2013

JQuery

Satya Johnny

satyajohnny@blogspot.com

Tree Notes

Syntax: **$(*selector*).*action* ()**

Example:

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

The #id Selector : `$("#test").hide();`

The .class Selector : `$(".test").hide();`

Callback Function : $(*selector*).hide(*speed,callback*);

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

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

jQuery HTML

- text() – Sets/ returns the text content of selected elements ***$("#test").text()***
- html() – Sets/returns the HTML content of selected elements ***$("#test").html()***
- val() – Sets/returns the value of form fields ***$("#test").val()***
- attr() – Get Attribute like link values ***$("#w3s").attr("href")***

Add/Remove New HTML Content

append()	prepend()	after()	before()
remove()	empty()		
addClass()	removeClass()	toggleClass()	css()

jQuery Tutorial

jQuery is a JavaScript Library.

jQuery greatly simplifies JavaScript programming.

jQuery is easy to learn.

The purpose of jQuery is to make it much easier to use JavaScript on your website.

What is jQuery?

jQuery is a lightweight, "write less, do more", JavaScript library.

The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Tip: In addition, jQuery has plugins for almost any task out there.

Will jQuery work in all browsers?
The jQuery team knows all about cross-browser issues, and they have written this knowledge into the jQuery library. jQuery will run exactly the same in all major browsers, including Internet Explorer 6!

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:

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: $().

The element Selector

The jQuery element selector selects elements based on the element name.

You can select all <p> elements on a page like this:

```
$("p")
```

Example

When a user clicks on a button, all <p> elements will be hidden:

Example

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">
</script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>
</html>
```

The #id Selector

The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.

An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.

To find an element with a specific id, write a hash character, followed by the id of the element:

```
$("#test")
```

Example

When a user clicks on a button, the element with id="test" will be hidden:

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p id="test">This is another paragraph.</p>
<button>Click me</button>
</body>
```

The .class Selector

The jQuery class selector finds elements with a specific class.

To find elements with a specific class, write a period character, followed by the name of the class:

```
$(".test")
```

```
<script>

$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide();
  });
});
</script>
</head>
<body>

<h2 class="test">This is a heading</h2>
<p class="test">This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>
```

More Examples of jQuery Selectors

Syntax	Description
$("*")	Selects all elements
$(this)	Selects the current HTML element
$("p.intro")	Selects all <p> elements with class="intro"
$("p:first")	Selects the first <p> element
$("ul li:first")	Selects the first element of the first
$("ul li:first-child")	Selects the first element of every
$("[href]")	Selects all elements with an href attribute
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"
$(":button")	Selects all <button> elements and <input> elements of type="button"
$("tr:even")	Selects all even <tr> elements
$("tr:odd")	Selects all odd <tr> elements

Functions In a Separate File

If your website contains a lot of pages, and you want your jQuery functions to be easy to maintain, you can put your jQuery functions in a separate .js file.

When we demonstrate jQuery in this tutorial, the functions are added directly into the <head> section. However, sometimes it is preferable to place them in a separate file, like this (use the src attribute to refer to the .js file):

Example

```
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">
</script>
<script src="my_jquery_functions.js"></script>
</head>
```

jQuery Event Methods

jQuery is tailor-made to respond to events in an HTML page.

What are Events?

All the different visitor's actions that a web page can respond to are called events.An event represents the precise moment when something happens.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element .**Here are some common DOM events:**

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

jQuery Event Methods

Event methods trigger or attach a function to an event handler for the selected elements.

The following table lists all the jQuery methods used to handle events.

Method	Description
bind()	Attaches event handlers to elements
blur()	Attaches/Triggers the blur event
change()	Attaches/Triggers the change event
click()	Attaches/Triggers the click event
dblclick()	Attaches/Triggers the double click event
delegate()	Attaches a handler to current, or future, specified child elements of the matching el
die()	Removed in version 1.9. Removes all event handlers added with the live()
error()	Deprecated in version 1.8. Attaches/Triggers the error event
event.currentTarget	The current DOM element within the event bubbling phase
event.data	Contains the optional data passed to an event method when the current executing handler is bo
event.delegateTarget	Returns the element where the currently-called jQuery event handler was
event.isDefaultPrevented()	Returns whether event.preventDefault() was called for the event object
event.isImmediatePropagationStopped()	Returns whether event.stopImmediatePropagation() was called for the event object
event.isPropagationStopped()	Returns whether event.stopPropagation() was called for the event object
event.namespace	Returns the namespace specified when the event was triggered
event.pageX	Returns the mouse position relative to the left edge of the document
event.pageY	Returns the mouse position relative to the top edge of the document
event.preventDefault()	Prevents the default action of the event

