SAMPLE CHAPTER

Extending jQuery

Keith Wood

FOREWORD BY Dave Methvin

MANNING

Extending jQuery

by Keith Wood

Chapter 1

brief contents

Part 1

Simple extensions

The most widely used JavaScript library on the web today, jQuery offers many functions to make life easy for front-end developers. You can make jQuery even better by extending it to provide additional functionality in a reusable format.

Chapter 1 contains a brief history of jQuery, then looks at what you can extend within jQuery. It finishes with a few examples of existing jQuery plugins, showing the breadth of possibilities.

In chapter 2 you'll find a description of the jQuery architecture and possible extension points, each of which is discussed in more detail. Then, to get you started, you'll see how to develop a simple plugin that you can use immediately.

The simplest extensions that you can create are enhanced selectors for jQuery—the building blocks behind finding the right element to operate upon. These are covered in chapter 3, with numerous examples of how to create your own.

1 jQuery extensions

This chapter covers

- jQuery's origins and purpose
- What you can extend in jQuery
- Examples of existing extensions

Today, jQuery is the most widely used JavaScript library on the web. It offers many functions to make life easier as a front-end developer, such as the ability to traverse the HTML Document Object Model (DOM) to find the elements you want to work with and apply animations to those elements. Moreover, the developers of jQuery have recognized that it can't (and shouldn't) do everything, and have provided extension points that allow additional functionality to be integrated into the normal jQuery processing. This foresight has contributed to its popularity.

In this book I explain how you can extend various aspects of jQuery to provide greater reuse and easier maintenance of your code. Alongside the standard plugin that operates on a collection of elements on a web page, you can create custom selectors, utility functions, custom animations, enhanced Ajax processors, custom events, and validation rules. I cover testing, packaging, and documenting your code to make sure that others can make maximum use of it as well.

1.1 jQuery background

The jQuery website defines jQuery as "a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers" (http://jquery.com).

It's a library of JavaScript functions that allows you to easily access the HTML DOM and inspect or update it, enabling you to provide more dynamic web pages and experiences in keeping with the Web 2.0 paradigm. Its main features are

- Element selection using a CSS-like syntax, with extensions
- Element traversal
- Element manipulation, including removal, content updates, and attribute changes
- Event handling, including custom events
- Effects and animations
- Ajax support
- A framework for extending its functionality (the subject of this book)
- Various utility functions
- Cross-browser support, including hiding differences between the browsers

jQuery is a freely available, open source library. It's currently licensed under the MIT License (http://jquery.org/license/). Previous versions were also licensed under the GNU General Public License, Version 2.

1.1.1 Origins

jQuery was initially developed by John Resig and was announced in January 2006, at BarCamp NYC.[1] He'd come across the Behaviour code written by Ben Nolan and saw the potential of its ideas—using pseudo-CSS style selectors to bind JavaScript functions to various elements in the DOM. But John wasn't happy with its verbosity and lack of hierarchical selectors.[2] His suggested syntax and subsequent implementation became the basis for jQuery.

Listing 1.1 shows Behaviour code to attach a click event handler to all `li` elements within an element with the ID `example`; the click event handler removes the clicked item. Listing 1.2 shows the now-familiar corresponding jQuery code.

Listing 1.1 Sample Behaviour code

```
Behaviour.register({
    '#example li': function(e){
        e.onclick = function(){
```

[1] John Resig, "BarCampNYC Wrap-up," http://ejohn.org/blog/barcampnyc-wrap-up/.
[2] John Resig, "Selectors in Javascript," http://ejohn.org/blog/selectors-in-javascript/.

```
            this.parentNode.removeChild(this);
        }
    }
});
```

Listing 1.2 Equivalent jQuery code

```
$('#example li').bind('click', function(){
    $(this).remove();
});
```

Why was it given the name jQuery? Originally, the library was called *jSelect* to reflect its ability to select elements within a web page. But when John checked for that name on the web, he found it was already taken, and changed the name to jQuery.[3]

1.1.2 Growth

Since its initial announcement, jQuery has been through numerous incremental releases, as shown in table 1.1 (not all versions are shown). Over the years, it's grown greatly in terms of functionality and size.

