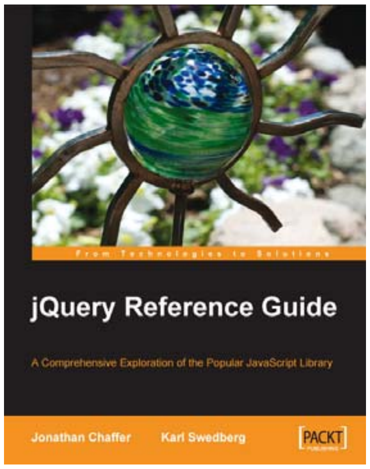
jQuery Reference Guide

Jonathan Chaffer
Karl Swedberg


jQuery Reference Guide
A Comprehensive Exploration of the Popular JavaScript Library
Jonathan Chaffer
Karl Swedberg

Chapter No. 1
"Anatomy of a jQuery Script"

In this package, you will find:

A Biography of the authors of the book

A preview chapter from the book, Chapter NO.1 "Anatomy of a jQuery Script"
A synopsis of the book’s content

Information on where to buy this book

About the Authors

Jonathan Chaffer is the Chief Technology Officer of Structure Interactive, an interactive agency located in Grand Rapids, Michigan. There he oversees web development projects using a wide range of technologies, and continues to collaborate on day-to-day programming tasks as well.

In the open-source community, Jonathan has been very active in the Drupal CMS project, which has adopted jQuery as its JavaScript framework of choice. He is the creator of the Content Construction Kit, a popular module for managing structured content on Drupal sites. He is responsible for major overhauls of Drupal’s menu system and developer API reference.

Jonathan lives in Grand Rapids with his wife, Jennifer.

I would like to thank Jenny, who thinks this is wonderful even if it bores her to tears. I’d also like to thank Karl for sharing my love for linguistics, producing a book that hopefully is grammatically immaculate enough to cover up any technical sins.

Karl Swedberg is a web developer at Structure Interactive in Grand Rapids, Michigan, where he spends much of his time implementing design with a focus on web standards—semantic HTML, well-mannered CSS, and unobtrusive JavaScript.

Before his current love affair with web development, Karl worked as a copy editor, a high-school English teacher, and a coffee house owner. His fascination with technology began in the early 1990s when he worked at Microsoft in Redmond, Washington, and it has continued unabated ever since.

Karl's other obsessions include photography, karate, English grammar, and fatherhood. He lives in Grand Rapids with his wife, Sara, and his two children, Benjamin and Lucia.

I wish to thank my wife, Sara, for her steadfast love and support during my far-flung adventures into esoteric nonsense. Thanks also to my two delightful children, Benjamin and Lucia. Jonathan Chaffer has my deepest respect and gratitude for his willingness to write this book with me and to explain the really difficult aspects of programming in a gentle manner when I just don't get it. Finally, I wish to thank John Resig for his brilliant JavaScript library and his ongoing encouragement for the book, as well as Rey Bango, Brandon Aaron, Klaus Hartl, Jörn Zaefferer, Dave Methvin, Mike Alsup, Yehuda Katz, Stefan Petre, Paul Bakaus, Michael Geary, Glen Lipka, and the many others who have provided help and inspiration along the way.

jQuery Reference Guide

jQuery is a powerful, yet easy-to-use JavaScript library that helps web developers and designers add dynamic, interactive elements to their sites, smoothing out browser inconsistencies and greatly reducing development time. In *jQuery Reference Guide*, you can investigate this library's features in a thorough, accessible format.

This book offers an organized menu of every jQuery method, function, and selector. Entries are accompanied by detailed descriptions and helpful recipes that will assist you in getting the most out of jQuery and avoiding the pitfalls commonly associated with JavaScript and other client-side languages. If you're still hungry for more, the book shows you how to cook up your own extensions with jQuery's elegant plug-in architecture.

You'll discover the untapped possibilities that jQuery makes available and hone your skills as you return to this guide time and again.

Demos of examples in this book are available at:

http:\\book.learningjquery.com.