event.relatedTarget	Returns which element being entered or exited on mouse movement.
event.result	Contains the last/previous value returned by an event handler triggered by
event.stopImmediatePropagation()	Prevents other event handlers from being called
event.stopPropagation()	Prevents the event from bubbling up the DOM tree, preventing any parent
event.target	Returns which DOM element triggered the event
event.timeStamp	Returns the number of milliseconds since January 1, 1970, when the event
event.type	Returns which event type was triggered
event.which	Returns which keyboard key or mouse button was pressed for the event
focus()	Attaches/Triggers the focus event
focusin()	Attaches an event handler to the focusin event
focusout()	Attaches an event handler to the focusout event
hover()	Attaches two event handlers to the hover event
keydown()	Attaches/Triggers the keydown event
keypress()	Attaches/Triggers the keypress event
keyup()	Attaches/Triggers the keyup event
live()	Removed in version 1.9. Adds one or more event handlers to current, or fu
load()	Deprecated in version 1.8. Attaches an event handler to the load event
mousedown()	Attaches/Triggers the mousedown event
mouseenter()	Attaches/Triggers the mouseenter event
mouseleave()	Attaches/Triggers the mouseleave event
mousemove()	Attaches/Triggers the mousemove event
mouseout()	Attaches/Triggers the mouseout event

mouseover()	Attaches/Triggers the mouseover event
mouseup()	Attaches/Triggers the mouseup event
off()	Removes event handlers attached with the on() method
on()	Attaches event handlers to elements
one()	Adds one or more event handlers to selected elements. This handler can o
$.proxy()	Takes an existing function and returns a new one with a particular context
ready()	Specifies a function to execute when the DOM is fully loaded
resize()	Attaches/Triggers the resize event
scroll()	Attaches/Triggers the scroll event
select()	Attaches/Triggers the select event
submit()	Attaches/Triggers the submit event
toggle()	Removed in version 1.9. Attaches two or more functions to toggle betwee
trigger()	Triggers all events bound to the selected elements
triggerHandler()	Triggers all functions bound to a specified event for the selected elements
unbind()	Removes an added event handler from selected elements
undelegate()	Removes an event handler to selected elements, now or in the future
unload()	Deprecated in version 1.8. Attaches an event handler to the unload event

jQuery Effects - Hide and Show

Hide, Show, Toggle, Slide, Fade, and Animate. WOW!

jQuery Effect Methods

The following table lists all the jQuery methods for creating animation effects.

Method	Description
animate()	Runs a custom animation on the selected elements
clearQueue()	Removes all remaining queued functions from the selected elements
delay()	Sets a delay for all queued functions on the selected elements
dequeue()	Removes the next function from the queue, and then executes the function
fadeIn()	Fades in the selected elements
fadeOut()	Fades out the selected elements
fadeTo()	Fades in/out the selected elements to a given opacity
fadeToggle()	Toggles between the fadeIn() and fadeOut() methods
finish()	Stops, removes and completes all queued animations for the selected elements
hide()	Hides the selected elements
queue()	Shows the queued functions on the selected elements
show()	Shows the selected elements
slideDown()	Slides-down (shows) the selected elements
slideToggle()	Toggles between the slideUp() and slideDown() methods
slideUp()	Slides-up (hides) the selected elements
stop()	Stops the currently running animation for the selected elements
toggle()	Toggles between the hide() and show() methods

jQuery Callback Functions

A callback function is executed after the current effect is 100% finished.

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:

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide("slow",function(){
      alert("The paragraph is now hidden");
    });
  });
});
</script>
</head>
<body>
<button>Hide</button>
<p>This is a paragraph with little content.</p>
</body>
Without call back
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide(1000);
    alert("The paragraph is now hidden");
  });
});
```

jQuery - Chaining

With jQuery, you can chain together actions/methods.Chaining allows us to run multiple jQuery methods (on the same element) within a single statement.

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);
```

jQuery HTML

Get Content and Attributes

jQuery contains powerful methods for changing and manipulating HTML elements and attributes.

