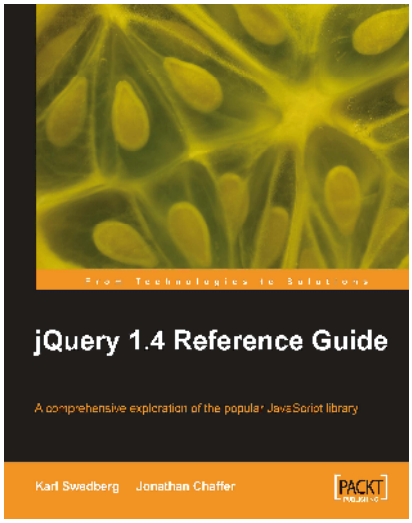
jQuery 1.4 Reference Guide

Karl Swedberg
Jonathan Chaffer



Chapter No.4
"DOM Manipulation Methods"

In this package, you will find:

A Biography of the authors of the book

A preview chapter from the book, Chapter NO.4 "DOM Manipulation Methods"

A synopsis of the book's content

Information on where to buy this book

About the Authors

Karl Swedberg is a web developer at Fusionary Media in Grand Rapids, Michigan, where he spends much of his time solving problems with JavaScript and implementing design. A member of the jQuery Project Team and an active contributor to the jQuery discussion list, Karl has presented at workshops and conferences, and provided corporate training in Europe and North America.

Before he got hooked on to web development, Karl worked as a copy editor, a high-school English teacher, and a coffee house owner. He gave up his dream of becoming a professional musician in the early 1990s about the same time that he stumbled into a job at Microsoft in Redmond, Washington. He sold his hollow-body Rickenbacker ages ago, but still keeps an acoustic guitar in the basement.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

I wish to thank my wife, Sara, for keeping me sane. Thanks also to my two delightful children, Benjamin and Lucia. Jonathan Chaffer has my deepest respect for his programming expertise and my gratitude for his willingness to write this book with me.

Many thanks to John Resig for creating the world's greatest JavaScript library and for fostering an amazing community around it. Thanks also to the folks at Packt Publishing, the technical reviewers of this book, the jQuery Cabal, and the many others who have provided help and inspiration along the way.

Jonathan Chaffer is a member of Rapid Development Group, a web development firm located in Grand Rapids, Michigan. His work there includes overseeing and implementing projects in a wide variety of technologies, with an emphasis on PHP, MySQL, and JavaScript.

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 for her tireless enthusiasm and support, Karl for the motivation to continue writing when the spirit was weak, and the Ars Technica community for constant inspiration toward technical excellence.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

jQuery 1.4 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 1.4 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.

What This Book Covers

In Chapter 1, *Anatomy of a jQuery Script*, 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, which allow you to quickly look up the details of any jQuery method. Chapter 2, *Selector Expressions*, lists every available selector for fi nding page elements.

Chapter 3, *DOM Traversal Methods*, builds on the previous chapter with a catalog of jQuery methods for finding page elements.

Chapter 4, *DOM Manipulation Methods*, describes every opportunity for inspecting and modifying the HTML structure of a page.

Chapter 5, *Event Methods*, details each event that can be triggered and reacted to by jQuery.

Chapter 6, *Effect Methods*, defi nes the range of animations built into jQuery, as well as the toolkit available for building your own.

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

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

Chapter 9, *jQuery Properties*, lists properties of the jQuery object that can be inspected for information about the browser environment.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

With the catalog of built-in functionality concluded, we'll dive into the extension mechanisms jQuery makes available. Chapter 10, *Plug-in API*, reveals these powerful ways to enhance jQuery's already robust capabilities using a plug-in.

Chapter 11, *Alphabetical Quick Reference*, offers a handy list of all methods and their arguments.

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

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

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

4

DOM Manipulation Methods

All of the methods in this chapter manipulate the DOM in some manner. A few of them simply change one of the attributes of an element, while others set an element's style properties. Still others modify entire elements (or groups of elements) themselves—inserting, copying, removing, and so on. All of these methods are referred to as **setters**, as they change the values of properties.

A few of these methods such as `.attr()`, `.html()`, and `.val()` also act as **getters**, retrieving information from DOM elements for later use.

General attributes

These methods get and set DOM attributes of elements.

.attr() (getter)

Get the value of an attribute for the first element in the set of matched elements.

```
.attr(attributeName)
```

Parameters

- `attributeName`: The name of the attribute to get

Return value

A string containing the attribute value.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

It's important to note that the `.attr()` method gets the attribute value for only the *first* element in the matched set. To get the value for each element individually, we need to rely on a looping construct such as jQuery's `.each()` method.

Using jQuery's `.attr()` method to get the value of an element's attribute has two main benefits:

- **Convenience**: It can be called directly on a jQuery object and chained to other jQuery methods.
- **Cross-browser consistency**: Some attributes have inconsistent naming from browser to browser. Furthermore, the values of some attributes are reported inconsistently across browsers, and even across versions of a single browser. The `.attr()` method reduces such inconsistencies.

.attr() (setter)

Set one or more attributes for the set of matched elements.

```
.attr(attributeName, value)
.attr(map)
.attr(attributeName, function)
```

Parameters (first version)

- `attributeName`: The name of the attribute to set
- `value`: A value to set for the attribute

Parameters (second version)

- `map`: A map of attribute-value pairs to set

Parameters (third version)

- `attributeName`: The name of the attribute to set
- `function`: A function returning the value to set

Return value

The jQuery object, for chaining purposes.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

The `.attr()` method is a convenient and powerful way to set the value of attributes, especially when setting multiple attributes or using values returned by a function. Let's consider the following image:

```
<img id="greatphoto" src="brush-seller.jpg"
  alt="brush seller" />
```

Setting a simple attribute

We can change the `alt` attribute by simply passing the name of the attribute and its new value to the `.attr()` method.

```
$('#greatphoto').attr('alt', 'Beijing Brush Seller');
```

We can *add* an attribute the same way.

```
$('#greatphoto')
  .attr('title', 'Photo by Kelly Clark');
```

Setting several attributes at once

To change the `alt` attribute and add the `title` attribute at the same time, we can pass both sets of names and values into the method at once using a map (JavaScript object literal). Each key-value pair in the map adds or modifies an attribute:

```
$('#greatphoto').attr({
  alt: 'Beijing Brush Seller',
  title: 'photo by Kelly Clark'
});
```

When setting multiple attributes, the quotation marks around attribute names are optional.

