- JQuery selectors allow you to select and manipulate HTML elements and CSS.
- JQuery is tailor-made to respond to events in an HTML page. You can use JQuery to attach a function to an event handler for the selected elements.
- JQuery is a fast, small, and feature-rich JavaScript Library.
- All selectors in JQuery start with the dollar sign and paretheses : $()

jQuery Syntax

The jQuery syntax is tailor made for **selecting** HTML elements and performing some **action** on the element(s).

Basic syntax is: **$(*selector*).*action*()**

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

Examples:

$(this).hide() - hides the current element.

$("p").hide() - hides all <p> elements.

$(".test").hide() - hides all elements with class="test".

$("#test").hide() - hides the element with id="test".

The Document Ready Event

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

```
$(document).ready(function(){

   // jQuery methods go here...

});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).

It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

Here are some examples of actions that can fail if methods are run before the document is fully loaded:

- Trying to hide an element that is not created yet
- Trying to get the size of an image that is not loaded yet

Tip: The jQuery team has also created an even shorter method for the document ready event:

```
$(function(){

   // jQuery methods go here...

});
```

Use the syntax you prefer. We think that the document ready event is easier to understand when reading the code.

jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s).

jQuery selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.

All selectors in jQuery start with the dollar sign and parentheses: $().

jQuery Syntax For Event Methods

In jQuery, most DOM events have an equivalent jQuery method.

To assign a click event to all paragraphs on a page, you can do this:

```
$("p").click();
```

The next step is to define what should happen when the event fires. You must pass a function to the event:

```
$("p").click(function(){
  // action goes here!!
});
```

jQuery Callback Functions

JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors.

To prevent this, you can create a callback function.

A callback function is executed after the current effect is finished.

Typical syntax: **$(*selector*).hide(*speed,callback*);**

Examples

The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

Example with Callback

```
$("button").click(function(){
    $("p").hide("slow", function(){
        alert("The paragraph is now hidden");
    });
});
```

Try it yourself »

jQuery Method Chaining

Until now we have been writing jQuery statements one at a time (one after the other).

However, there is a technique called chaining, that allows us to run multiple jQuery commands, one after the other, on the same element(s).

Tip: This way, browsers do not have to find the same element(s) more than once.

To chain an action, you simply append the action to the previous action.

The following example chains together the css(), slideUp(), and slideDown() methods. The "p1" element first changes to red, then it slides up, and then it slides down:

Example

```
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
```

Try it yourself »

jQuery DOM Manipulation

One very important part of jQuery is the possibility to manipulate the DOM.

jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes.

DOM = Document Object Model

The DOM defines a standard for accessing HTML and XML documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface and scripts to dynamically access and update the content, structure, and style of a doc

Get Content - text(), html(), and val()

Three simple, but useful, jQuery methods for DOM manipulation are:

- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields

The following example demonstrates how to get content with the jQuery text() and html() methods:

Example

```
$("#btn1").click(function(){
    alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
    alert("HTML: " + $("#test").html());
});
```

Try it yourself »

jQuery addClass() Method

The following example shows how to add class attributes to different elements. Of course you can select multiple elements, when adding classes:

Example

```
$("button").click(function(){
    $("h1, h2, p").addClass("blue");
    $("div").addClass("important");
});
```

Try it yourself »

jQuery removeClass() Method

The following example shows how to remove a specific class attribute from different elements:

Example

```
$("button").click(function(){
    $("h1, h2, p").removeClass("blue");
});
```

Try it yourself »

jQuery toggleClass() Method

The following example will show how to use the jQuery toggleClass() method. This method toggles between adding/removing classes from the selected elements:

Example

```
$("button").click(function(){
    $("h1, h2, p").toggleClass("blue");
});
```

Try it yourself »

jQuery css() Method

The css() method sets or returns one or more style properties for the selected elements.

Return a CSS Property

To return the value of a specified CSS property, use the following syntax:

```
css("propertyname");
```

The following example will return the background-color value of the FIRST matched element:

Example

```
$("p").css("background-color");
```

Try it yourself »

Set a CSS Property

To set a specified CSS property, use the following syntax:

```
css("propertyname","value");
```

The following example will set the background-color value for ALL matched elements:

Example

```
$("p").css("background-color", "yellow");
```

Try it yourself »

Set Multiple CSS Properties

To set multiple CSS properties, use the following syntax:

```
css({"propertyname":"value","propertyname":"value",...});
```

The following example will set a background-color and a font-size for ALL matched elements:

Example

