотека jQuery была создана легко расширяемой.

Но зачем расширять jQuery – ведь для восполнения недостатка функциональности можно писать автономные функции!

Все просто: расширяя библиотеку, мы можем использовать ее мощные особенности, в частности возможность выбора элементов.

Рассмотрим конкретный пример: в библиотеке jQuery отсутствует предопределенная функция, которая деактивировала бы элементы форм. Если во всех приложениях используются формы, пригодилась бы возможность применять, к примеру, такой синтаксис:

```
$("form#myForm input.special").disable();
```

К счастью, архитектура jQuery позволяет легко расширять имеющийся набор функций, расширяя обертку, возвращаемую вызовом `$()`. Рассмотрим базовую идиому, позволяющую этого достигнуть:

```
$.fn.disable = function() {
  return this.each(function() {
    if (typeof this.disabled != "undefined") this.disabled = true;
  });
}
```

Здесь много новых синтаксических конструкций, но не стоит из-за них сильно волноваться. Волнения сами собой улягутся, когда вы прочитаете следующие несколько глав. Эту базовую идиому вы еще много раз примените на практике.

Во-первых, конструкция `$.fn.disable` означает, что мы расширяем обертку `$` функцией с именем `disable`. Внутри этой функции коллекцию обернутых элементов DOM, над которыми будет выполняться операция, представляет идентификатор `this`.

Во-вторых, метод `each()` в этой обертке вызывается для обхода всех элементов коллекции. Подробнее этот и подобные ему методы рассматриваются в главе 2. Внутри функции, передаваемой методу `each()`, ключевое слово `this` является ссылкой на конкретный элемент DOM в текущей итерации. Пусть вас не смущает тот факт, что внутри вложенной функции ссылка `this` указывает на различные объекты. Когда вы напишете несколько функций расширения, это станет привычным и естественным.

Для каждого элемента выполняется проверка – есть ли у текущего элемента атрибут `disabled`, и если такой атрибут имеется, ему присваивается значение `true`. Результат метода `each()` (обертка) возвращается в вызывающую программу, чтобы обеспечить возможность составления цепочек нашим новым методом `disable()`, как это делают многие методы библиотеки jQuery. После этого можно написать такую инструкцию:

```
$("form#myForm input.special").disable().addClass("moreSpecial");
```

С точки зрения кода нашей страницы все выглядит так, как если бы наш новый метод `disable()` был встроен непосредственно в библиотеку! Этот прием обладает такой мощью, что большинство пользователей, начинающих изучать jQuery, обнаруживают в себе способность создавать небольшие расширения для jQuery, как только начали применять библиотеку.

Более того, наиболее инициативные пользователи расширили возможности jQuery наборами полезных функций, которые называются *подключаемыми модулями*, *модулями расширения* (*plugins*). Мы еще не раз поговорим об этом способе расширения jQuery, а в главе 9 представим вашему вниманию свободно распространяемые официальные модули расширения.

Прежде чем углубиться в тонкости использования jQuery, чтобы вдохнуть жизнь в наши страницы, любопытный спросит, можно ли применять jQuery совместно с другими библиотеками, такими как Prototype, ведь в них, как известно, тоже есть сокращение `$`. Ответ на этот вопрос вы найдете в следующем разделе.

1.3.6. Сочетание jQuery с другими библиотеками

jQuery предоставляет в распоряжение программиста набор мощных инструментов, способных удовлетворить насущные потребности большинства авторов страниц, но даже при этом иногда в странице требуется задействовать возможности нескольких библиотек JavaScript. Такие ситуации могут возникать, например, во время перевода приложения