What This Book Covers

In *Chapter 1* we'll begin by dissecting a working jQuery example. This script will serve as a roadmap for this book, directing you to the chapters containing more information on particular jQuery capabilities.

The heart of the book is a set of reference chapters that allow you to quickly look up the details of any jQuery method. *Chapter 2* lists every available selector for finding page elements.

Chapter 3 builds on the previous chapter with a catalog of jQuery methods for finding page elements.

Chapter 4 describes every opportunity for inspecting and modifying the HTML structure of a page.

Chapter 5 details each event that can be triggered and reacted to by jQuery.

Chapter 6 defines the range of animations built into jQuery, as well as the toolkit available for building your own.

Chapter 7 lists the ways in which jQuery can initiate and respond to server communication without refreshing the page.

Chapter 8 covers the remaining capabilities of the jQuery library that don't neatly fit into the other categories.

In the final three chapters, you'll dive into the extension mechanisms jQuery makes available. *Chapter 9* reveals four major ways to enhance jQuery's already robust capabilities using a plug-in.

Chapter 10 walks you through the advanced measurement tools available in the popular Dimensions plug-in.

Chapter 11 empowers you to bring AJAX technology and HTML forms together, a process which is made easy by the Form plug-in.

Appendix A provides a handful of informative websites on a wide range of topics related to jQuery, JavaScript, and web development in general.

Appendix B recommends a number of useful third-party programs and utilities for editing and debugging jQuery code within your personal development environment.

1
Anatomy of a jQuery Script

He's got a brand new start
Now he's a happy guy
– Devo,
"Happy Guy"

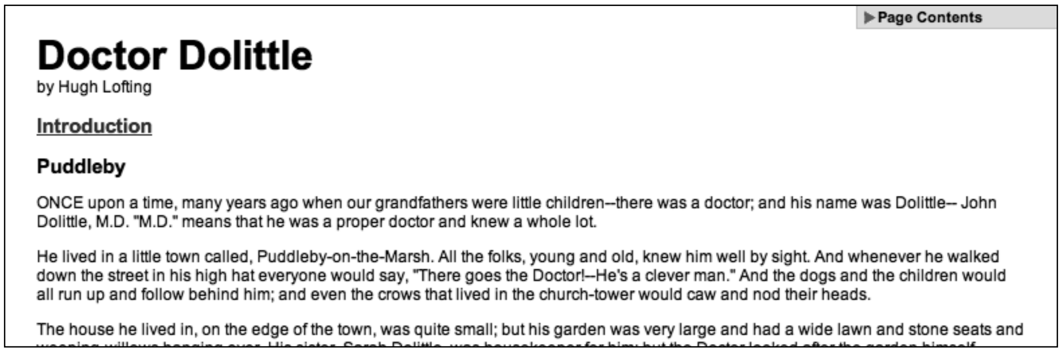
A typical jQuery script uses a wide assortment of the methods that the library offers. Selectors, DOM manipulation, event handling, and so forth come into play as required by the task at hand. In order to make the best use of jQuery, we need to keep in mind the wide range of capabilities it provides.

This book will itemize every method and function found in the jQuery library. Since there are many methods and functions to sort through, it will be useful to know what the basic categories of methods are, and how they come into play within a jQuery script. Here we will see a fully functioning script, and examine how the different aspects of jQuery are utilized in each part of the script.

A Dynamic Table of Contents

As an example of jQuery in action, we'll build a small script that will dynamically extract the headings from an HTML document and assemble them into a table of contents for that page.