```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

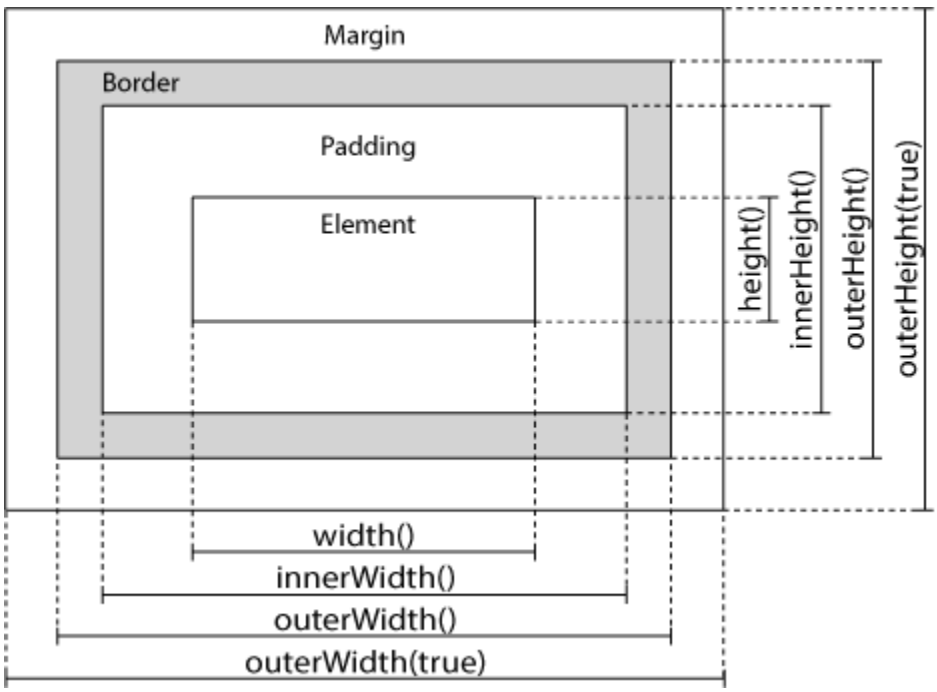
Try it yourself »

jQuery Dimension Methods

jQuery has several important methods for working with dimensions:

- width()
- height()
- innerWidth()
- innerHeight()
- outerWidth()
- outerHeight()

jQuery Dimensions



jQuery width() and height() Methods

The width() method sets or returns the width of an element (excludes padding, border and margin).

The height() method sets or returns the height of an element (excludes padding, border and margin).

The following example returns the width and height of a specified element:

Example

```
$("button").click(function(){
    var txt = "";
    txt += "Width: " + $("#div1").width() + "</br>";
    txt += "Height: " + $("#div1").height();
    $("#div1").html(txt);
});
```

Try it yourself »

jQuery innerWidth() and innerHeight() Methods

The innerWidth() method returns the width of an element (includes padding).

The innerHeight() method returns the height of an element (includes padding).

The following example returns the inner-width/height of a specified element:

Example

```
$("button").click(function(){
    var txt = "";
    txt += "Inner width: " + $("#div1").innerWidth() + "</br>";
```

```
    txt += "Inner height: " + $("#div1").innerHeight();
    $("#div1").html(txt);
});
```

Try it yourself »

jQuery outerWidth() and outerHeight() Methods

The outerWidth() method returns the width of an element (includes padding and border).

The outerHeight() method returns the height of an element (includes padding and border).

The following example returns the outer-width/height of a specified element:

Example

```
$("button").click(function(){
    var txt = "";
    txt += "Outer width: " + $("#div1").outerWidth() + "</br>";
    txt += "Outer height: " + $("#div1").outerHeight();
    $("#div1").html(txt);
});
```

Try it yourself »

The outerWidth(true) method returns the width of an element (includes padding, border, and margin).

The outerHeight(true) method returns the height of an element (includes padding, border, and margin).

Example

```
$("button").click(function(){
    var txt = "";
    txt += "Outer width (+margin): " + $("#div1").outerWidth(true) + "</br>";
    txt += "Outer height (+margin): " + $("#div1").outerHeight(true);
    $("#div1").html(txt);
});
```

Try it yourself »

jQuery More width() and height()

The following example returns the width and height of the document (the HTML document) and window (the browser viewport):

Example

```
$("button").click(function(){
    var txt = "";
    txt += "Document width/height: " + $(document).width();
    txt += "x" + $(document).height() + "\n";
    txt += "Window width/height: " + $(window).width();
    txt += "x" + $(window).height();
    alert(txt);
});
```

Try it yourself »

The following example sets the width and height of a specified element:

Example

```
$("button").click(function(){
    $("#div1").width(500).height(500);
});
```

Try it yourself »

jQuery Traversing - Ancestors

« Previous

Next Chapter »

An ancestor is a parent, grandparent, great-grandparent, and so on.

With jQuery you can traverse up the DOM tree to find ancestors of an element.

Traversing Up the DOM Tree

Three useful jQuery methods for traversing up the DOM tree are:

- parent()
- parents()
- parentsUntil()

jQuery parent() Method

The parent() method returns the direct parent element of the selected element.

This method only traverse a single level up the DOM tree.

