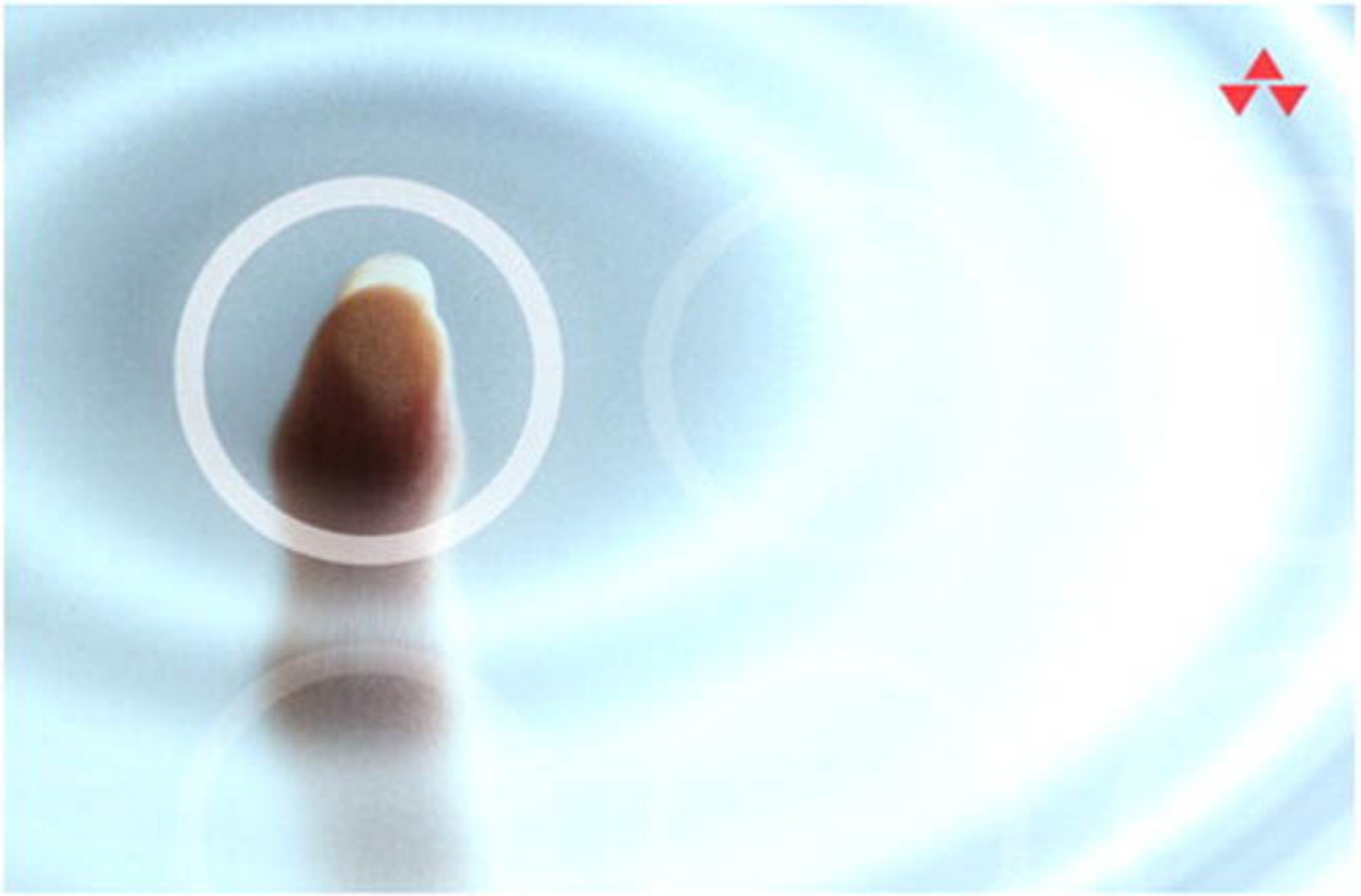


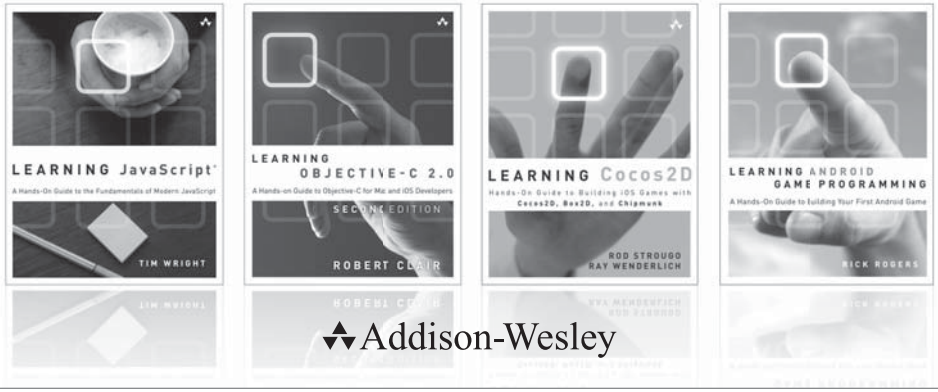
LEARNING jQuery

A Hands-on Guide to Building Rich Interactive Web Frontends

Learning jQuery

A Hands-on Guide to Building
Rich Interactive Web Front Ends

Addison-Wesley Learning Series



Visit informit.com/learningseries for a complete list of available publications.

The Addison-Wesley Learning Series is a collection of hands-on programming guides that help you quickly learn a new technology or language so you can apply what you've learned right away.

Each title comes with sample code for the application or applications built in the text. This code is fully annotated and can be reused in your own projects with no strings attached. Many chapters end with a series of exercises to encourage you to reexamine what you have just learned, and to tweak or adjust the code as a way of learning.

Titles in this series take a simple approach: they get you going right away and leave you with the ability to walk off and build your own application and apply the language or technology to whatever you are working on.

Learning jQuery

A Hands-on Guide to Building Rich Interactive Web Front Ends

Ralph Steyer

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Learning jQuery: A Hands-on Guide to Building Rich Interactive Web Front Ends

First published in the German language under the title jQuery by Addison-Wesley, an imprint of Pearson Education Deutschland GmbH, München. Copyright © 2011 by Pearson Education Deutschland GmbH.

ISBN-13: 978-0-321-81526-2
ISBN-10: 0-321-81526-2

Library of Congress Cataloging-in-Publication Data is on file.

Printed in the United States of America

First Printing May 2013

Trademarks

Warning and Disclaimer

Bulk Sales

Pearson offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales
international@pearsoned.com

Editor in Chief
Mark Taub

Acquisitions Editor
Mark Taber

Managing Editor
Sandra Schroeder

Project Editor
Mandie Frank

Copy Editor
Keith Cline

Indexer
Larry Sweazy

Proofreader
Megan Wade

Translator
Almut Dworak

Technical Editor
Brad Dayley

Publishing Coordinator
Vanessa Evans

Designer
Chuti Prasertsith

Compositor
Jake McFarland

Contents at a Glance

Table of Contents

About the Author

Ralph Steyer is a computer programmer, consultant, journalist, and book author with decades of experience in a wide variety of computer programming languages and technologies. He has a degree in mathematics from Frankfurt/Main University and is the author of several books on web programming, including *JavaScript Handbook* and *AJAX Frameworks* (Addison-Wesley).

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

You can email or write directly to let us know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that we cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail we receive, we might not be able to reply to every message.

When you write, please be sure to include this book's title and author, as well as your name and phone or email address.

Email:	feedback@developers-library.info
Mail:	Reader Feedback Addison-Wesley Developer's Library 800 East 96th Street Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at www.informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

Your purchase of this book includes access to a free online edition for 45 days through the Safari Books Online subscription service. Details are on the last page of this book.

This page intentionally left blank

1

Introduction

Rich Internet Applications (RIAs), with their somewhat vaguely worded *rich* opportunities, have significantly changed the way we use the Web over the past few years. And the speed of this change keeps increasing. The significance of classic desktop applications is being repositioned. Many types of programs that were traditionally used only as desktop application now suddenly appear on the Web, be it personal calendars, entire office programs, games, route planners, or communication programs. But apps for cell phones or smartphones are also increasingly based on web technology. This changes both the user behavior and the user expectation for Internet applications in general and the availability of services. As classic web applications, but with a certain extra value, RIAs are, on the one hand, always available if you have a halfway decent Internet access and a modern browser.[1] On the other hand, they are hardly distinguishable any more from classic desktop or mobile apps in terms of operation, performance, and visual appearance.

The most effective way to ensure that these rich opportunities are available usually involves using an appropriate web framework. Be aware, however, that if you use a framework, you become significantly dependent on a manufacturer or a project, and that you then no longer have complete control over the source code in your applications. In any case, using frameworks requires familiarizing yourself sufficiently with the relevant function libraries and working methods of the system. In contrast to grandiose advertising claims of some frameworks (and some tools), you can usually use them effectively only after you understand web programming concepts and have at least a basic knowledge of the underlying technology. Strictly speaking, you will profit most from frameworks the less you actually need them and the more you master the basics.

Regardless of these problems and disadvantages, however, there is much to be said for making use of frameworks and toolkits. They will certainly help you develop and maintain sophisticated websites much more quickly, effectively, and efficiently; and they enable you to offer a richer and more robust site.

1. The browser becomes a multifunctional access instrument for a specific task and thus replaces classic application types. In the future, users might only need a browser as application, or the operating system and the browser may merge so that they become indistinguishable.