Computed attribute values

By using a function to set attributes, we can compute the value based on other properties of the element. For example, we could concatenate a new value with an existing value as follows:

```
$('#greatphoto').attr('title', function() {
  return this.alt + ' - photo by Kelly Clark'
});
```

This use of a function to compute attribute values can be particularly useful when we modify the attributes of multiple elements at once.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.removeAttr()

Remove an attribute from each element in the set of matched elements.

```
.removeAttr(attributeName)
.removeAttr(function)
```

Parameters (first version)

- `attributeName`: An attribute to remove

Parameters (second version)

- `function`: A function returning the attribute to remove

Return value

The jQuery object, for chaining purposes.

Description

The `.removeAttr()` method uses the JavaScript `removeAttribute()` function, but it has the advantage of being able to be called directly on a jQuery object and it accounts for different attribute naming across browsers.

As of jQuery 1.4, the `.removeAttr()` function allows us to indicate the attribute to be removed by passing in a function.

Style properties

These methods get and set CSS-related properties of elements.

.css() (getter)

Get the value of a style property for the first element in the set of matched elements.

```
.css(propertyName)
```

Parameters

- `propertyName`: A CSS property

Return value

A string containing the CSS property value.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

The `.css()` method is a convenient way to get a style property from the first matched element, especially in light of the different ways browsers access most of those properties (the `getComputedStyle()` method in standards-based browsers versus the `currentStyle` and `runtimeStyle` properties in Internet Explorer) and the different terms browsers use for certain properties. For example, Internet Explorer's DOM implementation refers to the `float` property as `styleFloat`, while W3C standards-compliant browsers refer to it as `cssFloat`. The `.css()` method accounts for such differences, producing the same result no matter which term we use. For example, an element that is floated left will return the `left` string for each of the following three lines:

- `$('div.left').css('float');`
- `$('div.left').css('cssFloat');`
- `$('div.left').css('styleFloat');`

Also, jQuery can equally interpret the CSS and DOM formatting of multiple-word properties. For example, jQuery understands and returns the correct value for both `.css('background-color')` and `.css('backgroundColor')`.

.css() (setter)

Set one or more CSS properties for the set of matched elements.

```
.css(propertyName, value)
.css(map)
.css(propertyName, function)
```

Parameters (first version)

- `propertyName`: A CSS property name
- `value`: A value to set for the property

Parameters (second version)

- `map`: A map of property-value pairs to set

Parameters (third version)

- `propertyName`: A CSS property name
- `function`: A function returning the value to set

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Return value

The jQuery object, for chaining purposes.

Description

As with the `.attr()` method, the `.css()` method makes setting properties of elements quick and easy. This method can take either a property name and value as separate parameters, or a single map of key-value pairs (JavaScript object notation).

Also, jQuery can equally interpret the CSS and DOM formatting of multiple-word properties. For example, jQuery understands and returns the correct value for both `.css({'background-color': '#ffe', 'border-left': '5px solid #ccc'})` and `.css({backgroundColor: '#ffe', borderLeft: '5px solid #ccc'})`. Notice that with the DOM notation, quotation marks around the property names are optional. However, with CSS notation they're required due to the hyphen in the name.

As with `.attr()`, `.css()` allows us to pass a function as the property value as follows:

```
$('div.example').css('width', function(index) {
  return index * 50;
});
```

This example sets the widths of the matched elements to incrementally larger values.

.height() (getter)

Get the current computed height for the first element in the set of matched elements.

```
.height()
```

Parameters

None

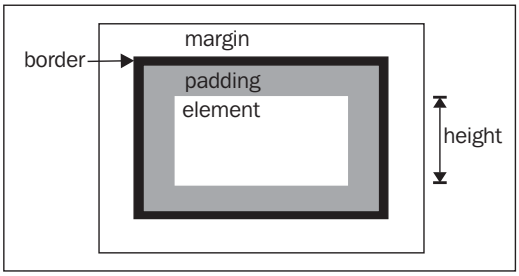
Return value

The height of the element in pixels

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

The difference between `.css('height')` and `.height()` is that the latter returns a unitless pixel value (for example, `400`), while the former returns a value with units intact (for example, `400px`). The `.height()` method is recommended when an element's height needs to be used in a mathematical calculation.



.height() (setter)

Set the CSS height of each element in the set of matched elements.

```
.height(value)
```

Parameters

- `value`: An integer representing the number of pixels, or an integer with an optional unit of measure appended

Return value

The jQuery object, for chaining purposes.

Description

When calling `.height(value)`, the value can be either a string (number and unit) or a number. If only a number is provided for the value, jQuery assumes a pixel unit. However, if a string is provided, any valid CSS measurement may be used for the height (such as `100px`, `50%`, or `auto`). Note that in modern browsers, the CSS height property does not include padding, border, or margin.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.innerHeight()

Get the current computed height for the first element in the set of matched elements, including padding but not border.

```
.innerHeight()
```

Parameters

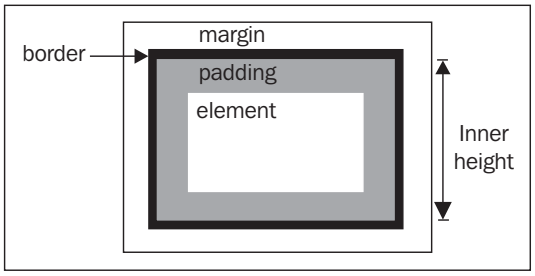
None

Return value

The height of the element in pixels, including top and bottom padding.

Description

This method is not applicable to `window` and `document` objects; for these use `.height()` instead.



.outerHeight()

Get the current computed height for the first element in the set of matched elements, including padding and border.

```
.outerHeight([includeMargin])
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Parameters

- `includeMargin`: A Boolean indicating whether to include the element's margin in the calculation

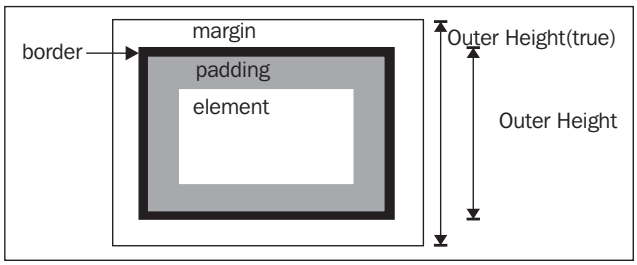
Return value

The height of the element, along with its top and bottom padding, border, and optionally margin, in pixels.

Description

If `includeMargin` is omitted or `false`, the padding and border are included in the calculation; if it's `true`, the margin is also included.

This method is not applicable to `window` and `document` objects, for these use `.height()` instead.



.width() (getter)

Get the current computed width for the first element in the set of matched elements.