Our table of contents will be nestled on the top right corner of the page:



▶ Page Contents

Doctor Dolittle

by Hugh Lofting

Introduction

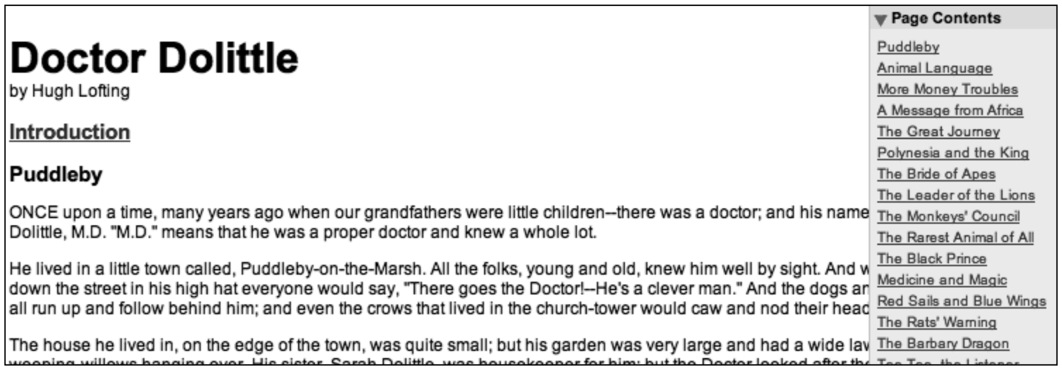
Puddleby

ONCE upon a time, many years ago when our grandfathers were little children--there was a doctor; and his name was Dolittle-- John Dolittle, M.D. "M.D." means that he was a proper doctor and knew a whole lot.

He lived in a little town called, Puddleby-on-the-Marsh. All the folks, young and old, knew him well by sight. And whenever he walked down the street in his high hat everyone would say, "There goes the Doctor!--He's a clever man." And the dogs and the children would all run up and follow behind him; and even the crows that lived in the church-tower would caw and nod their heads.

The house he lived in, on the edge of the town, was quite small; but his garden was very large and had a wide lawn and stone seats and

We'll have it collapsed initially as shown above, but a click will expand it to full height:



▼ Page Contents

Puddleby
Animal Language
More Money Troubles
A Message from Africa
The Great Journey
Polynesia and the King
The Bride of Apes
The Leader of the Lions
The Monkeys' Council
The Rarest Animal of All
The Black Prince
Medicine and Magic
Red Sails and Blue Wings
The Rats' Warning
The Barbary Dragon

Doctor Dolittle

by Hugh Lofting

Introduction

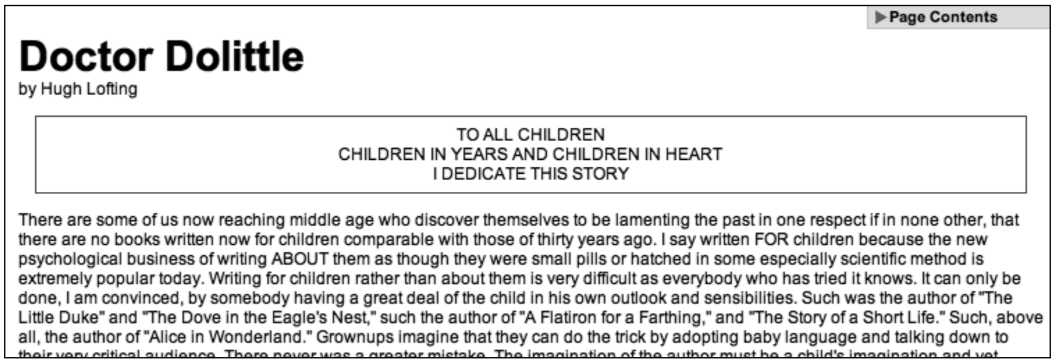
Puddleby

ONCE upon a time, many years ago when our grandfathers were little children--there was a doctor; and his name Dolittle, M.D. "M.D." means that he was a proper doctor and knew a whole lot.