> **Note**
>
> The preceding text included the terms *framework* and *toolkit* a few times. There is no standard definition for exactly what a framework is and how it differs from a toolkit. In fact, a reliable definition and differentiation is not very straightforward. But generally, the term *framework* implies a programming framework that already offers certain functionalities. A framework is not yet a finished program in itself, but merely provides a frame within which one or several programmers can create an application. A framework usually contains a library with useful predefined code structures, but also (in contrast to a pure library) specifies a certain control of the behavior patterns involved in using it (for example, a syntax or grammar). With a toolkit, the main focus is on a collection of programs (tools), but these can also be based on specific libraries or a syntax concept. Both a framework and, in particular, toolkits often provide widgets or components—in other words, elements that constitute a graphical user interface (GUI).

1.1 What Is This Book About?

This book provides an easy introduction to web programming with **jQuery**, **jQuery UI**, and **Mobile jQuery**. jQuery is a free and comprehensive framework built on top of the JavaScript language. It was originally developed by John Resig and released in January 2006 at BarCamp (NYC). It is now consistently developed further as an open source project. jQuery UI is built on top of jQuery and extends the jQuery framework with UI specific components. Similarly, jQuery Mobile is also built on top of jQuery and extends the jQuery framework with mobile device-specific components.

The framework offers a whole range of very helpful features (for example, easy-to-use functions for DOM manipulation and navigation, as well as basic AJAX support). Beyond this, the framework offers support for Cascading Style Sheets (CSS), an expanded event system, impressive effects and animations, various auxiliary functions, and numerous free plug-ins.

But where jQuery particularly excels is the seamless integration of the framework in many web platforms by large industry providers or their official support. For example, Microsoft uses jQuery in the development environment Visual Studio in combination with the ASP.NET MVC framework and Microsoft Asynchronous JavaScript and XML (AJAX). For example, if you create a new ASP.NET project, you can also integrate jQuery automatically (although not necessarily in the latest version).

In an ASP.NET MVC 3 or later web application in Visual Studio, you can even choose to integrate some jQuery plug-ins (such as jquery.validate.js for validating user input).

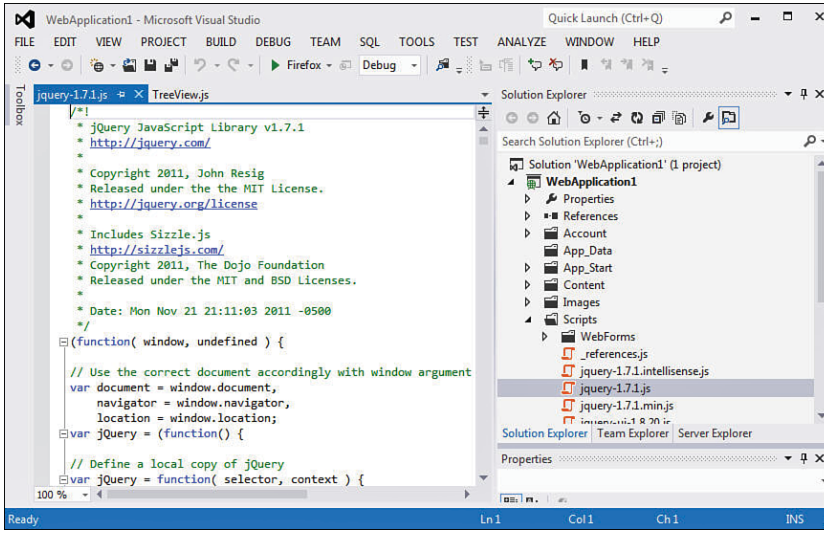


Figure 1.1 In an ASP.NET web application in Visual Studio 2010, you can also integrate jQuery.

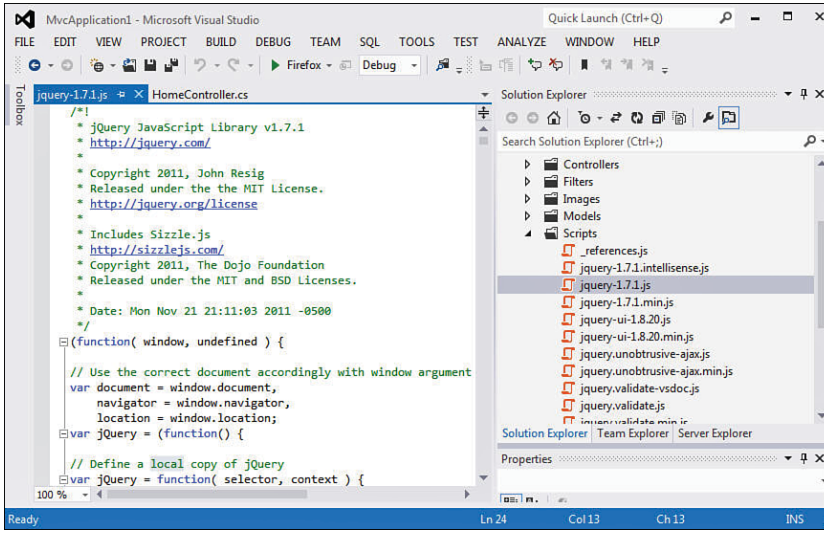


Figure 1.2 An ASP.NET MVC 4 web application with references to jQuery itself plus several jQuery plug-ins.

The links to the libraries are already created in the pregenerated source text and can be simply enabled by deleting the comment characters.

```
<html>
<head>
    <title>@View.Title</title>
    <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />
    @*
    <script src="@Url.Content("~/Scripts/jquery-1.4.1.min.js")" type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.unobtrusive-ajax.min.js")" type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")" type="text/javascript"></script>
    *@
</head>

<body>
```

Figure 1.3 In the generated code, you just need to enable the script references.

In addition to Visual Studio, various other web development tools offer jQuery. But even other suppliers with a different focus use jQuery (for example, the cell phone manufacturer Nokia in its Web Runtime platform, as well as Google, Dell, Mozilla, WordPress, Drupal, and Digg). The list of popular users reads like a Who's Who of the Web. The framework is also extremely popular on a wide scale, as many statistics prove. If you look at modern RIAs, more than 60% of them currently use jQuery and the jQuery UI, despite its powerful and impressive competitors.

1.1.1 What You Can Learn from This Book

In this book, you learn how to use jQuery for your own web applications—from simple websites to which you only want to add individual effects, right up to complex RIAs. This book is aimed at beginners, without starting right at zero, but it is not intended for an audience of freaks and programming experts either. You do not need to have lots of experience with AJAX or a framework or toolkit. But you should have a little bit of experience with web technology—more on this shortly when you read about the target audience.

The book follows the same basic structure in each chapter. A brief introduction precedes the more detailed topics, and a summary concludes the chapter.

The specific approach is this: After this chapter, which already provides all requirements for working with jQuery, we just jump in at the deep end and work through a few examples without much preparation. This is meant to give a feel for what you can do with jQuery.

We then cover some basic background information about the Web, JavaScript, AJAX, Extensible Markup Language (XML), JavaScript Object Notation (JSON), and so on. Then, we take a closer look at what working with jQuery involves. Next, we turn to selectors and filters. I believe that these options for selecting objects in the context of a website are one of the biggest highlights of the framework and form the basis for accessing the elements of a website. Many examples are provided to help you better understand what we mean.

Next comes the topic Dynamic Hypertext Markup Language (DHTML). In DHTML, the main focus is on changing websites based on certain events. Essentially (or at least in most cases), DHTML means the dynamic influencing of CSS properties. Once more, jQuery offers many

options for making this task much easier and compensating for numerous browser incompatibility issues.

In the previous paragraph, I used the word *event*. Event handling on the Web is a bottomless pit with many browsers. jQuery provides a solution. In this book, you learn how.

For many visitors, effects and animations are an eye-catcher in a website. Again, jQuery has a whole range of horses in the stable that need not fear the competition in this race.

Then we explicitly venture into the Web 2.0. So, we turn to AJAX and look at what jQuery can offer in this respect.

That is really all on the topic of jQuery. But wait a sec, wasn't there something else? The jQuery UI! So far, I have hardly mentioned it when describing the topics contained in this book. You might now think that the jQuery UI is something like the ugly duckling in the jQuery universe. Or perhaps uninteresting. This is far from the truth. The jQuery UI is the beautiful swan. Purely in terms of visual appearance, the jQuery UI offers much more than jQuery itself, even though using it is much easier—as long as you understand jQuery. jQuery is the basis that makes life easier regarding source code and programming, whereas the jQuery UI builds on it as an independent framework and excels with visually advanced interface components and a CSS theme framework. Of course, we also take a closer look at the jQuery UI in this book and work through many examples with the various widgets it offers. In addition, you learn in detail how to use options, events, methods, and theming to adapt it further. The theme framework and the ThemeRoller of the jQuery UI are also covered in detail.

Then there are also plug-ins in jQuery, as extensions of the framework. You will learn how to use foreign plug-ins in case you cannot find a certain function in the jQuery and the jQuery UI core libraries, and you will learn how to create and publish your own plug-ins.

Last but not least, this book describes how you can create mobile apps based on jQuery. This involves using the mobile framework that is directly based on native jQuery (just as the jQuery UI is).

> **Note**
>
> To make things clearer, we often work with code examples in this book. You should type in the complete code examples yourself (and, of course, you can then also modify them and experiment with them if you like). However, you will also find the listings on the companion website for this book.

1.2 Writing Conventions

This book uses various writing conventions intended to help you keep track of things more easily. Important terms are **emphasized in bold**. Sometimes also in *italics*. Above all, you are meant to be able to see if it is normal text or `program code`. Keyboard shortcuts and some other special things are also highlighted. This formatting is used consistently throughout this book.