Table 1.1 jQuery versions (not all are shown)

Version	Code date	Size	Notes
1.0	August 26, 2006	44.3 KB	First stable release
1.0.4	December 12, 2006	52.2 KB	Last 1.0 bug fix
1.1	January 14, 2007	55.6 KB	Selector performance improvements
1.1.4	August 23, 2007	65.6 KB	jQuery may be renamed
1.2	September 10, 2007	77.4 KB	
1.2.6	May 26, 2008	97.8 KB	
1.3	January 13, 2009	114 KB	Sizzle selector engine introduced into core, live events, and events overhaul
1.3.2	February 19, 2009	117 KB	
1.4	January 13, 2010	154 KB	Performance improvements, Ajax enhancements
1.4.1	January 25, 2010	156 KB	`height()` and `width()` added, `parseJSON()` added
1.4.2	February 13, 2010	160 KB	`delegate()` added, performance improvements
1.4.3	October 14, 2010	176 KB	CSS module rewrite, metadata handling
1.4.4	November 11, 2010	178 KB	
1.5	January 31, 2011	207 KB	Deferred callback management, Ajax module rewrite, traversal performance
1.5.2	March 31, 2011	214 KB	

[3] Comments by John Resig, "BarCampNYC Wrap-up," http://ejohn.org/blog/barcampnyc-wrap-up/.

Table 1.1 jQuery versions (not all are shown) *(continued)*

Version	Code date	Size	Notes
1.6	May 2, 2011	227 KB	Significant performance improvements to the `attr()` and `val()` functions, `prop()` added
1.6.4	September 12, 2011	232 KB	
1.7	November 3, 2011	243 KB	New Event APIs: `on()` and `off()`, event delegation performance
1.7.2	March 21, 2012	246 KB	
1.8.0	August 9, 2012	253 KB	Sizzle rewritten, animations reimagined, more modularity
1.8.3	November 13, 2012	261 KB	
1.9.0	January 14, 2013	261 KB	Tidy up for jQuery 2.0
1.9.1	February 4, 2013	262 KB	Bug and regression fixes
2.0.0	April 18, 2013	234 KB	Drop support for IE 6-8
1.10.0	May 24, 2013	267 KB	Version/feature synchronization with 2.x line
1.10.2	July 3, 2013	266 KB	
2.0.3	July 3, 2013	236 KB	

Although the size of the jQuery library has grown substantially, when you minimize the code (stripping unnecessary comments and whitespace) it's reduced to about one-third of its source size (the latest version is only 91 KB). When that minified version is served from the web in a gzip format, it's further reduced to about a third again, resulting in a download cost of about 32 KB for the latest version. By using one of the CDNs (content delivery networks) available, that file may already be cached on the client, removing the need to download it at all.

Using CDNs

To download jQuery from one of the CDNs that hosts it, include one of the `script` tags shown in this sidebar. You may need to change the version of jQuery requested to suit your requirements.

Using jQuery's CDN provided by MediaTemple

```
<script src="http://code.jquery.com/jquery-1.9.1.min.js">
</script>
```

You can include the jQuery Migration plugin from this site too, to assist in the transition from older versions of jQuery to jQuery 1.9 and later.

```
<script src="http://code.jquery.com/
   jquery-migrate-1.1.1.min.js"></script>
```

(continued)

Using Google's CDN[a]

```
<script src="http://ajax.googleapis.com/ajax/libs/
   jquery/1.9.1/jquery.min.js"></script>
```

All jQuery releases are available on the Google CDN, but jQuery doesn't control this CDN and there may be a delay between a jQuery release and its availability there.

Using Microsoft's CDN[b]

```
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/
   jquery-1.9.1.min.js"></script>
```

All jQuery releases are available on the Microsoft CDN, but jQuery doesn't control this CDN and there may be a delay between a jQuery release and its availability there.

a. Google Developers, "Google Hosted Libraries—Developer's Guide," https://developers.google.com/speed/libraries/devguide#jquery.
b. ASP.NET, "Microsoft Ajax Content Delivery Network," http://www.asp.net/ajaxlibrary/cdn.ashx.

jQuery now includes the *Sizzle* selector engine, which enables the fundamental ability to find the elements within the DOM upon which you wish to operate. Whenever possible, Sizzle delegates these selectors to the underlying browser implementation, but resorts to JavaScript when necessary to ensure a common experience across all the major browsers.