He lived in a little town called, Puddleby-on-the-Marsh. All the folks, young and old, knew him well by sight. And w down the street in his high hat everyone would say, "There goes the Doctor!--He's a clever man." And the dogs ar all run up and follow behind him; and even the crows that lived in the church-tower would caw and nod their heac

The house he lived in, on the edge of the town, was quite small; but his garden was very large and had a wide lav

At the same time, we'll add a feature to the main body text. The introduction of the text on the page will not be initially loaded, but when the user clicks on the word **Introduction**, the introductory text will be inserted in place from another file:



Doctor Dolittle

by Hugh Lofting

TO ALL CHILDREN
CHILDREN IN YEARS AND CHILDREN IN HEART
I DEDICATE THIS STORY

There are some of us now reaching middle age who discover themselves to be lamenting the past in one respect if in none other, that there are no books written now for children comparable with those of thirty years ago. I say written FOR children because the new psychological business of writing ABOUT them as though they were small pills or hatched in some especially scientific method is extremely popular today. Writing for children rather than about them is very difficult as everybody who has tried it knows. It can only be done, I am convinced, by somebody having a great deal of the child in his own outlook and sensibilities. Such was the author of "The Little Duke" and "The Dove in the Eagle's Nest," such the author of "A Flatiron for a Farthing," and "The Story of a Short Life." Such, above all, the author of "Alice in Wonderland." Grownups imagine that they can do the trick by adopting baby language and talking down to their very critical audience. There never was a greater mistake. The imagination of the author must be a child's imagination and yet

Before we reveal the script that performs these tasks, we should walk through the environment in which the script resides.

Obtaining jQuery

The official jQuery website (`http://jquery.com/`) is always the most up-to-date resource for code and news related to the library. To get started, we need a copy of jQuery, which can be downloaded right from the home page of the site. Several versions of jQuery may be available at any given moment; the latest uncompressed version will be most appropriate for us.

No installation is required for jQuery. To use jQuery, we just need to place it on our site in a public location. Since JavaScript is an interpreted language, there is no compilation or build phase to worry about. Whenever we need a page to have jQuery available, we will simply refer to the file's location from the HTML document.

Setting Up the HTML Document

There are three sections to most examples of jQuery usage— the HTML document itself, CSS files to style it, and JavaScript files to act on it. For this example, we'll use a page containing the text of a book:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
```

```
      <meta http-equiv="Content-Type" content="text/html;
                                                charset=utf-8"/>
      <title>Doctor Dolittle</title>
       <link rel="stylesheet" href="dolittle.css" type="text/css" />
      <script src="jquery.js" type="text/javascript"></script>
      <script src="dolittle.js" type="text/javascript"></script>
    </head>
    <body>
      <div id="container">
        <h1>Doctor Dolittle</h1>
        <div class="author">by Hugh Lofting</div>
        <div id="introduction">
          <h2><a href="introduction.html">Introduction</a></h2>
        </div>
        <div id="content">
          <h2>Puddleby</h2>
          <p>ONCE upon a time, many years ago when our grandfathers
             were little children--there was a doctor; and his name was
             Dolittle-- John Dolittle, M.D.  &quot;M.D.&quot; means
             that he was a proper doctor and knew a whole lot.
         </p>

             <!-- More text follows... -->

        </div>
      </div>
    </body>
  </html>
```

The actual layout of files on the server does not matter. References from one file to another just need to be adjusted to match the organization we choose. In most examples in this book, we will use relative paths to reference files (`../images/foo.png`) rather than absolute paths (`/images/foo.png`). This will allow the code to run locally without the need for a web server.

The stylesheet is loaded immediately after the standard `<head>` elements. Here are the portions of the stylesheet that affect our dynamic elements:

```
/* -----------------------------------
   Page Table of Contents
-------------------------------------- */
#page-contents {
  position: absolute;
  text-align: left;
```

```
    top: 0;
    right: 0;
    width: 15em;
    border: 1px solid #ccc;
    border-top-width: 0;
    border-right-width: 0;
    background-color: #e3e3e3;
  }
  #page-contents h3 {
    margin: 0;
    padding: .25em .5em .25em 15px;
    background: url(arrow-right.gif) no-repeat 0 2px;
    font-size: 1.1em;
    cursor: pointer;
  }
  #page-contents h3.arrow-down {
    background-image: url(arrow-down.gif);
  }
  #page-contents a {
    display: block;
    font-size: 1em;
    margin: .4em 0;
    font-weight: normal;
  }
  #page-contents div {
    padding: .25em .5em .5em;
    display: none;
    background-color: #efefef;
  }