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 that allows programs and*
> *the content, structure, and style of a document."*

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

```
<script>
$(document).ready(function(){
  $("#btn1").click(function(){    alert("Text: " + $("#test").text());
  });
  $("#btn2").click(function(){
    alert("HTML: " + $("#test").html());  });
});
</script></head> <body>
<p id="test">This is some <b>bold</b> text in a paragraph.</p>
<button id="btn1">Show Text</button>
<button id="btn2">Show HTML</button>
</body>
```

Get Attributes - attr()

The jQuery attr() method is used to get attribute values.

The following example demonstrates how to get the value of the href attribute in a link:

Example

```
$("button").click(function(){
  alert($("#w3s").attr("href"));
});
```

jQuery HTML / CSS Methods

The following table lists all the methods used to manipulate the HTML and CSS.The methods below work for both HTML and XML documents. Exception: the html() method.

Method	Description
addClass()	Adds one or more class names to selected elements
after()	Inserts content after selected elements
append()	Inserts content at the end of selected elements
appendTo()	Inserts HTML elements at the end of selected elements
attr()	Sets or returns attributes/values of selected elements
before()	Inserts content before selected elements
clone()	Makes a copy of selected elements
css()	Sets or returns one or more style properties for selected elements
detach()	Removes selected elements (keeps data and events)
empty()	Removes all child nodes and content from selected elements
hasClass()	Checks if any of the selected elements have a specified class name
height()	Sets or returns the height of selected elements
html()	Sets or returns the content of selected elements
innerHeight()	Returns the height of an element (includes padding, but not border)
innerWidth()	Returns the width of an element (includes padding, but not border)

insertAfter()	Inserts HTML elements after selected elements
insertBefore()	Inserts HTML elements before selected elements
offset()	Sets or returns the offset coordinates for selected elements (relative to the document)
offsetParent()	Returns the first positioned parent element
outerHeight()	Returns the height of an element (includes padding and border)
outerWidth()	Returns the width of an element (includes padding and border)
position()	Returns the position (relative to the parent element) of an element
prepend()	Inserts content at the beginning of selected elements
prependTo()	Inserts HTML elements at the beginning of selected elements
prop()	Sets or returns properties/values of selected elements
remove()	Removes the selected elements (including data and events)
removeAttr()	Removes one or more attributes from selected elements
removeClass()	Removes one or more classes from selected elements
removeProp()	Removes a property set by the prop() method
replaceAll()	Replaces selected elements with new HTML elements
replaceWith()	Replaces selected elements with new content
scrollLeft()	Sets or returns the horizontal scrollbar position of selected elements
scrollTop()	Sets or returns the vertical scrollbar position of selected elements
text()	Sets or returns the text content of selected elements
toggleClass()	Toggles between adding/removing one or more classes from selected elements
unwrap()	Removes the parent element of the selected elements
val()	Sets or returns the value attribute of the selected elements (for form elements)
width()	Sets or returns the width of selected elements
wrap()	Wraps HTML element(s) around each selected element
wrapAll()	Wraps HTML element(s) around all selected elements
wrapInner()	Wraps HTML element(s) around the content of each selected element

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

```
<script>
$(document).ready(function(){
  $("#btn1").click(function(){
    $("#test1").text("Hello world!");
  });
  $("#btn2").click(function(){
    $("#test2").html("<b>Hello world!</b>");
  });
  $("#btn3").click(function(){
    $("#test3").val("Dolly Duck");
  });
});
</script>
</head>

<body>
<p id="test1">This is a paragraph.</p>
<p id="test2">This is another paragraph.</p>
<p>Input field: <input type="text" id="test3" value="Mickey Mouse"></p>
<button id="btn1">Set Text</button>
<button id="btn2">Set HTML</button>
<button id="btn3">Set Value</button>
</body>
```

A Callback Function for text(), html(), and val()

All of the three jQuery methods above: text(), html(), and val(), also come with a callback function. The callback function has two parameters: the index of the current element in the list of elements selected and the original (old) value. You then return the string you wish to use as the new value from the function.