1.1.3 Today

jQuery has become the most popular JavaScript library on the internet and has been adopted by many organizations and individuals for use in their websites. It's formally supported by Microsoft and ships as part of the Visual Studio product suite. BuiltWith reports more than 60% of the top 10,000 websites use jQuery, along with more than 50% of the top million.[4] W3Techs reports jQuery usage at 55% of all websites and 90% of those using any JavaScript library.[5]

The plugin developer community is thriving, and most make their code freely available in the spirit of the underlying jQuery library. You can search the web for appropriate modules, or use the newly revamped "official" repository of jQuery plugins (http://plugins.jquery.com). Some plugins are great, with solid code, good documentation, and examples. Others aren't so good, being hard to use, buggy, and/or poorly documented. Once you've read this book and applied its principles, your plugins should fall into the former category.

---

4 BuiltWith, "jQuery Usage Statistics," http://trends.builtwith.com/javascript/jQuery.
5 W3Techs, "Usage of JavaScript libraries for websites," http://w3techs.com/technologies/overview/javascript_library/all.

You can also find a lot of activity on the jQuery forums (https://forum.jquery.com), with more than 250,000 responses to more than 110,000 questions. Within the forums you'll find special sections devoted to using and developing jQuery plugins.

The ongoing development of jQuery is now managed by the jQuery Foundation (http://jquery.org). It was formed in September 2009 to look after all the jQuery projects, including jQuery Core, jQuery UI, jQuery Mobile, Sizzle, and QUnit. Contributions and donations by the jQuery community provide the financial basis for this support.

1.2 Extending jQuery

If jQuery offers so much functionality, why would you want to extend it? To keep the size of the jQuery code manageable, only those functions that are generic and widely used are included in the core code (although there's debate over what's used and useful). Basic element accessing and modification, event handling, animation, and Ajax handling are provided as functionality that most users require, whereas more specialized abilities are left to others to add.

Fortunately, the jQuery team has recognized that core jQuery can't do everything, so they've provided numerous integration points where others can extend the functionality of jQuery while benefitting from its existing infrastructure and abilities.

As well as extending jQuery to provide additional functionality, packaging your extension as a plugin allows you to easily reuse those abilities on other web pages. As a result you have only one copy of the code to maintain, and any improvements are immediately applied wherever they're used. You can test your plugin code in isolation and under controlled circumstances to ensure that it works as expected.

1.2.1 What can you extend?

Just as the core library provides many abilities, you'll find numerous ways to extend jQuery. The ones I'll cover in this book are listed in the next sections.

SELECTORS AND FILTERS

jQuery selectors and filters allow you to identify and collect the elements from the web page that you wish to operate upon. Although standard selectors by node name, ID, and class are built into jQuery, there's scope for adding pseudo-class selectors (extending the CSS-defined pseudo-classes) that allow you to filter a previous selection consistently and succinctly. You can also add set filters that are aware of the entire collection of previously selected elements and each one's position within that list. Chapter 3 explains how to create these selectors.

By creating a custom selector, you can consolidate the selection process into one location, making it easier to reuse that code elsewhere, ensuring a consistent implementation across your projects. It's also easier to maintain the selector and immediately apply any bug fixes or enhancements to all instances.

COLLECTION PLUGINS

Collection plugins are functions that you can apply to collections of elements as retrieved by a selector. These functions are what most people think of when the term

jQuery plugin is used, and they make up the largest portion of the available third-party plugins. The new abilities supplied by a collection plugin are only limited by your imagination and can range from making simple attribute changes, through behavioral changes from monitoring events on those elements, to completely replacing the original component with an alternate implementation.

Chapter 4 presents a series of guidelines to use when you create your plugin, and chapter 5 describes the plugin framework that I use for my plugins and how it implements those guidelines. The guidelines encapsulate best practice approaches to writing your plugin, helping it to integrate well with jQuery while reducing the possibility of external code interfering with it, or of it affecting other code.

A key component to writing your plugin is testing its functionality, and using a unit test tool enables you to easily and consistently run tests on your code, proving that it works as expected. Once your code is ready to release, it needs to be packaged for distribution so that others can obtain it easily and integrate it with their own project. You should also provide a web page that demonstrates your plugin's capabilities to allow prospective users to see how it works and what it can do. And you must supply documentation for every aspect of your plugin to let others get the most out of it. Chapter 7 covers these aspects of plugin development.