I also want to add a special comment on source text passages of complete listings. For all complete listings and some larger source code fragments, you will see **numbering** of the source text lines. The numbers are, of course, not part of the source text itself; they are only meant to make it easier for you to find your way around, to point out a new line in the source code, and to make it clearer which part of the source code I am referring to. In rare cases, it might be necessary because of technical reasons to split a source text line over several book lines. In this case, the numbering of the source code lines indicates which passages are to be written into one line in the editor. So long as you do not see a new line number in the book listing, you still have to type everything into a single line in the editor. This is particularly important for longer strings (texts in quotation marks) that must not be divided up into several lines.

1.3 Who Is the Target Audience for This Book?

It is always tricky to anticipate who may be interested in a particular topic. But I have certain ideas and by now quite a lot of experience from jQuery seminars about who will be interested in finding out more about creating RIAs in the context of jQuery, their potential reasons, and the most likely readership of this book. I assume that you have already created websites and have already been programming in one form or another. JavaScript would be a great basis, but other programming techniques are just as welcome, although your learning curve will be slightly steeper as you go along. Style sheets should also be a familiar concept to you. If you do not have any previous experience with creating websites or working with HTML or programming, this book will probably pose quite a challenge for you (but this should not discourage you from reading it). I also assume that you are tired of the limitations of a static HTML site. Perhaps you already have some experience with dynamic websites (at least as a user), and you would probably like to find an easy way to create such interactive modern sites. jQuery is a fantastic method for achieving this.

More and more programmers of powerful techniques and environments such as Java or .NET are pressing into the area of web programming. Correspondingly, I want to also address readers with this type of background knowledge. For programmers who switch over from such powerful and strict worlds, it is often hard to find their way in the seemingly trivial (but, in fact, rather distinctive) world of web programming.

1.4 What Do You Need?

Let's turn to the requirements you should meet for working with this book and jQuery.

1.4.1 Hardware and Operating System

We are dealing with the Internet. So, of course, you need to have a computer with Internet access. No special requirements apply as far as the computer itself is concerned, but your hardware should be at least reasonable quality. The requirements of modern operating systems already determine the minimum level of the required hardware. All graphic operating systems such as Linux, Windows, or OS X will work fine as long as they are relatively up-to-date. The

exact type of the system you are using is not relevant for our purposes, just as in most cases on the Web.

1.4.2 jQuery and jQuery UI

Of course, you need to have jQuery itself so that you can re-create the examples in this book. You also need—in the later part of this book—the jQuery UI. You can download the most recent version of jQuery as well as past releases from the jQuery website: http://docs.jquery.com/Downloading_jQuery.

> **Tip**
>
> You can also download the current release from the project's home page at http://jquery.com/. You will see a large button that loads the JavaScript library directly. At the time of this writing, the current release is version 1.9.1.

You can download different variations, basically a minified version without comments and redundant spaces or line breaks that is used mainly in production or an uncompressed version that has comments in the source code and is easy to read but larger. The function of both versions is the same; they contain a JavaScript file that generally has the name jquery.js. This file, which will usually also have a version number in the filename and a description of its specific variant depending on its type,[2] is the central library of the framework that you integrate into your websites. If you download a variant with a zip archive from the Internet, simply extract it. You then just need to reference the JavaScript file in your website following the usual rules (more on this later).

> **Tip**
>
> If you click the link for downloading the jQuery file, most browsers just display the file, without first giving you the option to save it. After all, it is a JavaScript library, and as such is usually displayed as pure text. By contrast, if you click a zip file, you usually get the option to save it via the browser's download dialog that pops up when you click the file. In case of the jQuery library, you can display the code and then click the browser's option for saving the page to save the jQuery library locally.

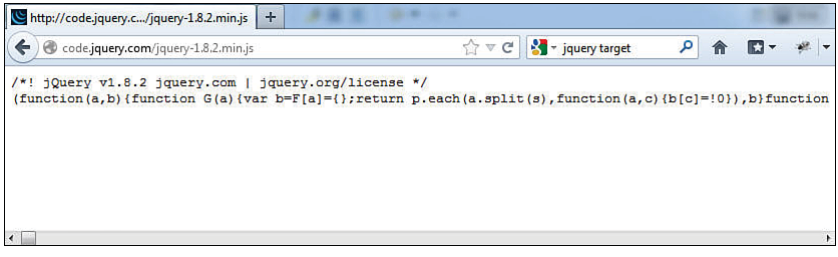


Figure 1.4 The jQuery file is displayed.

2. For example, jquery-1.7.2.min.js indicates the minified version 1.7.2, and jquery-1.7.2.js is the uncompressed version 1.7.2.

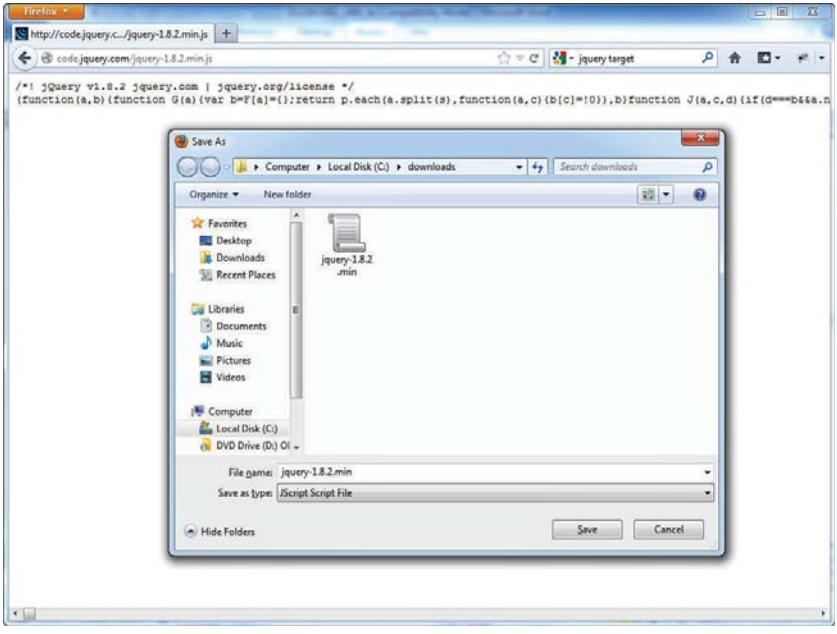


Figure 1.5 Use the browser's Save dialog to save the jQuery library locally.

If you want to use the jQuery UI, you need this framework, as well, because the jQuery UI is not contained in the normal jQuery JavaScript library. The jQuery UI is a separate project within the whole jQuery framework and contains other resources such as CSS files and graphics in addition to JavaScript files. You can find the home page of the project at http://jqueryui.com. There, you can load the framework via the **Download** link. Upon completion of the download, you get a compressed zip file that you can extract and make available on your server (just as with the jQuery library) and then integrate into your website via a central jQuery UI JavaScript file.

> **Caution**
>
> Regarding the versions, note that the jQuery UI versions always work with a specific version of jQuery itself and that incompatibilities can arise if the versions do not match. But the zip file always contains a version of jQuery, as well, the version that is the required minimum. There is more to be said about downloading the jQuery UI and its specific use, but I come back to this in more detail later in the chapters on the jQuery UI.

1.4.3 The Browsers

What you definitely need for programming with jQuery is, of course, a web browser that supports jQuery. After all, you want to be able to view your own sites so that you can test them. When using jQuery, you also need to take into account that the visitors of your websites have to comply with a certain minimum standard. As with most frameworks and toolkits, jQuery has anything but low requirements for the browser of a user who visits a website that works with jQuery. The minimum browser requirements may change with each new release of jQuery, but the following browsers are currently officially supported (you can check whether this still accurate at http://docs.jquery.com/Browser_Compatibility):

- Firefox[3]
- Internet Explorer
- Safari
- Opera
- Chrome[4]

Other browsers might work, but there is no official guarantee that they will. In the documentation, some browsers are officially listed as basically working, but with some known issues.

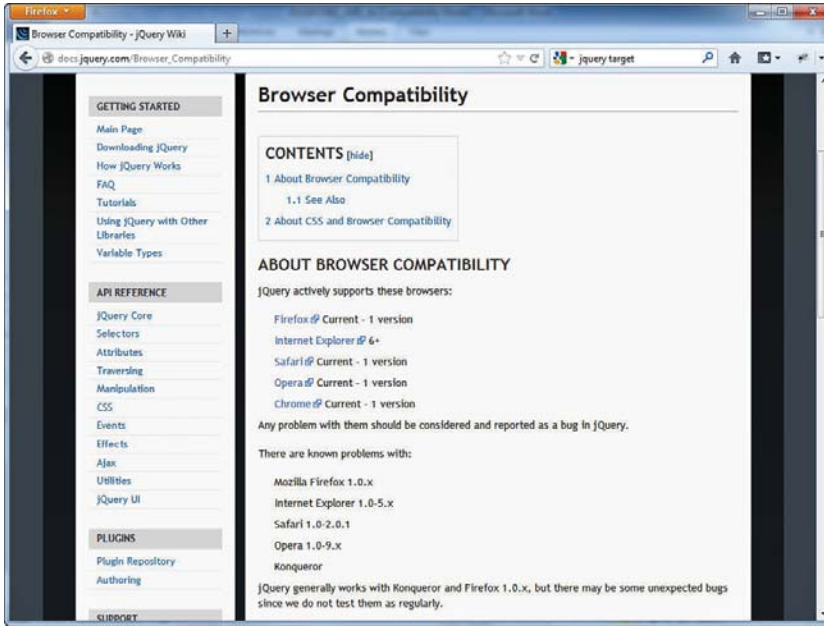


Figure 1.6 The officially supported browsers.