The following example demonstrates text() and html() with a callback function:

Example

```
$("#btn1").click(function(){
  $("#test1").text(function(i,origText){
    return "Old text: " + origText + " New text: Hello world!
    (index: " + i + ")";
  });
});

$("#btn2").click(function(){
  $("#test2").html(function(i,origText){
    return "Old html: " + origText + " New html: Hello <b>world!</b>
    (index: " + i + ")";
  });
});
```

Set Attributes - attr()

The jQuery attr() method is also used to set/change attribute values.

The following example demonstrates how to change (set) the value of the href attribute in a link:

Example

```
$("button").click(function(){
  $("#w3s").attr("href","http://www.w3schools.com/jquery");
});
```

jQuery - Add Elements

With jQuery, it is easy to add new elements/content.

Add New HTML Content

We will look at four jQuery methods that are used to add new content:

- append() - Inserts content at the end of the selected elements
- prepend() - Inserts content at the beginning of the selected elements
- after() - Inserts content after the selected elements
- before() - Inserts content before the selected elements

Remove Elements/Content

To remove elements and content, there are mainly two jQuery methods:

- remove() - Removes the selected element (and its child elements)
- empty() - Removes the child elements from the selected element

```
$("p").append("Some appended text.");
$("p").prepend("Some prepended text.");
```

Add Several New Elements With append() and prepend()

```
function appendText()
{
var txt1="<p>Text.</p>";              // Create element with HTML
var txt2=$("<p></p>").text("Text.");   // Create with jQuery
var txt3=document.createElement("p");  // Create with DOM
txt3.innerHTML="Text.";
$("p").append(txt1,txt2,txt3);        // Append the new elements
}
```

```
$("img").after("Some text after");

$("img").before("Some text before");
```

Add Several New Elements With after() and before()

```
function afterText()
{
var txt1="<b>I </b>";                  // Create element with HTML
var txt2=$("<i></i>").text("love ");     // Create with jQuery
var txt3=document.createElement("big");  // Create with DOM
txt3.innerHTML="jQuery!";
$("img").after(txt1,txt2,txt3);          // Insert new elements after img
}
```

Remove Elements/Content

To remove elements and content, there are mainly two jQuery methods:

- remove() - Removes the selected element (and its child elements)
- empty() - Removes the child elements from the selected element

```
$("#div1").remove();
$("#div1").empty();
```

Filter the Elements to be Removed

The jQuery remove() method also accepts one parameter, which allows you to filter the elements to be removed.

The parameter can be any of the jQuery selector syntaxes.

The following example removes all <p> elements with class="italic":

Example

```
$("p").remove(".italic");
```

---

jQuery Manipulating CSS

jQuery has several methods for CSS manipulation. We will look at the following methods:

- addClass() - Adds one or more classes to the selected elements
- removeClass() - Removes one or more classes from the selected elements
- toggleClass() - Toggles between adding/removing classes from the selected elements
- css() - Sets or returns the style attribute

jQuery - Dimensions

With jQuery, it is easy to work with the dimensions of elements and browser Window.