```
.width()
```

Parameters

None

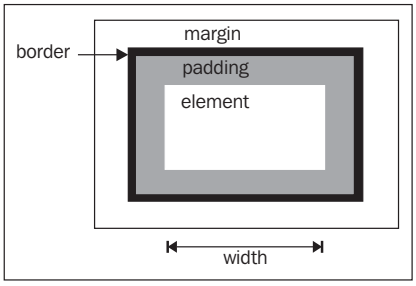
Return value

The width of the element in pixels.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

The difference between `.css(width)` and `.width()` is that the latter returns a unitless pixel value (for example, `400`), while the former returns a value with units intact (for example, `400px`). The `.width()` method is recommended when an element's width needs to be used in a mathematical calculation.



.width() (setter)

Set the CSS width of each element in the set of matched elements.

```
.width(value)
```

Parameters

- `value`: An integer representing the number of pixels, or an integer along with an optional unit of measure appended

Return value

The jQuery object, for chaining purposes.

Description

When calling `.width('value')`, the value can be either a string (number and unit) or a number. If only a number is provided for the value, jQuery assumes a pixel unit. However, if a string is provided, any valid CSS measurement may be used for the width (such as `100px`, `50%`, or `auto`). Note that in modern browsers, the CSS width property does not include padding, border, or margin.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.innerWidth()

Get the current computed width for the first element in the set of matched elements, including padding but not border.

```
.innerWidth()
```

Parameters

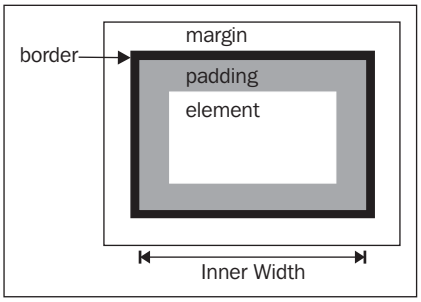
None

Return value

The width of the element, including left and right padding, in pixels.

Description

This method is not applicable to `window` and `document` objects, for these use `.width()` instead.



.outerWidth()

Get the current computed width for the first element in the set of matched elements, including padding and border.

```
.outerWidth([includeMargin])
```

Parameters

- `includeMargin`: A Boolean indicating whether to include the element's margin in the calculation

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

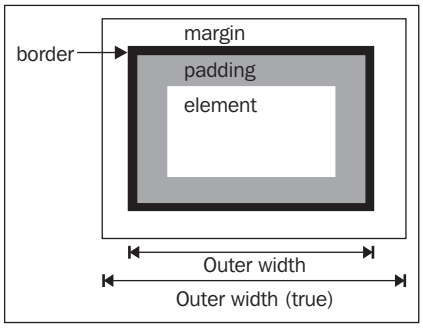
Return value

The width of the element, along with left and right padding, border, and optionally margin, in pixels.

Description

If `includeMargin` is omitted or `false`, the padding and border are included in the calculation; if it's `true`, the margin is also included.

This method is not applicable to `window` and `document` objects; for these use `.width()` instead.



.offset() (getter)

Get the current coordinates of the first element in the set of matched elements relative to the document.

```
.offset()
```

Parameters

None

Return value

An object containing the properties `top` and `left`.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

The `.offset()` method allows us to retrieve the current position of an element *relative to the document*. Contrast this with `.position()`, which retrieves the current position *relative to the offset parent*. When positioning a new element on top of an existing one for global manipulation (in particular, for implementing drag-and-drop), `.offset()` is more useful.

.offset() (setter)

Set the current coordinates of the first element in the set of matched elements relative to the document.

```
.offset(coordinates)
```

Parameters

- `coordinates`: An object containing the `top` and `left` properties, which are integers indicating the new top and left coordinates for the element

Return value

The jQuery object, for chaining purposes.

Description

The `.offset()` setter method allows us to reposition an element. The element's position is specified *relative to the document*. If the element's `position` style property is currently `static`, it will be set to `relative` to allow for this repositioning.

.position()

Get the current coordinates of the first element in the set of matched elements, relative to the offset parent.

```
.position()
```

Parameters

None

Return value

An object containing the properties `top` and `left`.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

The `.position()` method allows us to retrieve the current position of an element *relative to the offset parent*. Contrast this with `.offset()`, which retrieves the current position *relative to the document*. When positioning a new element near another one within the same DOM element, `.position()` is more useful.

.scrollTop() (getter)

Get the current vertical position of the scroll bar for the first element in the set of matched elements.

```
.scrollTop()
```

Parameters

None

Return value

The vertical scroll position in pixels.

Description

The vertical scroll position is the same as the number of pixels that are hidden from view above the scrollable area. If the scroll bar is at the very top, or if the element is not scrollable, this number will be `0`.

.scrollTop() (setter)

Set the current vertical position of the scroll bar for each of the sets of matched elements.

```
.scrollTop(value)
```

Parameters

- `value`: An integer indicating the new position to set the scroll bar to

Return value

The jQuery object, for chaining purposes.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.scrollLeft() (getter)

Get the current horizontal position of the scroll bar for the first element in the set of matched elements.

```
.scrollLeft()
```

Parameters

None

Return value

The horizontal scroll position in pixels.

Description

The horizontal scroll position is the same as the number of pixels that are hidden from view to the left of the scrollable area. If the scroll bar is at the very left, or if the element is not scrollable, this number will be `0`.

.scrollLeft() (setter)

Set the current horizontal position of the scroll bar for each of the set of matched elements.

```
.scrollLeft(value)
```

Parameters

- `value`: An integer indicating the new position to set the scroll bar to

Return value

The jQuery object, for chaining purposes

Class attributes

These methods inspect and manipulate the CSS classes assigned to elements.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.hasClass()

Determine whether any of the matched elements are assigned the given class.

```
.hasClass(className)
```

Parameters

- `className`: The class name to search for

Return value

A Boolean indicating whether the class is assigned to an element in the set.

Description

Elements may have more than one class assigned to them. In HTML, this is represented by separating the class names with a space:

```
<div id="mydiv" class="foo bar"></div>
```

The `.hasClass()` method will return `true` if the class is assigned to an element, and it will also return `true` if any other classes are assigned to it. For example, given the preceding HTML code, the following will return `true`:

- `$('#mydiv').hasClass('foo')`
- `$('#mydiv').hasClass('bar')`

.addClass()

Add one or more classes to each element in the set of matched elements.