3. The same basically applies to all identical browsers, such as Netscape Navigator and Mozilla.

4. Including its safe and, in comparison with Google's data-gobbling habits, more reticent variant Iron.

> **Note**
>
> The version numbers are rather high, but users with older browsers hardly stand a chance in the current Web, even if we do not take jQuery into account. The situation has changed from what it was a few years ago. Now, more and more website creators no longer support as many browsers as possible. Instead, they explicitly set a minimum level. Certain functions cannot be realized at all in older browsers, or only with an exorbitant amount of effort, and hardly anyone is prepared to pay this price given that only a handful of users still have such old-timers. Some developers even cease supporting certain browser types, to give the impression that they are producing particularly modern websites. It seems to be almost a "mark of quality" if sites do not work with Internet Explorer or are not displayed correctly (as far as I can determine, this has almost become a national sport in the United States). This trend is essentially the opposite of what was going on about 10 years ago. Back then, you would find many websites with the note *Optimized for Internet Explorer*. Now, it seems that many websites want to demonstrate their exceptional quality by no longer working in Internet Explorer (at least in the really old versions, up to 7). Perhaps this is their motto: *We are making modern websites, and to use these, you need to have a modern and powerful browser.*

I believe that this is the wrong approach. I think that even though Internet Explorer 9 is now available and a really good browser, we should still offer a kind of protected-species support for versions 7 and 8 (and to a certain extent even version 6). After all, many users on the Internet are forced to use Internet Explorer[5] because of company politics or because they do not have sufficient knowledge to use an alternative browser or are just happy with a particular browser. I believe that jQuery is taking the sensible approach in supporting the older versions of Internet Explorer, and we can live with the few limitations of the jQuery UI regarding Internet Explorer; more on this in the relevant chapters later (and the explicit Microsoft support reflects this, too).

Unless you have one of the listed browsers, you cannot reliably test your jQuery web application. As a creator of websites, you should definitely have several browsers available. Because even when using a reliable and well-established framework like jQuery, you still need to test web applications in all relevant browsers.

And generally, you do not know which browsers the visitors of your websites are using. So, it is a good idea to test your website even with browsers that do not have completely guaranteed support for jQuery. For example, even though there are known problems with Firefox 1.0.x, Internet Explorer 1.0 to 5.x, Safari 1.0 to 2.0.1, Opera 1.0 to 9.x, and Konqueror, jQuery generally does work with Konqueror or Firefox 1.0.x. Just not with all components. Ultimately, it depends on the feature you are using, and you can go and test it in the relevant browsers.

1.4.4 Different Operating Systems and Virtual Machines for Testing

As mentioned previously, the choice of operating system for working with jQuery when creating a website is largely one of personal preference or any given constraints. This does not concern the choice of a test environment. Ideally, you have several operating systems available

5. Even still occasionally in the antiquated version 6, as I was appalled to realize in 2010/2011 during my seminars (and those were really big companies, too).

for testing your applications, because, after all, the visitors to your website will also be using different operating systems.

Of course, Windows is the reference system per se. The majority of users on the Web use this system. But Linux and OS X are also widely used, and there are different Windows versions, too. Sometimes it is very interesting to see how different web applications behave under different operating systems, although the differences should not really be significant. So, use different operating systems to test your applications if at all possible.

You do not need to have several computers or to install another operating system in parallel to your operating system. Especially for Linux, there are excellent live CDs or live DVDs from which you can launch the operating system directly without any changes being made to your hard disk. For readers with sufficiently capable hardware, it could also be interesting to have a closer look at a virtual solution (virtual machine [VM]) such as VMware (http://www.vmware.com), Virtual PC by Microsoft (http://www.microsoft.com/windows/virtual-pc/default.aspx), or VirtualBox (http://www.virtualbox.org/). These are available for free, at least for private use, and they simulate another operating system within the currently running operating system. For example, you can use these VMs to start a Linux system from within Windows or vice versa, or you can install another version of Internet Explorer under Windows parallel to your current Windows installation. With AJAX, the guest system (in other words, the system that runs in the VM) can act as server or client, and you therefore have two completely separate systems on one computer, enabling you to test a client/server relationship just as you would in reality.

1.4.5 The Web Server for Realistic Testing

With AJAX, data from a browser is requested from a web server and integrated into the website without reloading the site. Therefore, for practical use and for testing such applications, you must have access to a web server on the Internet and be able to execute programs and scripts on it. Ultimately, this is necessary for an AJAX project in practice, as well. However, in practice it is not usual to be working directly on a web server on the Internet while you are still developing a web application (especially if you just want to test a few things). But even without AJAX, to properly test a web application, you need to test it under realistic conditions on a web server.

For those reasons, you should create a test environment with a web server on a local computer or in a local network. Linux distributions, in particular, almost always contain one or more web servers. Different development environments for web applications also have an integrated web server. Then you are on the safe side. But even if you do not automatically have a web server available or simply want to make things as easy as possible, you can make use of an all-inclusive package such as XAMPP, which you can simply download for different operating systems from the Internet (at http://www.apachefriends.org).

This package is a collection of programs relating to the web server Apache, including the database management system MySQL (together with phpMyAdmin for administrating the database management system) and PHP support, the FTP server FileZilla, plus several other web technologies. You just install this package with a simple assistant, and then you have a fully functional web server in basic configuration at your disposal.

Caution

Note that in their default settings, these packages by XAMPP are for local testing purposes only. To make things as simple as possible, all security settings are at a minimum.

As soon as the installation of XAMPP completes, you can either launch Apache manually or set it up so that Apache is integrated as a service or process in your operating system and can even be launched automatically when the computer starts up. XAMPP offers a helpful and easy-to-use control panel.

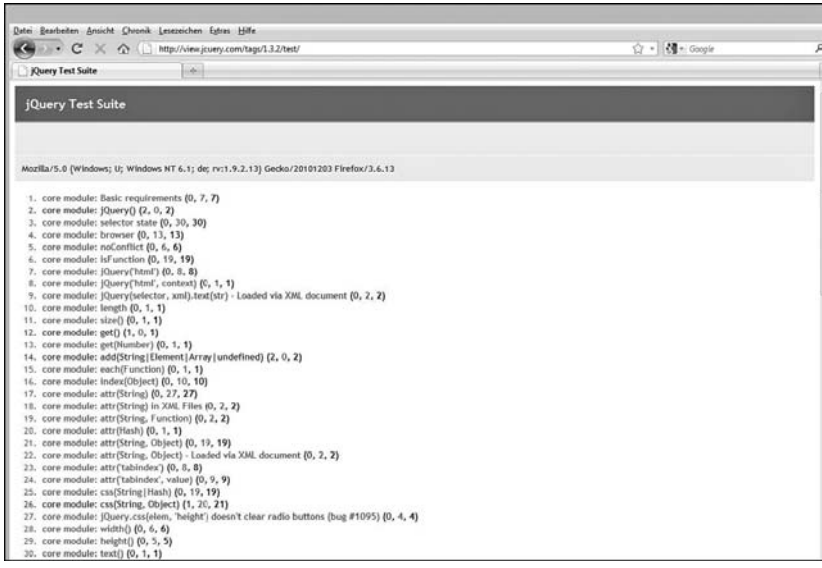


Figure 1.7 The XAMPP control panel: Apache, MySQL, and FileZilla are running.

Tip

Note that under XAMPP you have to follow the conventions commonly used on the Internet or Web regarding the path you specify. You cannot act as if you are working just under Windows (which, of course, means that you have to make sure that you work thoroughly and avoid potential problems right from the start). Under Apache, uppercase and lowercase is usually relevant. The best approach is to consistently use lowercase for directory names and filenames. And Windows users should note that you do not use a backslash for separating levels on the Internet, but instead use the slash.

1.4.6 The Development Tools

As backbone of an RIA, as for any website, you will almost always be using an HTML or XHTML frame. So, for creating the HTML source text and for all other techniques you use such as CSS, JavaScript, and so on, the minimum is a pure text editor, as included with any operating system.

In practice, though, you will probably use more powerful programming tools that support you in creating and analyzing the source text. Such programs may know some components of a programming or description language (such as HTML, CSS, or JavaScript) and support simple and sometimes even more complex standard processes, such as masking (the coded representation) of special characters, inserting source code templates, or aids for providing greater clarity by color-coding known commands. Some editors also offer the commands of a used language directly, which means the programmer can use menus or toolbars to choose them (sometimes with the mouse). Notepad++ (http://notepad-plus-plus.org/), for example, is an excellent editor offering this kind of support.

Another feature offered by some programs is different document views. This is often the case with pure HTML editors. You then have the choice of switching between the preview of a website (as it will look in the browser), a graphic editing mode, and above all a view of the HTML code itself.

Even in web programming, you can now make use of some proper integrated development environments (IDEs). These allow programming and executing from an integrated, common interface. A free yet very powerful IDE is Aptana (http://aptana.com/). It is based directly on the powerful development environment Eclipse (http://www.eclipse.org). Aptana offers a source code editor and has many features to directly support JavaScript, HTML, and CSS plus access to the DOM object model and even AJAX itself. The Code Assist function tries to autocomplete various user inputs, and syntax is marked with syntax highlighting (highlighting key terms and syntax structures in different colors). The option of displaying the properties and methods of objects is particularly interesting. The program even offers a debugger for JavaScript. Aptana also contains its own little web server (Jetty), via which you can test an AJAX application without installing an independent web server. If you create a project with Aptana, you will see that various popular JavaScript libraries are already integrated directly, among others jQuery, although there are usually more recent versions available on the Internet.