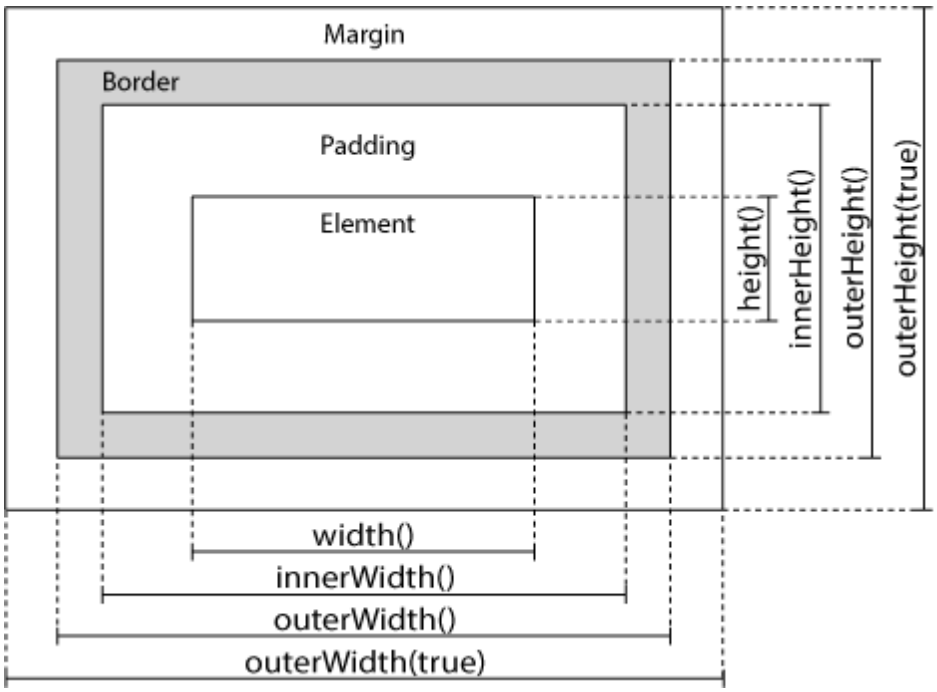
jQuery Dimension Methods

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 (includes NO padding, border, or margin).

The height() method sets or returns the height of an element (includes NO padding, border, or margin).

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

jQuery Traversing

What is Traversing?

jQuery traversing, which means "move through", are used to "find" (or select) HTML elements based on their relation to other elements. Start with one selection and move through that selection until you reach the elements you desire.

The image below illustrates a family tree. With jQuery traversing, you can easily move up (ancestors), down (descendants) and sideways (siblings) in the family tree, starting from the selected (current) element. This movement is called traversing - or moving through - the DOM.

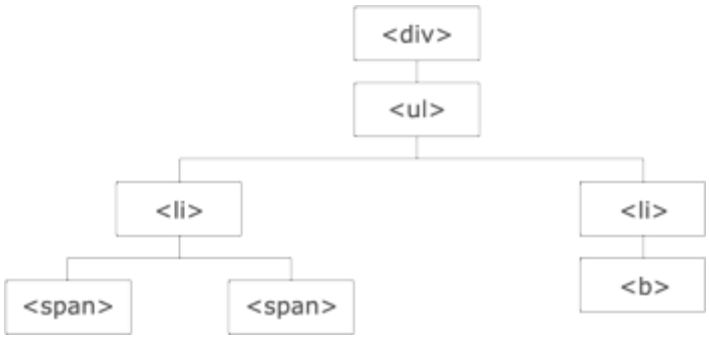


Illustration explained:

- The element is the **parent** of <ul>, and an **ancestor** of everything inside of it
- The <ul> element is the **parent** of both <li> elements, and a **child** of
- The left <li> element is the **parent** of , **child** of <ul> and a **descendant** of
- The element is a **child** of the left <li> and a **descendant** of <ul> and
- The two <li> elements are **siblings** (they share the same parent)
- The right <li> element is the **parent** of <b>, **child** of <ul> and a **descendant** of
- The <b> element is a **child** of the right <li> and a **descendant** of <ul> and

jQuery Traversing Methods

Method	Description
add()	Adds elements to the set of matched elements

addBack()	Adds the previous set of elements to the current set
children()	Returns all direct children of the selected element
closest()	Returns the first ancestor of the selected element
contents()	Returns all direct children of the selected element (including text and comment nodes)
each()	Executes a function for each matched element
end()	Ends the most recent filtering operation in the current chain, and return the set of matched elem
eq()	Returns an element with a specific index number of the selected elements
filter()	Reduce the set of matched elements to those that match the selector or pass the function's test
find()	Returns descendant elements of the selected element
first()	Returns the first element of the selected elements
has()	Returns all elements that have one or more elements inside of them
is()	Checks the set of matched elements against a selector/element/jQuery object, and return true if arguments
last()	Returns the last element of the selected elements
map()	Passes each element in the matched set through a function, producing a new jQuery object conta
next()	Returns the next sibling element of the selected element
nextAll()	Returns all next sibling elements of the selected element
nextUntil()	Returns all next sibling elements between two given arguments
not()	Remove elements from the set of matched elements
offsetParent()	Returns the first positioned parent element
parent()	Returns the direct parent element of the selected element
parents()	Returns all ancestor elements of the selected element
parentsUntil()	Returns all ancestor elements between two given arguments
prev()	Returns the previous sibling element of the selected element
prevAll()	Returns all previous sibling elements of the selected element
prevUntil()	Returns all previous sibling elements between two given arguments

siblings()	Returns all sibling elements of the selected element
slice()	Reduces the set of matched elements to a subset specified by a range of indices

jQuery - AJAX Introduction

AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page

What About jQuery and AJAX?

jQuery provides several methods for AJAX functionality.