FUNCTION PLUGINS

Function plugins are utility functions that don't directly operate on collections of elements. They offer additional abilities within the jQuery framework and usually use jQuery's own functionality to perform their duties. Chapter 6 details how to add these utility functions.

Examples of these function plugins include support for sending debugging messages to a console for monitoring code execution, or for retrieving and setting cookie values for a web page. By making these abilities available as a jQuery plugin, you provide the user with a familiar way to invoke the code and reduce possible interference with external code. Several of the guidelines mentioned earlier still apply to these sorts of plugins, as do the steps of testing, packaging, demonstrating, and documenting the plugin.

JQUERY UI WIDGETS

jQuery UI "is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library" (http://jqueryui.com/). It defines a widget framework that allows you to create plugins that work in a consistent manner and that can take advantage of the numerous themes available for styling the UI. Chapter 8 looks at the widget framework and how you can use it to build your own component.

The jQuery UI widget framework also implements the plugin guidelines presented in chapter 4 and provides common functionality to all jQuery UI widgets in a consistent manner. By basing your plugin on this framework, you gain these built-in abilities automatically and can concentrate on delivering your widget's unique functionality. If you apply the classes defined in the ThemeRoller styling to your new widget, it'll

immediately be visually integrated with other jQuery UI components and will change appearance if you apply a new theme.

Several of the jQuery UI widgets rely on mouse drag actions to implement their functionality, and the jQuery UI team has recognized the importance of this interaction. By having your widget extend the jQuery UI Mouse module instead of the basic Widget one, you gain support for drag operations, complete with customizable conditions for starting a drag, and can again focus on implementing the functionality of your own widget. Chapter 9 describes how to use the Mouse module to create a widget that depends on using the mouse.

JQUERY UI EFFECTS

jQuery UI also provides a set of effects that may be applied to elements within your page. You can use many of these to show or hide an element, such as `blind`, `clip`, `fold`, and `slide`. Some bring your attention to an element, such as `highlight` and `pulsate`. You can define your own effect and apply it to elements as you would the standard ones. Chapter 10 shows how to create new UI effects.

ANIMATING PROPERTIES

jQuery provides an animation framework that you can apply to any element style attribute that has a simple numeric value. It allows you to vary that attribute from one value to another, controlling the duration of the change and the incremental steps along the way. But if the value you want to animate isn't a simple numeric value, you need to implement the functionality yourself. For example, jQuery UI provides a module that allows you to animate from one color to another. In chapter 11 we'll create an animator for a complex attribute value.

AJAX PROCESSING

jQuery's Ajax functionality is one of its clear benefits, making it incredibly easy to load remote data and then process it. As part of the Ajax call, you can identify what type of data is expected by the success callback: plain text, HTML, XML, JSON. A conversion process happens behind the scenes to transform the byte stream received by the remote call into the appropriate format. You can add your own transformations to allow you to produce specialized formats directly by identifying what type you want returned. Chapter 12 details how to extend the Ajax processing to handle a common file format directly.

EVENT HANDLING

jQuery's event handling capabilities allow you to attach multiple event handlers to elements to respond to user interactions, system events, and custom triggers. jQuery provides several hooks to let you create your own event definitions and trigger points, resulting in code that's consistent with the existing functionality. Chapter 13 describes the implementation of a new event to simplify interactions with the mouse.

VALIDATION RULES

The Validation plugin written by Jörn Zaefferer is widely used to validate user entry on the client side before submitting completed values to the server. Although the plugin

isn't part of the core jQuery functionality, it also provides extension points that allow you to create custom validation rules and have them applied as part of the existing processing. Chapter 14 illustrates how to create your own validation rules and integrate them with the built-in behavior.

1.3 Extension examples

Hundreds of jQuery plugins are available on the web to improve your web page experience. The numbers are a testament to the power and simplicity of jQuery itself, and the developers' foresight in providing the extensions points that allow it to be enhanced. I can't cover all of these plugins in this book, but the following sections offer a brief sampling to show the extent of the possibilities.