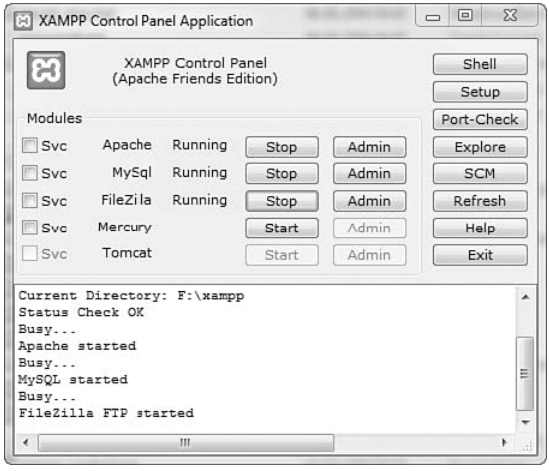


Figure 1.8 Aptana with direct support for jQuery.

> **Tip**
>
> Even if the version of jQuery available in Aptana for direct integration is not completely up-to-date, you should import the thus available (most recent, but potentially also outdated) jQuery library. You do not even need to use this outdated version. Instead, I recommend that you also download the most recent version separately and use this version in your web projects. But the import gives you the option of enabling in Aptana a code completion for the imported library (and this is not possible if you integrate the JavaScript file into a project via HTML alone). This code completion may then not be quite up-to-date, but is still immensely helpful.

Before you can use the code completion, you first have to complete the following steps:

1. Create a new web project via **File > New**.
2. Select **Default** from this list of available web project types.
3. After you have entered a name, you can either select a jQuery library directly (if it is already on offer), or you can click the **Install JavaScript Libraries** button. In the next dialog, you then choose **jQuery** under JavaScript Libraries. Where applicable, ensure that jQuery is also selected in the subsequent dialogs.
4. Under **Window > Preferences**, you can now go to the category Aptana Studio and open the subcategory **Editors** and then **JavaScript**. Here, you can select the code completion for jQuery under Code Assist. It is then available in all further default web projects. In other words, you need to complete these three steps only once, not for each and every project.

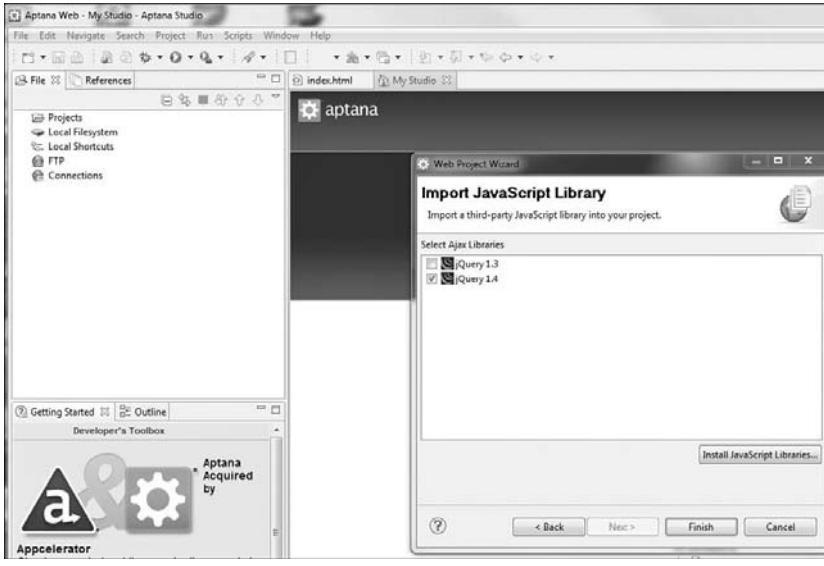


Figure 1.9 Enabling the code completion.

> **Note**
>
> The specific editor or IDE is not relevant for this book, although I am working mainly with Aptana.

As mentioned elsewhere, the big players in development also integrate jQuery into their tools; for example, Microsoft integrates it in Visual Studio from version 2008 onward. For creating web applications, the Web Developer integrated into Visual Studio is of particular interest. If your background involves ASP.NET, Visual Studio, which is also available as a free Express version (for example, as version 2012 under http://www.microsoft.com/visualstudio/eng/products/visual-studio-express-products), is an excellent development environment.

You can also expand the web browser Firefox with numerous extensions (add-ons) so that it offers many useful features for web development (for example, the DOM Inspector, Firebug, Live HTTP headers, and Web Developer). After installation, you can find the add-ons in the Extras menu. The simplest way of installing a Firefox extension is to go to the download page of the Mozilla project (https://addons.mozilla.org/), type the name of the add-on you want into the search field, and then click the installation hyperlink for your browser.

1.5 About the Author

To conclude this chapter, I give you a bit of information about myself. You will already know my name from the cover of this book or from reading the preface, but for the sake of politeness, allow me to introduce myself once more: My name is Ralph Steyer. I went to university in Frankfurt/Main (Germany) and studied mathematics (Diploma). I then spent several years working as a programmer and conceptual project member for a large insurance company in the Rhine-Main area, first with Turbo Pascal, later with C and C++. After 4 years, I spent 1 year working in database conception for a mainframe database under MVS. This experience was a great motivation for my step toward being self-employed because I realized that I did not want to do this long term.

Since 1995, I have been earning my living as a freelancer, switching on-the-fly between working as technical author, specialized journalist, IT lecturer, consultant, and programmer. In addition to these roles, I sometimes give lectures on web conferences, teach at various academies and one university, occasionally translate specialist books, and record online training videos. In my opinion, this makes quite a good mixture, preserves me from professional apathy, and keeps me close to the practice and at the forefront of development. In particular, I have the pleasure but also the burden of having to constantly stay current with new IT developments because the half-life of computer knowledge is rather short. Correspondingly, my job is sometimes tiring, but always exciting.

Summary

In this introductory chapter, you have found out who will guide you through this book, what this book is about, and who it is aimed at (in addition to its underlying structure). You now know what requirements you need to meet to start creating RIAs based on jQuery. And that is what we do next.

2

First Examples with jQuery

In this chapter, we make first contact with jQuery without any further preparations. In other words, we are jumping right into the deep end. I am anxious for you to get a feeling for what you can do with jQuery and what you can get out of this framework. Just accept for now that many questions regarding the source text have to remain open at this stage. Don't worry, though; these questions are answered over the next few chapters. The explanations on the listings also remain somewhat superficial at this stage, to avoid going off topic. We want to get into the practical application of jQuery as quickly as possible and just have some fun playing around, which means creating examples.

> **Note**
>
> For the examples in this chapter, but also most examples in the following chapters, it is not relevant which specific version of jQuery you are using. The examples in this book have been created with jQuery 1.8.2 or later, but often any version from 1.3 or at least 1.4.1 onward is sufficient.

2.1 Accessing Elements and Protecting the DOM

If you already have some basic knowledge of programming on the Web,[1] you already know that you can access the components of a web page via JavaScript or another script language in the browser via an object model with the name Document Object Model (DOM). For this type of access, there are several standard techniques,[2] each of which has its own weaknesses.

1. Given the target audience of this book, I assume you do.

2. For example, the methods `getElementById()` and `getElementsByTagName()` plus access via object fields or names.

In particular, you usually have to enter many characters when accessing just a single element of the web page (or a group). This involves a lot of effort and is susceptible to errors. Most frameworks therefore offer a system via which this access can take place with an abbreviated, unified approach. Plus the underlying mechanisms compensate for various weaknesses of the standard access methods, above all by compensating for browser-dependent particularities and supplementing various missing functions of the pure DOM concept. Particularly important is that this compensation has generally been tested on all officially supported browsers and therefore works rather reliably.

The following example demonstrates another extremely important function of jQuery—protecting the DOM. More on what this is all about later. For now, let's just say that different browsers process the web page differently on loading (parsing) the page, which can lead to a number of problems when the elements of the web page are accessed (especially if you try to access the elements of the web page too soon in a script—in other words, before the browser has correctly constructed the DOM). Here, jQuery offers a reliably method for mastering these problems.

The example also shows you in passing, as it were, how you can use jQuery as a standardized way of accessing contents of elements with text and reacting to events. But enough introduction. Here is our very first listing (ch2_1.html):[3]

Listing 2.1 **The First jQuery Example**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
   <meta http-equiv="Content-Type"
     content="text/html; charset=utf-8" />
   <title>The first jQuery example</title>
   <script type="text/javascript"
       src="lib/jquery-1.8.min.js"></script>
   <script type="text/javascript">
     $(document).ready(function(){
       $("#a").click(function(){
         $("#output").html("Boring :-(");
       });
       $("#b").click(function(){
         $("#output").html("A nice game :-)");
       });
       $("#c").click(function(){
```

3. The quotations are from the movie *War Games*—one of the first movies about a hacker. I highly recommend that you watch it the next time it is on TV.