  /* -----------------------------------
     Introduction
  ------------------------------------- */
  .dedication {
    margin: 1em;
    text-align: center;
    border: 1px solid #555;
    padding: .5em;
  }
```

After the stylesheet is referenced, the JavaScript files are included. It is important that the script tag for the jQuery library be placed *before* the tag for our custom scripts; otherwise, the jQuery framework will not be available when our code attempts to reference it.

Writing the jQuery Code

Our custom code will go in the second, currently empty, JavaScript file which we included from the HTML using `<script src="dolittle.js" type="text/javascript"></script>`. Despite how much it accomplishes, the script is fairly short:

```
jQuery.fn.toggleNext = function() {
  this.toggleClass('arrow-down')
    .next().slideToggle('fast');
};

$(document).ready(function() {
  $('<div id="page-contents"></div>')
    .prepend('<h3>Page Contents</h3>')
    .append('<div></div>')
    .prependTo('body');

  $('#content h2').each(function(index) {
    var $chapterTitle = $(this);
    var chapterId = 'chapter-' + (index + 1);
    $chapterTitle.attr('id', chapterId);
    $('<a></a>').text($chapterTitle.text())
      .attr({
        'title': 'Jump to ' + $chapterTitle.text(),
        'href': '#' + chapterId
      })
      .appendTo('#page-contents div');
  });

  $('#page-contents h3').click(function() {
    $(this).toggleNext();
  });

  $('#introduction > h2 a').click(function() {
    $('#introduction').load(this.href);
    return false;
  });
});
```

We now have a dynamic table of contents that brings users to the relevant portion of the text, and an introduction that is loaded on demand.

Script Dissection

This script has been chosen specifically because it illustrates the widespread capabilities of the jQuery library. Now that we've seen the code as a whole, we can identify the categories of methods used therein.

> We will not discuss the operation of this script in much detail here, but a similar script is presented as a tutorial on the Learning jQuery web log: `http://www.learningjquery.com/2007/06/automatic-page-contents`.

Selector Expressions

Before we can act on an HTML document, we need to locate the relevant portions. In our script, we sometimes use a simple approach to finding an element:

```
$('#introduction')
```

This expression creates a new jQuery object that references the element with the ID `introduction`. On the other hand, sometimes we require a more intricate selector:

```
$('#introduction > h2 a')
```

Here we produce a jQuery object potentially referring to many elements. Elements are included if they are anchor tags, but only if they are descendants of `<h2>` elements that are themselves children of an element with the ID `introduction`.

These **selector expressions** can be as simple or complex as we need. Chapter 2 will enumerate all of the selectors available to us and how they can be combined.

DOM Traversal Methods

Sometimes we have a jQuery object that already references a set of DOM elements, but we need to perform an action on a different, related set of elements. In these cases, **DOM traversal** methods are useful. We can see this in part of our script:

```
this.toggleClass('arrow-down')
  .next()
  .slideToggle('fast');
```

Because of the context of this piece of code, the keyword `this` refers to a jQuery object (it often refers instead to a DOM element). In our case, this jQuery object is in turn pointing to the `<h3>` heading of the table of contents. The `.toggleClass` method call manipulates this heading element. The subsequent `.next()` operation changes the element we are working with, though, so that the following `.slideToggle` method call acts on the `<div>` containing the table of contents rather than its header. The methods that allow us to freely move about the DOM tree like this are listed in Chapter 3.

DOM Manipulation Methods

Finding elements is not enough; we want to be able to change them as well. Such changes can be as straightforward as changing a single attribute:

```
$chapterTitle.attr('id', chapterId);
```

Here we modify the ID of the matched element on the fly.

Sometimes the changes are further-reaching, on the other hand:

