9
jQuery Properties

In addition to the many methods jQuery offers for interacting with the DOM and data, a number of properties are available for inspecting both the browser environment and individual jQuery objects.

Some of the examples in this chapter use the `$.print()` function to print results to the page. This is a simple plug-in, which will be discussed in Chapter 10, *Plug-in API*.

Global properties

These properties are associated with the global jQuery object. They allow us to retrieve information about the user agent (web browser) that is executing the script and its features.

$.browser

Information about the user agent displaying the page.

The `$.browser` property allows us to detect which web browser is accessing the page, as reported by the browser itself. It contains flags for each of the four most prevalent browser classes (Internet Explorer, Mozilla, Safari, and Opera) as well as version information.

```
$.print($.browser);
```

```
{
  version: 1.9.1.3,
  safari: false,
  opera: false,
  msie: false,
  mozilla: true
}
```

This property is available immediately. Therefore, it is safe to use it to determine whether to call `$(document).ready()` or not.

> The `$.browser` property is deprecated in jQuery 1.3, but there are no immediate plans to remove it.

As `$.browser` uses `navigator.useragent` to determine the platform, it is vulnerable to spoofing by the user or misrepresentation by the browser itself. It is always best to avoid browser-specific code entirely wherever possible. The `$.support` property is available for detection of support for particular features rather than relying on `$.browser`.

$.support

> Information about the browser's support for specific rendering and JavaScript features.

Rather than using `$.browser` to detect the current user agent and alter the page presentation based on which browser is running, it is a good practice to perform **feature detection**. This means that prior to executing code that relies on a browser feature, we test to ensure that the feature works properly. To make this process simpler, jQuery performs many such tests and makes the results available to us as properties of the `$.support` object.

$.support.boxModel

This property is `true` if the current page has been rendered according to the specifications of the W3C CSS box model.

`http://www.w3.org/TR/CSS2/box.html`

$.support.cssFloat

This property is `true` if the name of the property containing the CSS `float` value is `.cssFloat`, as required by the Document Object Model CSS specification.

```
http://www.w3.org/TR/DOM-Level-2-Style/css.html#CSS-CSS2Properties-
cssFloat
```

$.support.hrefNormalized

This property is `true` if the `.getAttribute()` method retrieves the `href` attribute of elements unchanged, rather than normalizing it to a fully qualified URL.

```
http://www.w3.org/TR/DOM-Level-3-Core/core.html#ID-666EE0F9
```

$.support.htmlSerialize

This property is `true` if the browser is able to serialize/insert `<link>` elements using the `.innerHTML` property of elements.

```
http://www.w3.org/TR/2008/WD-html5-20080610/serializing.html#html-
fragment
```

$.support.leadingWhitespace

This property is `true` if the browser inserts content with `.innerHTML` exactly as provided; specifically, if leading whitespace characters are preserved.

```
http://www.w3.org/TR/2008/WD-html5-20080610/dom.html#innerhtml0
```

$.support.noCloneEvent

This property is `true` if cloned DOM elements are created without event handlers (that is, if the event handlers on the source element are not cloned).

```
http://www.w3.org/TR/DOM-Level-2-Events/events.html#Events-
Registration-interfaces-h3
```

$.support.objectAll

This property is `true` if the `.getElementsByTagName()` method returns all descendant elements when called with a wildcard argument (`'*'`).

```
http://www.w3.org/TR/WD-DOM/level-one-core.html#ID-745549614
```

$.support.opacity

This property is `true` if the browser respects the CSS property `opacity`.

`http://www.w3.org/TR/css3-color/#transparency`

$.support.scriptEval

This property is `true` if inline scripts are automatically evaluated and executed when inserted to the document using standard DOM manipulation methods, such as `.appendChild`.

`http://www.w3.org/TR/2008/WD-html5-20080610/tabular.html#script`

$.support.style

This property is `true` if inline styles for an element can be accessed through the DOM attribute called `style`, as required by the DOM Level 2 specification. In this case, `.getAttribute('style')` can retrieve this value; in Internet Explorer, `.cssText` is used for this purpose.

`http://www.w3.org/TR/DOM-Level-2-Style/css.html#CSS-ElementCSSInlineStyle`

$.support.tbody

This property is `true` if a `<table>` element can exist without a `<tbody>` element. According to the HTML specification, this subelement is optional, so the property should be `true` in a fully compliant browser. If `false`, we must account for the possibility of the browser injecting `<tbody>` tags implicitly.

`http://dev.w3.org/html5/spec/Overview.html#the-table-element`

jQuery object properties

Each jQuery object we create contains a number of properties alongside its methods. These properties allow us to inspect various attributes of the object.

.length

The number of DOM elements matched by the jQuery object.

Suppose we had a simple unordered list on the page:

```
<ul>
  <li>foo</li>
  <li>bar</li>
</ul>
```

We can determine the number of list items by examining `.length`.

```
$.print('Length: ' + $('li').length);
```

This will output the count of items:

Length: 2

.selector

The selector string that was used to create the jQuery object.

The `.live()` method for binding event handlers uses this property to determine how to perform its searches. Plug-ins that perform similar tasks may also find the property useful.

This property contains a string representing the matched set of elements. However, if DOM traversal methods have been called on the object, the string may not be a valid jQuery selector expression. For example, examine the value of the property on a newly-created jQuery object:

```
$.print($('ul li.foo').selector);
```

ul li.foo

Compare this with the value if the same elements are selected with a series of method calls:

```
$.print($('ul').find('li').filter('.foo').selector);
```

ul li.filter(.foo)

For this reason, the value of `.selector` is generally most useful immediately following the original creation of the object. Consequently, the `.live()` method should only be used in this scenario.

.context

The DOM context that was used to create the jQuery object.

The `.live()` method for binding event handlers uses this property to determine the root element to use for its event delegation needs. Plug-ins that perform similar tasks may also find the property useful.

The value of this property is typically equal to `document`, as this is the default context for jQuery objects if none is supplied. The context may differ if, for example, the object was created by searching within an `<iframe>` or XML document.