```
.addClass(className)
.addClass(function)
```

Parameters (first version)

- `className`: One or more class names to be added to the class attribute of each matched element

Parameters (second version)

- `function`: A function returning one or more space-separated class names to be added

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Return value

The jQuery object, for chaining purposes.

Description

It's important to note that this method does not replace a class. It simply adds the class, appending it to any which may already be assigned to the elements.

More than one class may be added at a time, separated by a space, to the set of matched elements.

```
$('p').addClass('myClass yourClass');
```

This method is often used with `.removeClass()` to switch elements' classes from one to another.

```
$('p').removeClass('myClass noClass').addClass('yourClass');
```

Here, the `myClass` and `noClass` classes are removed from all paragraphs; while `yourClass` is added.

As of jQuery 1.4, the `.addClass()` method allows us to set the class name by passing in a function.

```
$('ul li:last').addClass(function() {
  return 'item-' + $(this).index();
});
```

Given an unordered list with five `<li>` elements, this example adds the `item-4` class to the last `<li>`.

.removeClass()

Remove one or all classes from each element in the set of matched elements.

```
.removeClass([className])
.removeClass([function])
```

Parameters (first version)

- `className` (optional): A class name to be removed from the class attribute of each matched element

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Parameters (second version)

- `function` (optional): A function returning one or more space-separated class names to be removed

Return value

The jQuery object, for chaining purposes.

Description

If a class name is included as a parameter, then only that class will be removed from the set of matched elements. If no class names are specified in the parameter, all classes will be removed.

More than one class may be removed at a time, separated by a space, from the set of matched elements.

```
$('p').removeClass('myClass yourClass')
```

This method is often used with `.addClass()` to switch elements' classes from one to another.

```
$('p').removeClass('myClass noClass').addClass('yourClass');
```

Here, the `myClass` and `noClass` classes are removed from all paragraphs; while `yourClass` is added.

To replace all existing classes with another class, we can use `.attr('class', 'newClass')` instead.

The `.removeClass()` method allows us to indicate the class to be removed by passing in a function.

```
$('li:last').removeClass(function() {
  return $(this).prev().attr('class');
});
```

This example removes the class name of the penultimate `<li>` from the last `<li>`.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.toggleClass()

Add or remove a class from each element in the set of matched elements, depending on either its presence or the value of the `addOrRemove` argument.

```
.toggleClass(className)
.toggleClass(className, addOrRemove)
.toggleClass(function[, addOrRemove])
```

Parameters (first version)

- `className`: A class name to be toggled in the class attribute of each element in the matched set

Parameters (second version)

- `className`: A class name to be toggled in the class attribute of each element in the matched set
- `addOrRemove`: A Boolean indicating whether to add or remove the class

Parameters (third version)

- `function`: A function that returns a class name to be toggled in the class attribute of each element in the matched set
- `addOrRemove` (optional): A Boolean indicating whether to add or remove the class

Return value

The jQuery object, for chaining purposes.

Description

This method takes one or more class names as its parameter. In the first version, if an element in the matched set of elements already has the class, then it is removed; if an element does not have the class, then it is added. For example, we can apply `.toggleClass()` to a simple `<div>` as follows:

```
<div class="tumble">Some text.</div>
```

The first time we apply `$('div.tumble').toggleClass('bounce')`, we get the following:

```
<div class="tumble bounce">Some text.</div>
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

The second time we apply `$('div.tumble').toggleClass('bounce')`, the `<div>` class is returned to the single `tumble` value as follows:

```
<div class="tumble">Some text.</div>
```

Applying `.toggleClass('bounce spin')` to the same `<div>` alternates between `<div class="tumble bounce spin">` and `<div class="tumble">`.

The second version of `.toggleClass()` uses the second parameter for determining whether the class should be added or removed. If this parameter's value is `true`, then the class is added; if `false`, the class removed. In essence, the statement:

```
$('#foo').toggleClass(className, addOrRemove);
```

is equivalent to

```
if (addOrRemove) {
  $('#foo').addClass(className);
}
else {
  $('#foo').removeClass(className);
}
```

As of jQuery 1.4, the `.toggleClass()` method allows us to indicate the class name to be toggled by passing in a function.

```
$('div.foo').toggleClass(function() {
  if ($(this).parent().is('.bar') {
    return 'happy';
  } else {
    return 'sad';
  }
});
```

This example will toggle the `happy` class for `<div class="foo">` elements if their parent element has a class of `bar`; otherwise, it will toggle the `sad` class.

DOM replacement

These methods are used to remove content from the DOM and replace it with new content.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.html() (getter)

Get the HTML contents of the first element in the set of matched elements.

```
.html()
```

Parameters

None

Return value

A string containing the HTML representation of the element.

Description

This method is not available on XML documents.

In an HTML document, we can use `.html()` to get the contents of any element. If our selector expression matches more than one element, only the first one's HTML content is returned. Consider the following code:

```
$('div.demo-container').html();
```

In order for the content of the following `<div>` to be retrieved, it would have to be the first one in the document.

```
<div class="demo-container">
  <div class="demo-box">Demonstration Box</div>
</div>
```

The result would look like this:

** Demonstration Box **

.html() (setter)

Set the HTML contents of each element in the set of matched elements.

```
.html(htmlString)
.html(function)
```

Parameters (first version)

- `htmlString`: A string of HTML to set as the content of each matched element

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Parameters (second version)

- `function`: A function returning the HTML content to set

Return value

The jQuery object, for chaining purposes.

Description

The `.html()` method is not available in XML documents.

When we use `.html()` to set the content of elements, any content that was in those elements is completely replaced by the new content. Consider the following HTML code:

```
<div class="demo-container">
  <div class="demo-box">Demonstration Box</div>
</div>
```

We can set the HTML contents of `<div class="demo-container">` as follows:

```
$('div.demo-container')
  .html('<p>All new content. <em>You bet!</em></p>');
```

That line of code will replace everything inside `<div class="demo-container">`.

```
<div class="demo-container">
  <p>All new content. <em>You bet!</em></p>
</div>
```

As of jQuery 1.4, the `.html()` method allows us to set the HTML content by passing in a function.

```
$('div.demo-container').html(function() {
  var emph = '<em>' + $('p').length + ' paragraphs!</em>';
  return '<p>All new content for ' + emph + '</p>';
});
```

Given a document with six paragraphs, this example will set the HTML of `<div class="demo-container">` to `<p>All new content for <em>6 paragraphs!</em></p>`.