```
18          $("#output").html("A strange game. " +
19               "The only winning move " +
20              "is not to play.");
21        });
22      });
23    </script>
24  </head>
25  <body>
26    <h1>Welcome to WOPR</h1>
27    <h3>Shall we play a game</h3>
28    <button id="a">Tic Tac Toe</button>
29    <button id="b">Chess</button>
30    <button id="c">
31      Worldwide Thermonuclear War</button>
32    <div id="output"></div>
33  </body>
34 </html>
```

Just create the HTML file in a separate directory and save it under the listed name.

In practice, you would usually save all your resources that are part of a project within a separate directory. For a web project, the best solution is to create these directories in the shared folder of your web server. In the case of Apache/XAMPP, this is usually the directory htdocs. This has the advantage that—if the web server is running—you can run the test directly via HTTP and a proper web call, not just load the file via the FILE protocol into the browser (in other words, the classic opening as file or simply dragging the file into the browser). The latter is not a realistic, practice-related test because later the pages also have to be requested by the visitor via a web server.

If you are working with an integrated development environment (IDE) such as Aptana or the Visual Studio Web Developer, you can usually display a web page directly from the IDE via an integrated web server. In Aptana, this is done via the **Run** command, and in Web Developer (a Firefox add-on) you can use the shortcut **Ctrl+F5.**

> **Note**
>
> In this book, all examples are sorted by chapter and listed accordingly on the companion website (http://jquery.safety-first-rock.de).

In lines 7 and 8, you see the reference to an external JavaScript file—the jQuery library that in this specific case resides in the subdirectory lib of the project directory where the website is saved. This structure has now become widely accepted in practice. This means that the jQuery library also has to be located in exactly that place. But, of course, you can instead choose to use a different path structure.

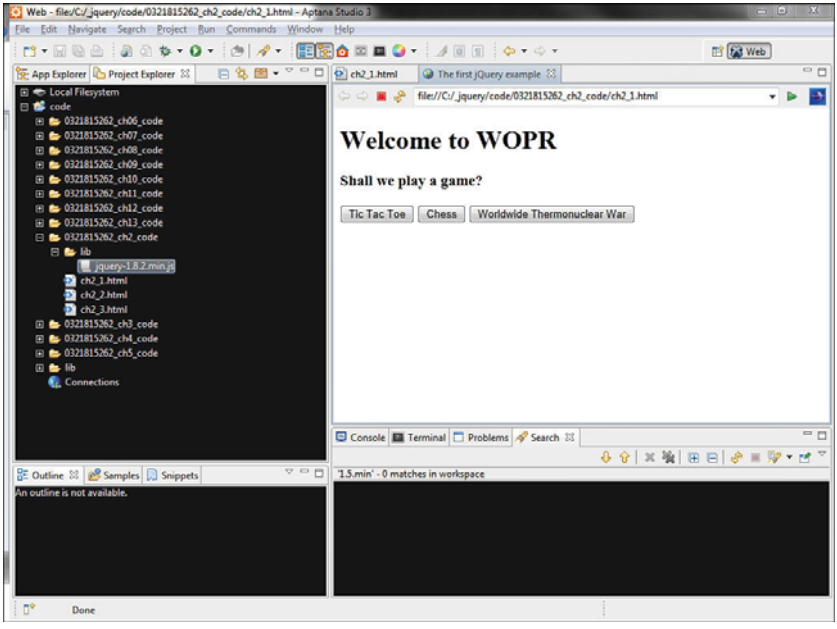


Figure 2.1 In this project, the jQuery library is located in the directory lib, seen from the perspective of the website.

Line 9 to 23 contains a normal JavaScript container. In it, the web page is addressed with `$(document)` (line 10). The function `$()` is a shorthand notation for referencing an element of the web page. You also see these shortened access notations in lines 11, 12, 14, 15, 17, and 18. But here, an element ID is used as a parameter.

> **Note**
>
> Note that an element (in terms of jQuery) as a parameter of `$()` is not enclosed in quotation marks, whereas an ID (or another selector) is enclosed in quotation marks.

Let's now take a quick look at the method `ready()` that starts in line 10 and goes up to line 22. This method ensures that the calls it contains are only executed when the web page has been fully loaded and the DOM is correctly constructed. As hinted at before and without going into too much detail, this is already a feature whose value cannot be appreciated highly enough.

> **Note**
>
> For readers with the corresponding knowledge and experience, the method `ready()` is an alternative for the event handler `onload` that you can write in HTML in the body of a web page or under JavaScript for the corresponding DOM object. But this event handler is seen as extremely *unreliable* because it is insufficiently implemented in various browsers. It is a good idea to avoid it wherever possible.

Within the `ready()` method, three event handlers each specify the reaction when clicking the listed elements. In our examples, these are three buttons marked with a unique ID.

> **Note**
>
> The method `click()` encapsulates (you probably guessed it) the function call of the event handler `onclick`.

The allocation to the correct function is achieved via the ID and triggering the function within the method `click()`. Note that we are using an anonymous function here (without an identifier).

It also gets interesting if a user clicks one of the buttons. This displays a specific text output in a section of the web page. We are again using `$()` and an ID for selecting the section (a `div` block) and the method `html()` for accessing the content.

Figure 2.2 The web page with the three buttons; the user has just clicked the third button.

> **Note**
>
> In all following examples, we omit writing or using the `DOCTYPE` statement. For the sake of completeness, it does belong there, but omitting it does not have any effect for us, and because it is always the same, writing it down over and over again is just a waste of space in this book. In the examples on the companion website, the statement is included because it forms part of the correct standard.

2.2 Editing the Web Page with DHTML à la jQuery

Generally, you can design the visual appearance of a web page much better and more effectively with style sheets than with pure HTML. In particular, they make it easier to separate layout and structure of the site. These statements are probably old hat to you, as true as they are.

If you now change the style sheets of a site dynamically via JavaScript, we are talking about Dynamic Hypertext Markup Language (DHTML). But animation effects such as showing and hiding parts of a web page via other JavaScript techniques also form part of this. In the following example, we look at how you can carry out animated web page changes with jQuery quickly, simply, conveniently, and yet reliably in the various browsers. In this example, we change the Cascading Style Sheets (CSS) class of an element dynamically.

First, let's look at a little CSS file that should be integrated into the following web page and saved in the lib directory (ch2_2.css):

Listing 2.2 **The File with the External Style Sheets**

```
01 body {
02  background: lightgray;color: blue;
03 }
04 div {
05  background: white;font-size: 14px;
06 }
07 .mClass {
08  background: red; color: yellow; font-size: 24px;
09 }
```

Nothing much happens in the CSS file. It determines the background and foreground color of the entire web page and all elements of the type `div`, plus the font size for all elements of the type `div`.

Of primary interest is the class described in lines 7–9. It is not yet to be used on loading the following web page, but is to be assigned dynamically in case of a user action (ch2_2.html):

Listing 2.3 **Changing the CSS Class**

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02  <head>
03  <meta http-equiv="Content-Type"
04    content="text/html; charset=utf-8" />
05  <title>The second jQuery example</title>
06  <link type="text/css" rel="stylesheet"
07    href="lib/ch2_2.css" />
08  <script type="text/javascript"
09    src="lib/jquery-1.8.2.min.js"></script>
```

```
10  <script type="text/javascript">
11    $(document).ready(function(){
12      $("#a").click(function(){
13        $("#c").addClass("mClass");
14      });
15      $("#b").click(function(){
16        $("#c").removeClass("mClass");
17      });
18    });
19  </script>
20  </style>
21 </head>
22 <body>
23  <h1>Editing Style Sheets with jQuery</h1>
24  <button id="a">Add CSS class</button>
25  <button id="b">Remove CSS class</button><hr/>
26  <div id="c">He who knows all the answers
27    has not been asked all the questions.
28  </div><hr/>
29  <div id="c">Be not afraid of going slowly,
30   be afraid only of standing still.</div>
31 </body>
32 </html>
```

In the example, you can see two buttons below a heading and two texts within a `div` section that is separated by a separator in each case. This is pure HTML. Plus in lines 6 and 7 you can see the link to the CSS file.

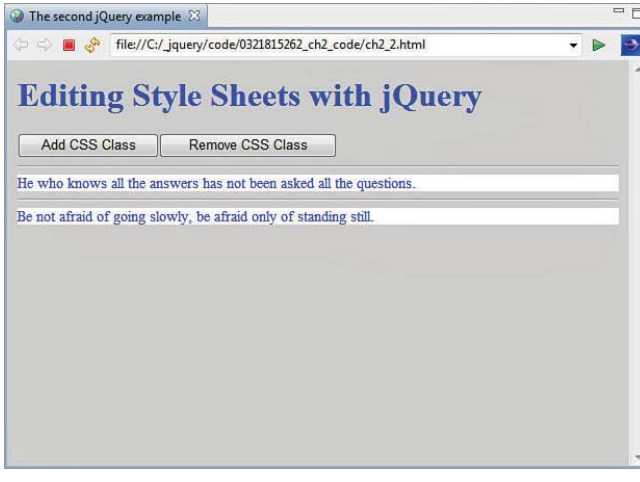


Figure 2.3 The site after loading

But now we want to use jQuery to manipulate the text below the buttons or the first `div` container. That is why the `div` container has an ID. The text below it is intended for comparison purposes.

For accessing the elements of the web page, the example uses jQuery mechanisms already mentioned in the first example. To react to the relevant click on a button, we again use the method `click()`. So far, there is nothing new.

Now you should notice that we do not yet assign the CSS class from the linked CSS file to an element on loading the web page. But take a look at line 13.

Listing 2.4 **Adding a CSS Class**

```
$("#c").addClass("mClass");
```

As the name of the method `addClass()` already implies, calling this method assigns the indicated style sheet class to the preceding element. This happens dynamically without the web page having to be reloaded in any way. The function is triggered when the user clicks the corresponding button, as you can see from the surrounding `click()` method.

Figure 2.4 The CSS class has been assigned.

In line 16, you can see how the class is removed again following the same pattern. This time, we use the method `removeClass()`. If you test the example, you will see that font and background are changed accordingly.

Tip

Alternatively, you could use the method `toggleClass()` in this example. This removes or adds a CSS class, always depending on the state. If the class is already assigned, it is then removed, and vice versa.