The following example returns the direct parent element of each elements:

Example

```
$(document).ready(function(){
    $("span").parent();
});
```

Try it yourself »

jQuery parents() Method

The parents() method returns all ancestor elements of the selected element, all the way up to the document's root element (<html>).

The following example returns all ancestors of all elements:

Example

```
$(document).ready(function(){
    $("span").parents();
});
```

Try it yourself »

You can also use an optional parameter to filter the search for ancestors.

The following example returns all ancestors of all elements that are <ul> elements:

Example

```
$(document).ready(function(){
    $("span").parents("ul");
});
```

Try it yourself »

jQuery Traversing - Descendants

« Previous

Next Chapter »

A descendant is a child, grandchild, great-grandchild, and so on.

With jQuery you can traverse down the DOM tree to find descendants of an element.

Traversing Down the DOM Tree

Two useful jQuery methods for traversing down the DOM tree are:

- children()
- find()

jQuery children() Method

The children() method returns all direct children of the selected element.

This method only traverse a single level down the DOM tree.

The following example returns all elements that are direct children of each elements:

Example

```
$(document).ready(function(){
    $("div").children();
});
```

Try it yourself »

You can also use an optional parameter to filter the search for children.

The following example returns all <p> elements with the class name "first", that are direct children of :

Example

```
$(document).ready(function(){
    $("div").children("p.first");
});
```

Try it yourself »

jQuery find() Method

The find() method returns descendant elements of the selected element, all the way down to the last descendant.

The following example returns all elements that are descendants of :

Example

```
$(document).ready(function(){
    $("div").find("span");
});
```

Try it yourself »

The following example returns all descendants of :

Example

```
$(document).ready(function(){
    $("div").find("*");
});
```

Try it yourself »

Traversing Sideways in The DOM Tree

There are many useful jQuery methods for traversing sideways in the DOM tree:

- siblings()

- next()
- nextAll()
- nextUntil()
- prev()
- prevAll()
- prevUntil()

jQuery siblings() Method

The siblings() method returns all sibling elements of the selected element.

The following example returns all sibling elements of <h2>:

Example

```
$(document).ready(function(){
    $("h2").siblings();
});
```

Try it yourself »

You can also use an optional parameter to filter the search for siblings.

The following example returns all sibling elements of <h2> that are <p> elements:

Example

```
$(document).ready(function(){
    $("h2").siblings("p");
});
```

Try it yourself »

jQuery next() Method

The next() method returns the next sibling element of the selected element.

The following example returns the next sibling of <h2>:

Example

```
$(document).ready(function(){
    $("h2").next();
});
```

Try it yourself »

jQuery nextAll() Method

The nextAll() method returns all next sibling elements of the selected element.

The following example returns all next sibling elements of <h2>:

Example

```
$(document).ready(function(){
    $("h2").nextAll();
});
```

Try it yourself »

jQuery nextUntil() Method

The nextUntil() method returns all next sibling elements between two given arguments.

The following example returns all sibling elements between a <h2> and a <h6> element:

Example

```
$(document).ready(function(){
    $("h2").nextUntil("h6");
});
```

Try it yourself »

jQuery prev(), prevAll() & prevUntil() Methods

The prev(), prevAll() and prevUntil() methods work just like the methods above but with reverse functionality: they return previous sibling elements (traverse backwards along sibling elements in the DOM tree, instead of forward).

jQuery Traversing - Filtering

« Previous

Next Chapter »

Narrow Down The Search For Elements

The three most basic filtering methods are first(), last() and eq(), which allow you to select a specific element based on its position in a group of elements.

Other filtering methods, like filter() and not() allow you to select elements that match, or do not match, a certain criteria.

jQuery first() Method

The first() method returns the first element of the selected elements.

The following example selects the first <p> element inside the first element:

Example

```
$(document).ready(function(){
    $("div p").first();
});
```

Try it yourself »

jQuery last() Method

The last() method returns the last element of the selected elements.

The following example selects the last <p> element inside the last element:

Example

```
$(document).ready(function(){
    $("div p").last();
});
```

Try it yourself »

jQuery eq() method

The eq() method returns an element with a specific index number of the selected elements.

The index numbers start at 0, so the first element will have the index number 0 and not 1. The following example selects the second <p> element (index number 1):

Example

```
$(document).ready(function(){
    $("p").eq(1);
});
```

Try it yourself »

jQuery filter() Method

The filter() method lets you specify a criteria. Elements that do not match the criteria are removed from the selection, and those that match will be returned.

The following example returns all <p> elements with class name "intro":

Example

```
$(document).ready(function(){
    $("p").filter(".intro");
});
```

Try it yourself »

jQuery not() Method

The not() method returns all elements that do not match the criteria.

Tip: The not() method is the opposite of filter().

The following example returns all <p> elements that do not have class name "intro":

Example

```
$(document).ready(function(){
    $("p").not(".intro");
});
```

Try it yourself »