.text() (getter)

Get the combined text contents of each element in the set of matched elements, including their descendants.

```
.text()
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Parameters

None

Return value

A string containing the combined text contents of the matched elements.

Description

Unlike the `.html()` method, `.text()` can be used in both XML and HTML documents. The result of the `.text()` method is a string containing the combined text of all matched elements. Consider the following HTML code:

```
<div class="demo-container">
  <div class="demo-box">Demonstration Box</div>
  <ul>
    <li>list item 1</li>
    <li>list <strong>item</strong> 2</li>
  </ul>
</div>
```

The code `$('div.demo-container').text()` would produce the following result:

Demonstration Box list item 1 list item 2

.text() (setter)

Set the content of each element in the set of matched elements to the specified text.

```
.text(textString)
.text(function)
```

Parameters (first version)

- `textString`: A string of text to set as the content of each matched element

Parameters (second version)

- `function`: A function returning the text to set as the content

Return value

The jQuery object, for chaining purposes.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

Unlike the `.html()` method, `.text()` can be used in both XML and HTML documents.

We need to be aware that this method escapes the string provided as necessary so that it will render correctly in HTML. To do so, it calls the DOM method `.createTextNode()`, which replaces special characters with their HTML entity equivalents (such as `<` for <). Consider the following HTML code:

```
<div class="demo-container">
  <div class="demo-box">Demonstration Box</div>
  <ul>
    <li>list item 1</li>
    <li>list <strong>item</strong> 2</li>
  </ul>
</div>
```

The code `$('div.demo-container').text('<p>This is a test.</p>');` will produce the following DOM output:

```
<div class="demo-container">
  &lt;p&gt;This is a test.&lt;/p&gt;
</div>
```

It will appear on a rendered page as though the tags were exposed as follows:

** This is a test **

As of jQuery 1.4, the `.text()` method allows us to set the text content by passing in a function.

```
$('ul li').text(function() {
  return 'item number ' + ($(this).index() + 1);
});
```

Given an unordered list with three `<li>` elements, this example will produce the following DOM output:

```
<ul>
  <li>item number 1</li>
  <li>item number 2</li>
  <li>item number 3</li>
</ul>
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.val() (getter)

Get the current value of the first element in the set of matched elements.

```
.val()
```

Parameters

None

Return value

A string containing the value of the element, or an array of strings if the element can have multiple values.

Description

The `.val()` method is primarily used to get the values of form elements. In the case of `<select multiple="multiple">` elements, the `.val()` method returns an array containing each selected option.

.val() (setter)

Set the value of each element in the set of matched elements.

```
.val(value)
.val(function)
```

Parameters (first version)

- `value`: A string of text or an array of strings to set as the value property of each matched element

Parameters (second version)

- `function`: A function returning the value to set

Return value

The jQuery object, for chaining purposes.

Description

This method is typically used to set the values of form fields. For `<select multiple="multiple">` elements, multiple `<option>` can be selected by passing in an array.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

The `.val()` method allows us to set the value by passing in a function.

```
$('input:text.items').val(function() {
  return this.value + ' ' + this.className;
});
```

This example appends the string "items" to the text inputs' values.

.replaceWith()

Replace each element in the set of matched elements with the provided new content.

```
.replaceWith(newContent)
```

Parameters

- `newContent`: The content to insert. This might be an HTML string, a DOM element, or a jQuery object.

Return value

The jQuery object, for chaining purposes. See the following section for details.

Description

The `.replaceWith()` method allows us to remove content from the DOM and insert new content in its place with a single call. Consider this DOM structure:

```
<div class="container">
  <div class="inner first">Hello</div>
  <div class="inner second">And</div>
  <div class="inner third">Goodbye</div>
</div>
```

We can replace the second inner `<div>` with specified HTML.

```
$('.second').replaceWith('<h2>New heading</h2>');
```

This results in the following structure:

```
<div class="container">
  <div class="inner first">Hello</div>
  <h2>New heading</h2>
  <div class="inner third">Goodbye</div>
</div>
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

We could equally target all inner `<div>` elements at once.

```
$('.inner').replaceWith('<h2>New heading</h2>');
```

This causes all of them to be replaced.

```
<div class="container">
  <h2>New heading</h2>
  <h2>New heading</h2>
  <h2>New heading</h2>
</div>
```

Alternatively, we could select an element to use as the replacement.

```
$('.third').replaceWith($('.first'));
```

This results in the following DOM structure:

```
<div class="container">
  <div class="inner second">And</div>
  <div class="inner first">Hello</div>
</div>
```

From this example, we can see that the selected element replaces the target by being moved from its old location, and not by being cloned.

The `.replaceWith()` method, like most jQuery methods, returns the jQuery object so that other methods can be chained onto it. However, it must be noted that the *original* jQuery object is returned. This object refers to the element that has been removed from the DOM, not the new element that has replaced it.

.replaceAll()

Replace each target element with the set of matched elements.

```
.replaceAll(target)
```

Parameters

- `target`: A selector expression indicating which element(s) to replace

Return value

The jQuery object, for chaining purposes.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

The `.replaceAll()` method is corollary to `.replaceWith()`, but with the source and target reversed. Consider the following DOM structure:

```
<div class="container">
  <div class="inner first">Hello</div>
  <div class="inner second">And</div>
  <div class="inner third">Goodbye</div>
</div>
```

We can create an element, and then replace other elements with it.

```
$('<h2>New heading</h2>').replaceAll('.inner');
```

This causes all of them to be replaced.

```
<div class="container">
  <h2>New heading</h2>
  <h2>New heading</h2>
  <h2>New heading</h2>
</div>
```

Alternatively, we could select an element to use as the replacement:

```
$('.first').replaceAll('.third');
```

This results in the following DOM structure:

```
<div class="container">
  <div class="inner second">And</div>
  <div class="inner first">Hello</div>
</div>
```

From this example, we can see that the selected element replaces the target by being moved from its old location, and not by being cloned.

DOM insertion, inside

These methods allow us to insert new content inside an existing element.

.prepend()

Insert content specified by the parameter at the beginning of each element in the set of matched elements.

```
.prepend(content)
.prepend(function)
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Parameters (first version)

- `content`: An element, an HTML string, or a jQuery object to insert at the beginning of each element in the set of matched elements

Parameters (second version)

- `function`: A function that returns an HTML string to insert at the beginning of each element in the set of matched elements

Return value

The jQuery object, for chaining purposes.

Description

The `.prepend()` and `.prependTo()` methods perform the same task. The major difference is in the syntax, specifically in the placement of the content and target. With `.prepend()`, the selector expression preceding the method is the container into which the content is inserted. With `.prependTo()`, on the other hand, the content precedes the method either as a selector expression or as markup created on the fly. It is then inserted into the target container.