2.3 Animatedly Reducing and Enlarging of an Element

Now we want to use jQuery to animatedly reduce and enlarge an element to hide or show it. First, let's look at the external CSS file in the subdirectory lib, in which a property is defined that has specific consequences for the following animation (ch2_3.css):

Listing 2.5 **The CSS File**

```
01 body {
02  background: lightgray;color: blue;
03 }
04 #i1 {
05  width:300px; height:225px;
06 }
07 #i2 {
08  height:225px;
09 }
10 #h2{
11  background: white; color:#0000FF; font-size: 18px;
12 }
```

The specification that is interesting for the following example is the width data in line 5. The ID used as selector references an image. The width specification influences the type of the animation that follows. But first, let's look at the web page itself. It basically contains two images and some text below. We want to animate all three elements (ch2_3.html):

Listing 2.6 **Reducing or Enlarging Two Images and Some Text**

```
...
06    <link type="text/css" rel="stylesheet"
07      href="lib/ch2_3.css" />
08    <script type="text/javascript"
09      src="lib/jquery-1.8.min.js"></script>
10    <script type="text/javascript">
11      $(document).ready(function(){
12        $("#toggle1").click(function(event){
13          $('#i1').slideToggle('slow');
14        });
15        $("#toggle2").click(function(event){
16          $('#i2').slideToggle('slow');
17        });
```

```
18        $("#toggle3").click(function(event){
19          $('#h2').slideToggle('slow');
20        });
21      });
22    </script>
23  </head>
24  <body>
25    <h1>Animated showing and hiding
26   of an image and text with jQuery</h1>
27    <button id="toggle1">Toggle Image 1</button>
28    <button id="toggle2">Toggle Image 2</button>
29    <button id="toggle3">Toggle Text</button><hr/>
30    <img src="images/i1.jpg" id="i1" />
31    <img src="images/i2.jpg" id="i2" /><hr/>
32    <h2 id="h2">A ski jump</h2>
33  </body>
34 </html>
```

At the core of this animation is the method `slideToggle()`. This name is also very telling. You can use this effect to show or hide objects depending on the current state, or to reduce or enlarge them. In other words, the current state is toggled to the opposite state. You can see it applied in lines 13, 16, and 19.

Tip

As you can probably see, a temporal interval is specified as a parameter. It determines how long the animation should take. You can pass such parameters for the speed in most animations in jQuery. Permitted parameters are `slow`, `normal`, `fast`, or a specification of time in milliseconds. The specification in milliseconds is not enclosed by quotation marks.

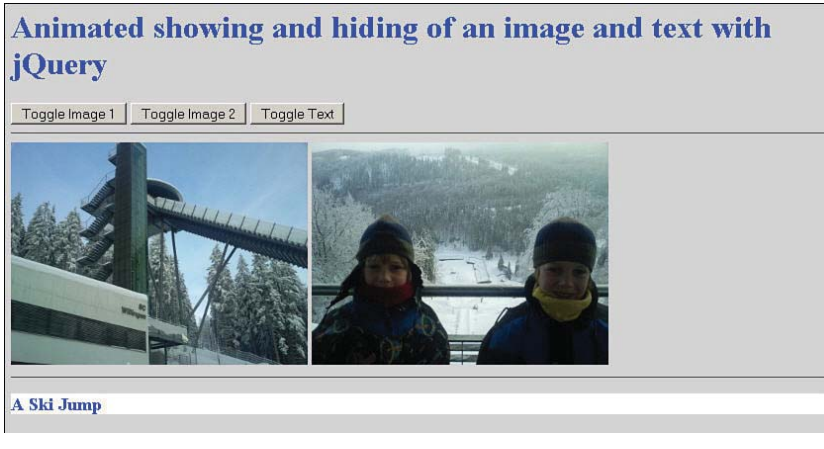


Figure 2.5 The original looks like this.

If you reconstruct the animation of the first image, you will see that reducing the image results in a reduction in image height and the image then disappears altogether. Vice versa, the image grows upward from that point if you enlarge it.

Figure 2.6 Here, the first image is squashed down.

Of massive importance for this behavior is that the width of this image is specified via the CSS rule for the ID `i1`. This prevents the width from also being reduced. The animation of the second image whose width is not specified shows what that looks like. You will see that on reducing the image, it shrinks into the lower-left corner of the image and then disappears altogether. Vice versa, the image grows outward from this point to the top right.

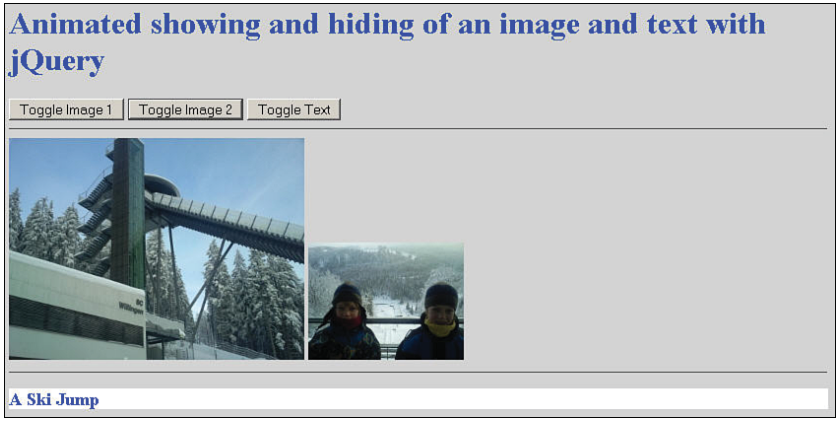


Figure 2.7 The second image shrinks animatedly in width and height.

But now observe what happens to the text if you click the third button. The heading also disappears but only in terms of height.

For the effect of `slideToggle()`, it matters to what type of element the animation technique is applied, and the CSS rules that have been previously applied to an element also play a role.

The animations in the example are basically independent from one another. If you set the interval for running the animation long enough, you can have the animations run in parallel.

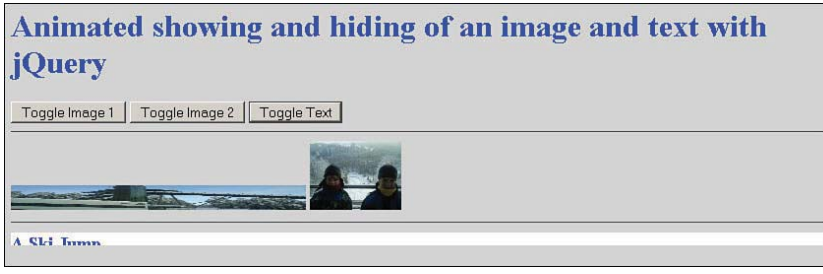


Figure 2.8 The three elements are animated in parallel.

In that case, note that the content positioned lower down is moved upward or could be repositioned vertically if a preceding element has disappeared completely. (In effect, it is removed from the text flow.)

But what happens in the example if you click the same button several times? This answer might surprise you: The events are cumulated. This means they are executed consecutively; the next event is executed only when the previous one has been fully processed. So, clicking the button again does not cause the current animation to stop and the next one to commence immediately. If that is what you want to achieve, you have to program it explicitly.

2.4 Changing Attributes Dynamically

This section shows how you can dynamically change attributes for an element of the web page. To this purpose, jQuery offers the extremely flexible and useful method `attr()`. With this, you can dynamically change one or several attributes of an element. In curly brackets, you set a value pair as parameter, first specifying the attribute, followed by a colon and then a string with the new value. Alternatively, you can also specify two string parameters. In this variation, the first parameter represents the attribute name and the second parameter the value. (In this case, you can change only one attribute.) If you only want to request the value of an attribute, you just enter the name of the attribute as a string parameter.

> **Note**
>
> For the sake of simplicity, we change only one attribute in the following example. If you want to change several attributes at once, however, you just need to write additional value pairs in the curly brackets, separated by commas.

For our example, we want to replace an image in the web page by changing the value of the attribute `src` of an `<img>` tag (ch2_4.html).

Listing 2.7 **Changing Attributes on an Element**

```
...
06    <script type="text/javascript"
07      src="lib/jquery-1.8.min.js"></script>
08    <script type="text/javascript">
09      $(document).ready(function(){
10        $("#toggle1").click(function(){
11          $("img").attr({
12            src: "images/i1.jpg"
13          });
14        });
15        $("#toggle2").click(function(){
16          $("img").attr(
17            "src", "images/i2.jpg"
18          );
19        });
20      });
21    </script>
22  </head>
23  <body>
24    <h1>Replacing an image</h1>
25    <button id="toggle1">Image 1</button>
26    <button id="toggle2">Image 2</button>
27    <hr/><img src="images/i1.jpg"/>
28 </html>
```

We change the value via the notation in the curly brackets once, and via the two string parameters once. As mentioned earlier, we replace the value `src` in each case.

Summary

This chapter provided just a few examples, but they already serve well to demonstrate central key facts of jQuery. In particular, you should memorize the function `$()` and the method `ready()`. But techniques for specifying reactions, such as the method `click()`, are also of elementary importance. And animation techniques such as `addClass()`, `toggleClass()`, `removeClass()`, and `slideToggle()` are also going to be helpful in practice later for DHTML effects. In this chapter, you also learned about changing attribute values (`attr()`). You will more fully understand these techniques as you work through other chapters of this book and have delved deeper into the overall concept of jQuery.