```
$('<div id="page-contents"></div>')
  .prepend('<h3>Page Contents</h3>')
  .append('<div></div>')
  .prependTo('body');
```

This part of the script illustrates that the **DOM manipulation** methods can not only alter elements in place, but also remove, shuffle, and insert them. These lines add a new heading at the beginning of `<div id="page-contents">`, insert another `<div>` container at the end of it, and place the whole thing at the beginning of the document body. Chapter 4 will detail these and many more ways to modify the DOM tree.

Event Methods

Even when we can modify the page at will, our pages will sit in place, unresponsive. We need **event methods** to react to user input, making our changes at the appropriate time:

```
$('#introduction > h2 a').click(function() {
  $('#introduction').load(this.href);
  return false;
});
```

In this snippet we register a handler that will execute each time the selected anchor tag is clicked. The click event is one of the most common ones observed, but there are many others; the jQuery methods that interact with them are discussed in Chapter 5.

Chapter 5 also discusses a very special event method, `.ready`:

```
$(document).ready(function() {
  // ...
});
```

This method allows us to register behavior that will occur immediately when the structure of the DOM is available to our code—even before the images have loaded.

Effect Methods

The event methods allow us to react to user input; the **effect methods** let us do this with style. Instead of immediately hiding and showing elements, we can do so with an animation:

```
this.toggleClass('arrow-down')
  .next()
  .slideToggle('fast');
```

This method performs a fast sliding transition on the element, alternately hiding and showing it with each invocation. The built-in effect methods are listed in Chapter 6, as is the way to create new ones.

AJAX Methods

Many modern websites employ techniques to load content when requested without a page refresh; jQuery allows us to accomplish this with ease. The **AJAX Methods** initiate these content requests and allow us to monitor their progress:

```
$('#introduction > h2 a').click(function() {
  $('#introduction').load(this.href);
  return false;
});
```

Here the `.load` method allows us to get another HTML document from the server and insert it in the current document, all with one line of code. This and more sophisticated mechanisms of retrieving information from the server are listed in Chapter 7.

Miscellaneous Methods

Some methods are harder to classify than others. The jQuery library incorporates several **miscellaneous methods** that serve as shorthand for common JavaScript idioms.

Even basic tasks like iteration are simplified by jQuery:

```
$('#content h2').each(function(index) {
  // ...
});
```

The `.each` method seen here steps through the matched elements in turn, performing the enclosed code on all of matched elements. In this case, the method helps us to collect all of the headings on the page so that we can assemble a complete table of contents. More helper functions such as this can be found in Chapter 8.

Plug-In API

We need not confine ourselves to built-in functionality either. The **plug-in API** that is part of jQuery allows us to augment the capabilities already present with new ones that suit our needs. Even in the small script we've written here, we've found the use for a plug-in:

```
jQuery.fn.toggleNext = function() {
  this.toggleClass('arrow-down')
    .next().slideToggle('fast');
};
```

This code defines a new `.toggleNext` jQuery method that slides the following element open and shut. We can now call our new method later when needed:

```
$('#page-contents h3').click(function() {
  $(this).toggleNext();
});
```

Whenever code could be reused outside the current script, it might do well as a plug-in. Chapter 9 will cover the plug-in API used to build these extensions.

Summary

We've now seen a complete, functional jQuery-powered script. This example, though small, brings a significant amount of interactivity and usability to the page. The script has illustrated the major types of tools offered by jQuery, as well. We've observed how the script finds items in the DOM and changes them as necessary. We've witnessed response to user action, and animation to give feedback to the user after the action. We've even seen how to pull information from the server without a page refresh, and how to teach jQuery brand new tricks in the form of plug-ins.

We'll be stepping through each function, method, and selector expression in the jQuery library now, chapter by chapter. In illustrating many of them, a customized logging function will aid our examples. This `.log` method prints text to the screen in a useful context; we'll dissect it as an example of a plug-in at the end of Chapter 9.

Each method will be introduced with a summary of its syntax and a list of its parameters and return value. Then we will offer a discussion, which will provide examples where applicable. For further reading about any method, consult the online resources listed in Appendix A.