Consider the following HTML code:

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

We can create content and insert it into several elements at once.

```
$('.inner').prepend('<p>Test</p>');
```

Each `<div class="inner">` element gets the following new content:

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">
    <p>Test</p>
    Hello
  </div>
  <div class="inner">
    <p>Test</p>
    Goodbye
  </div>
</div>
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

We can also select an element on the page and insert it into another:

```
$('.container').prepend($('h2'));
```

If an element selected this way is inserted elsewhere, it will be moved into the target (not cloned).

```
<div class="container">
  <h2>Greetings</h2>
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

However, if there are more than one target elements, cloned copies of the inserted elements will be created for each target after the first.

.prependTo()

Insert every element in the set of matched elements at the beginning of the target.

```
.prependTo(target)
```

Parameters

- `target`: A selector, element, HTML string, or jQuery object; the matched set of elements will be inserted at the beginning of the element(s) specified by this parameter

Return value

The jQuery object, for chaining purposes.

Description

The `.prepend()` and `.prependTo()` methods perform the same task. The major difference is in the syntax, specifically in the placement of the content and target. With `.prepend()`, the selector expression preceding the method is the container into which the content is inserted. With `.prependTo()`, on the other hand, the content precedes the method either as a selector expression or as markup created on the fly, and is inserted into the target container.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Consider the following HTML code:

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

We can create content and insert it into several elements at once.

```
$('<p>Test</p>').prependTo('.inner');
```

Each inner `<div>` element gets the following new content:

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">
    <p>Test</p>
    Hello
  </div>
  <div class="inner">
    <p>Test</p>
    Goodbye
  </div>
</div>
```

We can also select an element on the page and insert it into another.

```
$('h2').prependTo($('.container'));
```

If an element selected this way is inserted elsewhere, it will be moved into the target (not cloned).

```
<div class="container">
  <h2>Greetings</h2>
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

However, if there are more than one target elements, cloned copies of the inserted elements will be created for each target after the first.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.append()

Insert content specified by the parameter at the end of each element in the set of matched elements.

```
.append(content)
.append(function)
```

Parameters (first version)

- `content`: An element, an HTML string, or a jQuery object to insert at the end of each element in the set of matched elements

Parameters (second version)

- `function`: A function that returns an HTML string to insert at the end of each element in the set of matched elements

Return value

The jQuery object, for chaining purposes.

Description

The `.append()` and `.appendTo()` methods perform the same task. The major difference is in the syntax, specifically in the placement of the content and target. With `.append()`, the selector expression preceding the method is the container into which the content is inserted. With `.appendTo()`, on the other hand, the content precedes the method either as a selector expression or as markup created on the fly, and is inserted into the target container.

Consider the following HTML code:

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

We can create content and insert it into several elements at once.

```
$('.inner').append('<p>Test</p>');
```

Each inner `<div>` element gets the following new content:

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

```
    Hello
    <p>Test</p>
  </div>
  <div class="inner">
    Goodbye
    <p>Test</p>
  </div>
</div>
```

We can also select an element on the page and insert it into another.

```
$('.container').append($('h2'));
```

If an element selected this way is inserted elsewhere, it will be moved into the target (not cloned).

```
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
  <h2>Greetings</h2>
</div>
```

However, if there is more than one target element, cloned copies of the inserted elements will be created for each target after the first.

.appendTo()

Insert every element in the set of matched elements at the end of the target.

```
.appendTo(target)
```

Parameters

- `target`: A selector, element, HTML string, or jQuery object; the matched set of elements will be inserted at the end of the element(s) specified by this parameter

Return value

The jQuery object, for chaining purposes.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

The `.append()` and `.appendTo()` methods perform the same task. The major difference is in the syntax, specifically in the placement of the content and target. With `.append()`, the selector expression preceding the method is the container into which the content is inserted. With `.appendTo()`, on the other hand, the content precedes the method either as a selector expression or as markup created on the fly, and is inserted into the target container.

Consider the following HTML code:

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

We can create content and insert it into several elements at once.

```
$('<p>Test</p>').appendTo('.inner');
```

Each inner `<div>` element gets the following new content:

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">
    Hello
    <p>Test</p>
  </div>
  <div class="inner">
    Goodbye
    <p>Test</p>
  </div>
</div>
```

We can also select an element on the page and insert it into another.

```
$('h2').append($('.container'));
```

If an element selected this way is inserted elsewhere, it will be moved into the target (not cloned).

```
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
  <h2>Greetings</h2>
</div>
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

However, if there are more than one target elements, cloned copies of the inserted elements will be created for each target after the first.

DOM insertion, outside

These methods allow us to insert new content outside an existing element.

.before()

Insert content specified by the parameter before each element in the set of matched elements.

```
.before(content)
.before(function)
```

Parameters (first version)

- `content`: An element, an HTML string, or a jQuery object to insert before each element in the set of matched elements

Parameters (second version)

- `function`: A function that returns an HTML string to insert before each element in the set of matched elements

Return value

The jQuery object, for chaining purposes.

Description

The `.before()` and `.insertBefore()` methods perform the same task. The major difference is in the syntax, specifically in the placement of the content and target. With `.before()`, the selector expression preceding the method is the container before which the content is inserted. With `.insertBefore()`, on the other hand, the content precedes the method either as a selector expression or as markup created on the fly, and is inserted before the target container.

Consider the following HTML code:

```
<div class="container">
  <h2>Greetings</h2>
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

We can create content and insert it before several elements at once.

```
$('.inner').before('<p>Test</p>');
```

Each inner `<div>` element gets the following new content:

```
<div class="container">
  <h2>Greetings</h2>
  <p>Test</p>
  <div class="inner">Hello</div>
  <p>Test</p>
  <div class="inner">Goodbye</div>
</div>
```

We can also select an element on the page and insert it before another.

```
$('.container').before($('h2'));
```

If an element selected this way is inserted elsewhere, it will be moved before the target (not cloned).

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

However, if there is more than one target element, cloned copies of the inserted elements will be created for each target after the first.

.insertBefore()

Insert every element in the set of matched elements before the target.

```
.insertBefore(target)
```

Parameters

- `target`: A selector, element, HTML string, or jQuery object; the matched set of elements will be inserted before the element(s) specified by this parameter.

Return value

The jQuery object, for chaining purposes.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

The `.before()` and `.insertBefore()` methods perform the same task. The major difference is in the syntax, specifically in the placement of the content and target. With `.before()`, the selector expression preceding the method is the container before which the content is inserted. With `.insertBefore()`, on the other hand, the content precedes the method either as a selector expression or as markup created on the fly, and is inserted before the target container.