1.3.1 jQuery UI

The jQuery UI project (http://jqueryui.com/) is built on top of the core jQuery library as a collection of plugins. It encompasses several widgets, including Tabs, Datepicker, and Dialog (see figure 1.1), as well as various UI behaviors such as Draggable and Droppable. In addition, it provides several animations for use in showing or hiding elements, or in drawing your attention to them.

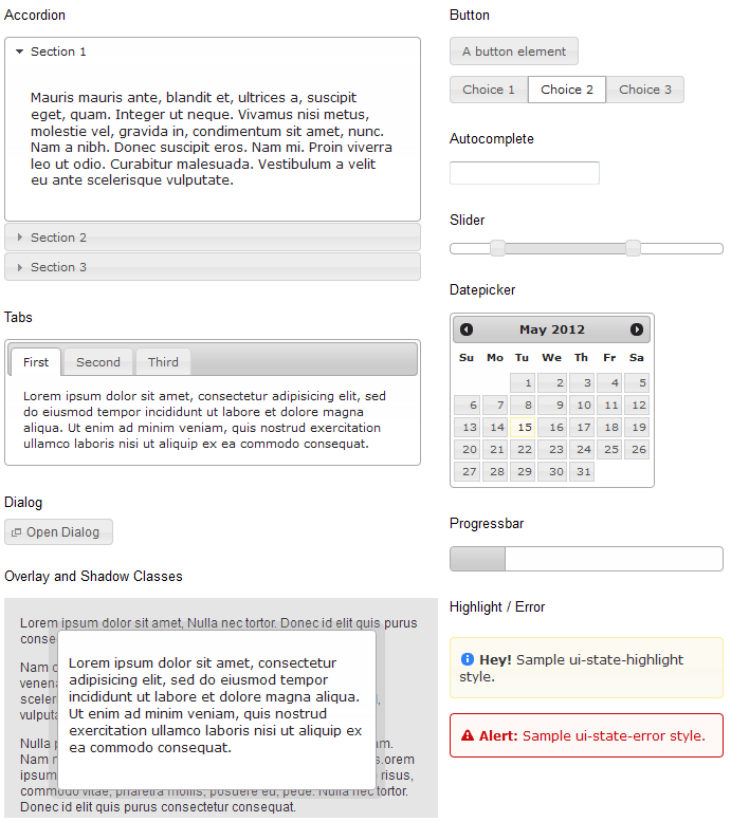

Figure 1.1 Sampler of jQuery UI widgets and styles

jQuery UI uses its own widget framework to provide a consistent base for its UI components. The framework manages widget creation and destruction, maintenance of state, and interactions with the mouse. Chapters 8 and 9 examine the widget framework and describe how to create your own widgets based on it.

The project integrates its components and behaviors with the ThemeRoller tool (http://jqueryui.com/themeroller/) to simplify generating a consistent theme that defines the appearance of all of its widgets.

Numerous demonstrations and comprehensive documentation accompany jQuery UI, allowing you to make the most of its abilities. Through the package's modular design, you can create a custom download that only includes the parts you need. Alternatively, you can load the package from one of the CDNs on which it's hosted, along with the standard themes.

1.3.2 Validation

As mentioned earlier, Jörn Zaefferer's Validation plugin[6] is widely used to provide client-side validation (see figure 1.2). It simplifies the assignment of validation rules to elements and manages their state and associated error messages. It aims to be unobtrusive—only generating an error when the form is submitted or a field is changed.