3

Basic Knowledge

In the preceding chapter, you worked through the first few examples with jQuery. Now it's time to turn to the underlying basics—not yet in the detail of jQuery itself, but in the world where jQuery is used and how the framework is anchored within it. This world is the Web with Hypertext Markup Language (HTML) or Extensible HTML (XHTML), Cascading Style Sheets (CSS), JavaScript, Extensible Markup Language (XML), JavaScript Object Notation (JSON), and Asynchronous JavaScript and XML (AJAX). After all, just as many other frameworks and toolkits in this area, jQuery is a JavaScript extension for websites that provides certain CSS features (in case of the jQuery UI, even custom CSS themes) and support for AJAX in addition to an independent syntax. Without elementary basic knowledge of the underlying technologies, you will hardly be able to use jQuery effectively and appropriately. In all our basic explanations, we focus, of course, on the jQuery aspect in this chapter as you learn how to integrate the jQuery library into your web page and what you need to look out for. Now let's get cracking so we can delve deeper into jQuery later.

> **Note**
>
> What this chapter cannot, should not, and does not do is to give you a complete introduction to the relevant techniques. Here you find only the basic information absolutely necessary in relation to jQuery. If you need to, use additional sources to find out more. The appendix contains more information about the most important basics of JavaScript as the core technology of jQuery.

3.1 The Web, Web 2.0, and the Client/Server Principle on the Internet

To start with, let's take a brief look at the World Wide Web (WWW); after all, you already know the basics facts about the WWW. At its core, the Web is based on the service protocol Hypertext Transfer Protocol (HTTP) for communication, the document description language HTML, and its strict XML-based twin XHTML plus the program types web server and web browser.

Just like all services available on the Internet, the Web is a classic client/server system, where each action consists of a cycle of requesting a service and providing the service. More specifically, this means that on the Web it is practically always a situation of a browser (the client) requesting a file (usually a web page or content that is to be integrated into a web page) and in some cases sending further requests for external resources referenced within this site (for example, graphics, videos, flash animations, or external JavaScript or CSS files). These are then displayed in the browser together with the web page or are otherwise processed. The central control mechanism is always the (X)HTML file.

3.1.1 Programming on the Web

Over the years, the Web has evolved into a system where you can program both on the server and on the client in terms of offering content. The hysteria about the dangers of client-side programming seems to have calmed down in recent years, and nearly all modern websites use client-side programming for that part of the business logic that should quite sensibly be taken care of in the client. After all these years, the only relevant representative of the client-side field that is still around is JavaScript, but by now it is used and accepted widely. Just take a look at some popular websites on the Internet. Not one of these sites can manage without JavaScript. And almost all users on the Web have JavaScript enabled in their browser. After all, who wants to do without the full and unlimited use of popular websites such as Google, Amazon, eBay, Facebook, Twitter, Wikipedia, or Yahoo! As a creator of websites, you can, in turn, assume that most clients have JavaScript plus its associated techniques.[1] This means that most arguments against using JavaScript frameworks such as jQuery or Dojo Toolkit, Prototype, YUI, and so on are no longer valid.

> **Note**
>
> As mentioned earlier in this book, various manufacturers are trying to establish proprietary techniques such as Silverlight in the client system to break through the limitations of JavaScript and web browsers as clients. But it will take some time before these are supported and accepted on a wide scale. It is unclear, at this point, whether these proprietary efforts will succeed.

1. According to the statistics I consulted, the support for JavaScript is fluctuating massively. Some statistics assume 99.9% availability, others perceive "only" 99.1% availability. I hope you are aware of the irony in my statementI to spell it out, you can currently assume that there is almost unlimited availability.

3.1.2 The Web 2.0

As ingenious and fundamentally simple as the concept of the Web is, there is a basic and serious problem with HTTP, HTML, and the concept of classic web browsers. When the browser requests new data from a web server, the latter always has to send a complete web page in response. Or to be more precise, the browser interprets the response in such a way that it completely replaces the site previously displayed in the browser with this new content. Obviously, this is very inefficient. This is where AJAX enters the stage. (You will read more about its technical background shortly.)

Generally, this is a procedure that ensures a reaction from a web application in (almost) real time, although new data is being requested from the web server. Instead of resulting in a complete web page with data that is in principle already present in the browser, an AJAX data request will result in only the really new data being sent by the web server and then using DHTML methods to "build it in to" the web page that is already loaded in the client. This does not usually even interrupt the normal user interaction with the web application by loading new data. Therefore, you can now create sites on the Web that are very much focused on interaction with the user. Thanks to Google, in particular, AJAX has now become established as standard procedure for such interactive websites. And since about 2005, the buzzword *Web 2.0* has been a collective term for most interactive websites. Often, Web 2.0 is also referred to as "participatory" or "interactive" web, because users are no longer just consumers but also contribute content themselves. Just think of blogs; tweets; wikis; or communities such as Xing, Facebook, MySpace, and so on. But even if users, for example, enter data in an online calendar, this will result in a different representation of the website (for example, the event is displayed—and, of course, also saved on the server). In this respect, that is also a form of participating in the Web 2.0.

3.2 JavaScript and Its Relationship to jQuery

Because jQuery is fundamentally a JavaScript library, we, of course, need to take a closer look at this language. Although marketing statements of jQuery and various other frameworks claim it will do many things with regard to JavaScript for you, to use jQuery you ought to have basic knowledge of JavaScript. And if you want to use jQuery really effectively (for example, to create plug-ins), you should even have good knowledge of JavaScript.

> **Note**
>
> The next few chapters lead you deeper into the world of JavaScript, where it is appropriate in connection with jQuery. For now, just a bit of basic information on JavaScript will suffice. For more basic concepts and information on JavaScript, see the appendix.

Generally, script languages on the Web are some of the most important extensions of HTML or XHTML and implement the client-side logic of a web application. These script languages are interpreter languages that are translated and executed within a host program (the browser) at runtime. This particularly applies to JavaScript.

> **Note**
>
> New browsers have a just-in-time compiler for JavaScript. This expands the interpreter principle for JavaScript with the option of keeping already translated code in the memory and making it available in a more performant way in case of a repeat execution. This is a key factor for Rich Internet Applications (RIAs) that are meant to behave in the same manner as desktop applications in terms of performance.

Essentially, jQuery is a JavaScript library. In other words, jQuery is based only on a function that every modern browser offers. You do not need a plug-in or another type of extension for the browser to have the functions in this library available. So, you are not adding it to the browser or even the operating system (as can be the case with competing technologies). This is, on the one hand, a great advantage, but on the other hand, you can only realize those things in the library that can be achieved with JavaScript or Dynamic HTML (DHTML) and Cascading Style Sheets plus Document Object Model (DOM) manipulation. But to be able to offer these exciting and powerful effects and services by jQuery using these simple basic technologies, they had to be stretched to their limits. As a consequence, not every browser can be fully supported (especially not older browsers).

If you are really good at programming with JavaScript, you could reproduce all the functionality of jQuery yourself. But that would involve quite some work and effort. The jQuery team has already invested many years of work in developing this library, and you can profit from this work for your own purposes.

3.2.1 The General Integration of JavaScript in Websites

JavaScript is to be seen as direct complement to and extension of HTML or XHTML and is intended for use as an integrated component of a corresponding website frame. Web scripts are directly written in plain text into a website or integrated and interpreted at runtime. Various techniques exist for connecting scripts to a website. Let's take a quick look at two of them.

The `<script>` Container in the Website

The connection of a JavaScript with a website can take place, for example, via a direct notation of the JavaScript in the website. The JavaScript statements are simply written as plain text into the corresponding (X)HTML file. The beginning of a script is marked via a separate control statement that is still part of HTML and forms with its matching end tag a container for the script instructions. Via the `<script>` element, you can specify that anything within the enclosed container is a script. So inside such a container, you are executing JavaScript.

Take a look at the following code snippet.

Listing 3.1 **A Code Snippet with the Direct Notation of JavaScript in a Website**

```
<body>
...
 <script>
```

```
 ... script statements
 </script>
...
</body>
```

If you do not specify a language in the script tag, all known browsers use JavaScript as default—or in the case of Internet Explorer, its clone JScript. But you are then exploiting the very high tolerance level of browsers because really you should specify the script language you are using. You can use the parameter `language` or `type` to indicate which script language this is.[2]

Here you can see two alternative examples.

Listing 3.2 **Alternative Specifications of Script Language**

```
<script language="JavaScript">
<script type="text/javascript">
```

Although uppercase and lowercase is irrelevant when specifying the value of `language`, you need to use lowercase for specifying the MIME type[3] via `type`. For `language`, you can also specify the version of JavaScript—for example, as `JavaScript1.3` or `JavaScript1.5`. You simply add the appropriate version number to the token `JavaScript`—not separated by a space. This causes any browsers that do not yet master this version to ignore the script block.

> **Caution**
>
> Avoid using `type` and `language` in parallel. If you specify the `type` as well as the `language`, the `language` data will be ignored. This means that any version you may want to specify via `language` has no effect.

In the past, we used to write HTML comments into the script container (`<!-- ... -->` or even `<!-- ... //-->`) to prevent browsers that do not know JavaScript from simply displaying the instructions. Today, this is completely unnecessary because such browsers last came on the market in around 1996 and are therefore no longer existent in practice. Note that this comment does not refer to browsers where JavaScript is simply disabled.

When integrating a `<script>` element into a website, the question arises where it should go. There is no clear answer to this question. Basically, the element belongs in the header of a website.

But in practice, you will find `<script>` elements in any place within a website. In principle, such an element can even stand after the website itself. The script statements are simply processed by a browser when the website is loaded into the browser and parsed from top

2. Officially, the specification of the MIME type is required.

3. A MIME type determines the type of content. It first specifies a main category such as `text` or `image` and then (separated by a slash) a subcategory such as `html`, `css`, or `javascript`.