Consider the following HTML code:

```
<div class="container">
  <h2>Greetings</h2>
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

We can create content and insert it before several elements at once.

```
$('<p>Test</p>').insertBefore('.inner');
```

Each inner `<div>` element gets the following new content:

```
<div class="container">
  <h2>Greetings</h2>
  <p>Test</p>
  <div class="inner">Hello</div>
  <p>Test</p>
  <div class="inner">Goodbye</div>
</div>
```

We can also select an element on the page and insert it before another.

```
$('h2').insertBefore($('.container'));
```

If an element selected this way is inserted elsewhere, it will be moved before the target (not cloned).

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

However if there are more than one target elements, cloned copies of the inserted elements will be created for each target after the first.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.after()

Insert content specified by the parameter after each element in the set of matched elements.

```
.after(content)
.after(function)
```

Parameters (first version)

- `content`: An element, HTML string, or jQuery object to insert after each element in the set of matched elements

Parameters (second version)

- `function`: A function that returns an HTML string to insert after each element in the set of matched elements

Return value

The jQuery object, for chaining purposes.

Description

The `.after()` and `.insertAfter()` methods perform the same task. The major difference is in the syntax, specifically in the placement of the content and target. With `.after()`, the selector expression preceding the method is the container after which the content is inserted. With `.insertAfter()`, on the other hand, the content precedes the method either as a selector expression or as markup created on the fly, and is inserted after the target container.

Consider the following HTML code:

```
<div class="container">
  <h2>Greetings</h2>
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

We can create content and insert it after several elements at once.

```
$('.inner').after('<p>Test</p>');
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Each inner `<div>` element gets the following new content:

```
<div class="container">
  <h2>Greetings</h2>
  <div class="inner">Hello</div>
  <p>Test</p>
  <div class="inner">Goodbye</div>
  <p>Test</p>
</div>
```

We can also select an element on the page and insert it after another.

```
$('.container').after($('h2'));
```

If an element selected this way is inserted elsewhere, it will be moved after the target (not cloned).

```
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
<h2>Greetings</h2>
```

However, if there are more than one target elements, cloned copies of the inserted element will be created for each target after the first.

.insertAfter()

Insert every element in the set of matched elements after the target.

```
.insertAfter(target)
```

Parameters

- `target`: A selector, element, HTML string, or jQuery object; the matched set of elements will be inserted after the element(s) specified by this parameter

Return value

The jQuery object, for chaining purposes.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Description

The `.after()` and `.insertAfter()` methods perform the same task. The major difference is in the syntax, specifically in the placement of the content and target. With `.after()`, the selector expression preceding the method is the container after which the content is inserted. With `.insertAfter()`, on the other hand, the content precedes the method either as a selector expression or as markup created on the fly, and is inserted after the target container.

Consider the following HTML code:

```
<div class="container">
  <h2>Greetings</h2>
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

We can create content and insert it after several elements at once.

```
$('<p>Test</p>').insertAfter('.inner');
```

Each inner `<div>` element gets the following content:

```
<div class="container">
  <h2>Greetings</h2>
  <div class="inner">Hello</div>
  <p>Test</p>
  <div class="inner">Goodbye</div>
  <p>Test</p>
</div>
```

We can also select an element on the page and insert it after another.

```
$('h2').insertAfter($('.container'));
```

If an element selected this way is inserted elsewhere, it will be moved after the target (not cloned).

```
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
<h2>Greetings</h2>
```

However, if there are more than one target elements, cloned copies of the inserted elements will be created for each target after the first.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

DOM insertion, around

These methods allow us to insert new content surrounding existing content.

.wrap()

Wrap an HTML structure around each element in the set of matched elements.

```
.wrap(wrappingElement)
.wrap(wrappingFunction)
```

Parameters (first version)

- `wrappingElement`: An HTML snippet, selector expression, jQuery object, or DOM element specifying the structure to wrap around the matched elements

Parameters (second version)

- `wrappingFunction`: A callback function that generates a structure to wrap around the matched elements

Return value

The jQuery object, for chaining purposes.

Description

The `.wrap()` function can take any string or object that could be passed to the `$()` factory function to specify a DOM structure. This structure may be nested several levels deep, but should contain only one inmost element. The structure will be wrapped around each of the elements in the set of matched elements.

Consider the following HTML code:

```
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

Using `.wrap()`, we can insert an HTML structure around the inner `<div>` elements as follows:

```
$('.inner').wrap('<div class="new" />');
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

The new `<div>` element is created on the fly and added to the DOM. The result is a new `<div>` wrapped around each matched element.

```
<div class="container">
  <div class="new">
    <div class="inner">Hello</div>
  </div>
  <div class="new">
    <div class="inner">Goodbye</div>
  </div>
</div>
```

The second version of this method allows us to specify a callback function instead. This callback function will be called once for every matched element. It should return a DOM element, a jQuery object, or an HTML snippet in which to wrap the corresponding element. For example:

```
$('.inner').wrap(function() {
  return '<div class="' + $(this).text() + '" />';
});
```

This will cause each `<div>` to have a class corresponding to the text it wraps.

```
<div class="container">
  <div class="Hello">
    <div class="inner">Hello</div>
  </div>
  <div class="Goodbye">
    <div class="inner">Goodbye</div>
  </div>
</div>
```

.wrapAll()

Wrap an HTML structure around all elements in the set of matched elements.

```
.wrapAll(wrappingElement)
```

Parameters

- `wrappingElement`: An HTML snippet, selector expression, jQuery object, or DOM element specifying the structure to wrap around the matched elements

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Return value

The jQuery object, for chaining purposes.

Description

The `.wrapAll()` function can take any string or object that could be passed to the `$()` factory function to specify a DOM structure. This structure may be nested several levels deep, but should contain only one inmost element. The structure will be wrapped around all of the elements in the set of matched elements as a single group.

Consider the following HTML code:

```
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

Using `.wrapAll()`, we can insert an HTML structure around the inner `<div>` elements as follows:

```
$('.inner').wrapAll('<div class="new" />');
```

The new `<div>` element is created on the fly and added to the DOM. The result is a new `<div>` wrapped around all matched elements.

```
<div class="container">
  <div class="new">
    <div class="inner">Hello</div>
    <div class="inner">Goodbye</div>
  </div>
</div>
```

.wrapInner()

Wrap an HTML structure around the content of each element in the set of matched elements.