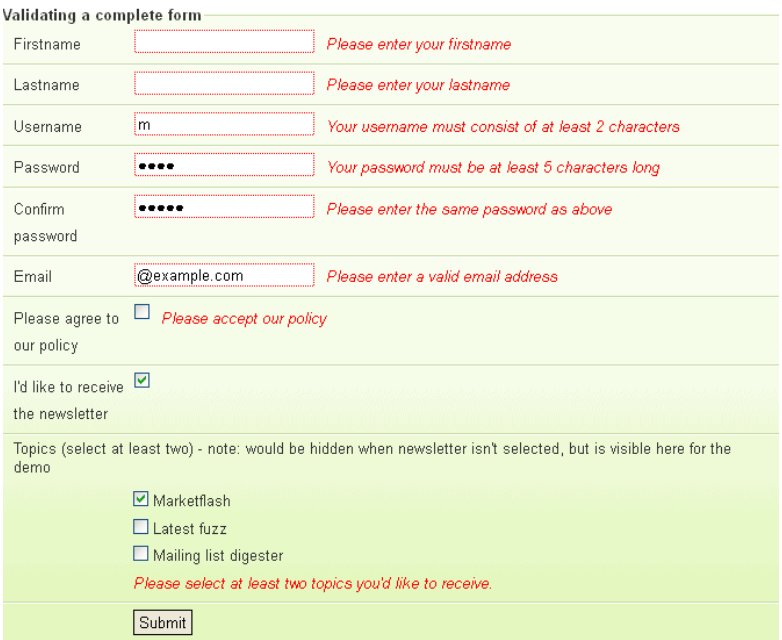

Figure 1.2 The Validation plugin in action, showing various error messages (in italics) resulting from validation issues, alongside the affected fields

[6] jQuery Validation Plugin, http://jqueryvalidation.org/.

Rules can be specified inline as attributes on each field, in code for named elements, or via a function chained to a jQuery selection. Numerous built-in validation rules are available, including `required`, `digits`, `date`, `email`, and `url`. Some validation rules can take additional parameters to modify their behavior, such as `minlength` and `maxlength`. Rules can be made dependent on the state of other elements on the page.

This plugin provides its own extension point, allowing you to define custom validation rules that you can then apply to the specified elements in the same manner as built-in ones. Chapter 14 describes how to write these rules.

Each rule has an associated error message for display to the user. These messages can be individually overridden, or can be translated into one of more than 30 other languages included in the package. You can control the positioning and grouping of error messages via options to the initialization call.

The plugin has extensive documentation and examples to assist you in its use. All told, it's a well-written and documented plugin, as well as a highly useful one.

1.3.3 Graphical slider

Plugins can enhance your web page by presenting content in a different and more appealing fashion. For example, the Nivo Slider plugin (http://nivo.dev7studios.com/) converts a simple list of images into a slideshow with various transitions between the pictures.

The eye-catching display shown in figure 1.3 is the result of applying the Nivo Slider to the HTML in listing 1.3. Although this is the default presentation, it's easy to generate and it looks good. As you'd expect, you'll find numerous options for customizing the plugin's appearance and behavior.

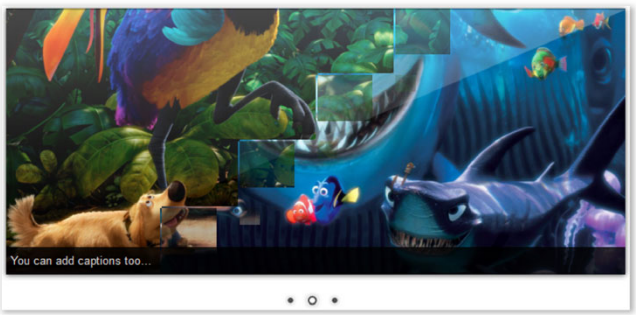

Figure 1.3 The Nivo Slider in action

Listing 1.3 Markup for a graphical slider

```
<div class="slider-wrapper">
    <div id="slider" class="nivoSlider">
        <img src="images/slide1.jpg" alt="" />
        <img src="images/slide2.jpg" alt=""
            title="You can add captions too..." />
        <img src="images/slide3.jpg" alt="" />
    </div>
</div>
```

Figure 1.4 Google Map integration with the gMap plugin

1.3.4 Google Maps integration

Some plugins wrap existing APIs to make them easier to access or to hide any cross-browser differences. The gMap plugin (http://gmap.nurtext.de/) is one such example, allowing you to integrate a Google Map into your web page. Although you could use Google Maps' own JavaScript API, plugins like this one encapsulate that functionality to provide a simpler interface.

The map shown in figure 1.4 results from the code in the following listing, demonstrating how easy the plugin is to use.

Listing 1.4 Adding a Google Map

```
$('#map').gMap({zoom: 4,
    markers: [{address: 'Brisbane, Australia',
        html: 'Brisbane, Australia', popup: true}]
});
```

1.3.5 Cookies

The jQuery Cookie plugin (https://github.com/carhartl/jquery-cookie) makes it easy to interact with the cookies associated with a web page. This plugin differs from previous examples in that its functionality doesn't apply to specific elements on the web page. Instead it offers a utility function that lets you work with cookies for the entire page.