```
.wrapInner(wrappingElement)
.wrapInner(wrappingFunction)
```

Parameters (first version)

- `wrappingElement`: An HTML snippet, a selector expression, a jQuery object, or a DOM element specifying the structure to wrap around the content of the matched elements

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Parameters (second version)

- `wrappingFunction`: A callback function that generates a structure to wrap around the content of the matched elements

Return value

The jQuery object, for chaining purposes.

Description

The `.wrapInner()` function can take any string or object that could be passed to the `$()` factory function to specify a DOM structure. This structure may be nested several levels deep, but should contain only one inmost element. The structure will be wrapped around the content of each of the elements in the set of matched elements.

Consider the following HTML code:

```
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

Using `.wrapInner()`, we can insert an HTML structure around the content of the inner `<div>` elements as follows:

```
$('.inner').wrapInner('<div class="new" />');
```

The new `<div>` element is created on the fly and added to the DOM. The result is a new `<div>` wrapped around the content of each matched element.

```
<div class="container">
  <div class="inner">    <div class="new">Hello</div>
  </div>
  <div class="inner">
    <div class="new">Goodbye</div>
  </div>
</div>
```

The second version of this method allows us to specify a callback function instead. This callback function will be called once for every matched element; it should return a DOM element, jQuery object, or HTML snippet in which to wrap the content of the corresponding element. For example:

```
$('.inner').wrapInner(function() {
  return '<div class="' + this.nodeValue + '" />';
});
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

This will cause each new `<div>` to have a class corresponding to the text it wraps.

```
<div class="container">
  <div class="inner">
    <div class="Hello">Hello</div>
  </div>
  <div class="inner">
    <div class="Goodbye">Goodbye</div>
  </div>
</div>
```

DOM copying

This method allows us to make copies of elements.

.clone()

Create a copy of the set of matched elements.

```
.clone([withEvents])
```

Parameters

- `withEvents` (optional): A Boolean indicating whether event handlers should be copied along with the elements

Return value

A new jQuery object referencing the created elements.

Description

The `.clone()` method, when used in conjunction with one of the insertion methods, is a convenient way to duplicate elements on a page. Consider the following HTML code:

```
<div class="container">
  <div class="hello">Hello</div>
  <div class="goodbye">Goodbye</div>
</div>
```

As shown in the *Description* for `.append()`, normally when we insert an element somewhere in the DOM, it is moved from its old location. So, suppose this code is used:

```
$('.hello').appendTo('.goodbye');
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

The resulting DOM structure will be as follows:

```
<div class="container">
  <div class="goodbye">
    Goodbye
    <div class="hello">Hello</div>
  </div>
</div>
```

To prevent this and instead create a copy of the element, we could write the following:

```
$('.hello').clone()
.appendTo('.goodbye');
```

This will produce the following:

```
<div class="container">
  <div class="hello">Hello</div>
  <div class="goodbye">
    Goodbye
    <div class="hello">Hello</div>
  </div>
</div>
```

Note that when using the `.clone()` method, we can modify the cloned elements or their contents before (re-)inserting them into the document.

Normally, any event handlers bound to the original element are *not* copied to the clone. The optional `withEvents` parameter allows us to change this behavior, and instead make copies of all the event handlers as well, bound to the new copy of the element. As of jQuery 1.4, all element data attached by the `.data()` method will be copied along with the event handlers.

DOM removal

These methods allow us to delete elements from the DOM.

.empty()

Remove all child nodes of the set of matched elements from the DOM.

```
.empty()
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Parameters

None

Return value

The jQuery object, for chaining purposes.

Description

This method removes not only child (and other descendant) elements, but also any text within the set of matched elements. This is because according to the DOM specification, any string of text within an element is considered a child node of that element. Consider the following HTML code:

```
<div class="container">
  <div class="hello">Hello</div>
  <div class="goodbye">Goodbye</div>
</div>
```

We can target any element for removal.

```
$('.hello').empty();
```

This will result in a DOM structure with the "Hello" text deleted.

```
<div class="container">
  <div class="hello"></div>
  <div class="goodbye">Goodbye</div>
</div>
```

If we had any number of nested elements inside `<div class="hello">`, they would be removed, too. Other jQuery constructs such as data or event handlers are erased as well.

.remove()

Remove the set of matched elements from the DOM.

```
.remove([selector])
```

Parameters

- `selector` (optional): A selector expression that filters the set of matched elements to be removed

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Return value

The jQuery object, for chaining purposes. Note that the removed elements are still referenced by this object, even though they are no longer in the DOM.

Description

Similar to `.empty()`, the `.remove()` method takes elements out of the DOM. We use `.remove()` when we want to remove the element itself, as well as everything inside it. In addition to the elements themselves, all bound events and jQuery data associated with the elements are removed.

Consider the following HTML code:

```
<div class="container">
  <div class="hello">Hello</div>
  <div class="goodbye">Goodbye</div>
</div>
```

We can target any element for removal.

```
$('.hello').remove();
```

This will result in a DOM structure with the `<div>` element deleted.

```
<div class="container">
  <div class="goodbye">Goodbye</div>
</div>
```

If we had any number of nested elements inside `<div class="hello">`, they would be removed, too. Other jQuery constructs such as data or event handlers are erased as well.

We can also include a selector as an optional parameter. For example, we could rewrite the previous DOM removal code as follows:

```
$('div').remove('.hello');
```

This would result in the same DOM structure.

```
<div class="container">
  <div class="goodbye">Goodbye</div>
</div>
```


For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

.detach()

Remove the set of matched elements from the DOM.

```
.detach([selector])
```

Parameters

- `selector` (optional): A selector expression that filters the set of matched elements to be removed.

Return value

The jQuery object, for chaining purposes. Note that the removed elements are still referenced by this object, even though they are no longer in the DOM.

Description

The `.detach()` method is the same as `.remove()`, except that `.detach()` keeps all jQuery data associated with the removed elements. This method is useful when removed elements are to be reinserted into the DOM at a later time.

.unwrap()

Remove the parents of the set of matched elements from the DOM, leaving the matched elements in their place.

```
.unwrap()
```

Parameters

None

Return value

The jQuery object, for chaining purposes.

Description

The `.unwrap()` method removes the element's parent. This is effectively the inverse of the `.wrap()` method. The matched elements (and their siblings, if any) replace their parents within the DOM structure.

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book

Where to buy this book

You can buy jQuery 1.4 Reference Guide from the Packt Publishing website: `http://www.packtpub.com/jquery-1-4-reference-guide/book`

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

www.PacktPub.com

For More Information: www.PacktPub.com/jquery-1-4-reference-guide/book