Creating a cookie is as simple as providing its name and value:

```
$.cookie('introShown', true);
```

You can provide additional parameters to customize the cookie—setting its expiry period (by default, cookies expire at the end of the current session), the domain and path to which it applies, whether the cookie requires secure transmission, and whether the cookie value is encoded.

```
$.cookie('introShown', true, {expires: 30, path: '/'});
```

Retrieving a cookie value is only a matter of providing its name. If there's no cookie with a given name, a `null` is returned.

```
var introShown = $.cookie('introShown');
```

Delete a cookie by setting its value to `null`.

```
$.cookie('introShown', null);
```

The Cookie plugin is covered in detail in chapter 6.

1.3.6 Color animation

Basic jQuery includes animation abilities for element attributes that consist of a simple numeric value. Any other format for an attribute requires a special handler to be able to animate it correctly. As part of the Effects module in the jQuery UI project (http://jqueryui.com), you can animate colors (http://jqueryui.com/animate/), which may be set to a hexadecimal value (`#DDFFE8` or `#DFE`), an RGB triplet [`rgb(221, 255, 232)` or `rgb(86%, 100%, 91%)`], or a named color (`lime`).

After converting the various color formats into a common format, each component of the color (red/green/blue) is separately animated from its starting value to its finishing value. By providing this ability as an animation plugin, you can then use the standard jQuery functionality to apply it:

```
$('#myDiv').animate({backgroundColor: '#DDFFE8'});
$('#myDiv').animate({width: 200, backgroundColor: '#DFE'});
```

Chapter 11 covers animation plugins.

What you need to know

jQuery is the most widely used JavaScript library on the web.

jQuery provides basic and commonly used functionality, but is designed to be extended in many different ways.

There is a thriving third-party plugin community built around jQuery.

The abilities of a plugin are only limited by your imagination.

1.4 Summary

jQuery has grown to be the most widely used JavaScript library on the web today. Although it has a lot of built-in functionality, it concentrates on providing the basic

infrastructure and features used by many people across many websites. Recognizing that it can't provide everything for everyone, it includes numerous extension points where others can extend its behavior.

You can add functionality to nearly every part of jQuery, from defining custom selectors, through animating non-numeric attribute values and generating new events, to creating full-blown UI components. The only limit is your imagination.

Creating a plugin for your code lets you more easily reuse it in many of your web pages. It reduces your testing and maintenance burdens because you have only one copy of the script.

You'll see in the next chapter how easy it is to extend jQuery by creating a simple plugin, before delving deeper into the best-practice design of more complex plugins.

WEB DEVELOPMENT

Extending jQuery

Keith Wood

jQuery, the most popular JavaScript library, helps make client-side scripting of HTML easy. It offers many built-in abilities to traverse and alter the DOM, but it can't do everything. Fortunately, you can tap into jQuery's numerous extension points to create your own selectors and filters, plugins, animations, and more. This book shows you how.

Extending jQuery teaches you to build custom extensions to the jQuery library. In it, you'll discover how to write plugins and how to design them for maximum reuse. You'll also learn to write new widgets and effects for the jQuery UI. Along the way, you'll explore extensions in key areas including Ajax, events, animation, and validation.

What's Inside

- Create jQuery UI widgets and effects
- Make extensions available for distribution and reuse
- Build your own libraries

This book assumes intermediate-level knowledge of jQuery and JavaScript. No experience writing plugins or other extensions is required.

Keith Wood has developed over 20 jQuery plugins including the original Datepicker, World Calendar, Countdown, and SVG.

"Delves into just about every facet of extending jQuery's functionality."
—From the Foreword by Dave Methvin, President jQuery Foundation

"A must-read for all serious web developers."
—Ecil Teodoro, IBM

"Finally, a complete resource on the extension of jQuery and its framework."
—Daniele Midi Whitelemon Design Studio

"A well-written and technically excellent guide."
—Brady Kelly Erisia Web Development

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit manning.com/ExtendingjQuery

ISBN 13: 978-1-617291-03-6
ISBN 10: 1-617291-03-X
54499
9 781617 291036

MANNING $44.99 / Can $47.99 [INCLUDING eBOOK]