With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post - And you can load the external data directly into the selected HTML elements of your web page!

Without jQuery, AJAX coding can be a bit tricky!

Writing regular AJAX code can be a bit tricky, because different browsers have different syntax for AJAX implem
write extra code to test for different browsers. However, the jQuery team has taken care of this for us, so that w
one single line of code.

jQuery load() Method

The jQuery load() method is a simple, but powerful AJAX method.

The load() method loads data from a server and puts the returned data into the selected element.

Syntax:

```
$(selector).load(URL,data,callback);
```

The required URL parameter specifies the URL you wish to load.

The optional data parameter specifies a set of querystring key/value pairs to send along with the request.

The optional callback parameter is the name of a function to be executed after the load() method is completed.

Example:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">
</script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").load("demo_test.txt");
  });
});
</script>
</head>
<body>

<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>
<button>Get External Content</button>
```

It is also possible to add a jQuery selector to the URL parameter.

The following example loads the content of the element with id="p1", inside the file "demo_test.txt", into a specific element:

Example

```
$("#div1").load("demo_test.txt #p1");
```

The optional callback parameter specifies a callback function to run when the load() method is completed. The callback function can have different parameters:

- responseTxt - contains the resulting content if the call succeed
- statusTxt - contains the status of the call
- xhr - contains the XMLHttpRequest object

The following example displays an alert box after the load() method completes. If the load() method has succeed, it displays "External content loaded successfully!", and if it fails it displays an error message:

Example

```
$("button").click(function(){
  $("#div1").load("demo_test.txt",function(responseTxt,statusTxt,xhr){
    if(statusTxt=="success")
      alert("External content loaded successfully!");
    if(statusTxt=="error")
      alert("Error: "+xhr.status+": "+xhr.statusText);
```

```
  });
});
```

jQuery AJAX Methods

AJAX is the art of exchanging data with a server, and update parts of a web page - without reloading the whole page.

The following table lists all the jQuery AJAX methods:

Method	Description
$.ajax()	Performs an async AJAX request
$.ajaxPrefilter()	Handle custom Ajax options or modify existing options before each request is sent and before the
$.ajaxSetup()	Sets the default values for future AJAX requests
$.ajaxTransport()	Creates an object that handles the actual transmission of Ajax data
$.get()	Loads data from a server using an AJAX HTTP GET request
$.getJSON()	Loads JSON-encoded data from a server using a HTTP GET request
$.getScript()	Loads (and executes) a JavaScript from a server using an AJAX HTTP GET request
$.param()	Creates a serialized representation of an array or object (can be used as URL query string for AJA
$.post()	Loads data from a server using an AJAX HTTP POST request
ajaxComplete()	Specifies a function to run when the AJAX request completes
ajaxError()	Specifies a function to run when the AJAX request completes with an error
ajaxSend()	Specifies a function to run before the AJAX request is sent
ajaxStart()	Specifies a function to run when the first AJAX request begins
ajaxStop()	Specifies a function to run when all AJAX requests have completed
ajaxSuccess()	Specifies a function to run when an AJAX request completes successfully
load()	Loads data from a server and puts the returned data into the selected element
serialize()	Encodes a set of form elements as a string for submission
serializeArray()	Encodes a set of form elements as an array of names and values

jQuery - AJAX get() and post() Methods

The jQuery get() and post() methods are used to request data from the server with an HTTP GET or POST request

HTTP Request: GET vs. POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource

GET is basically used for just getting (retrieving) some data from the server. **Note:** The GET method may return cached data.

POST can also be used to get some data from the server. However, the POST method NEVER caches data, and is often used to send data along with the request.

To learn more about GET and POST, and the differences between the two methods, please read our HTTP Methods GET vs POST chapter.

---

jQuery $.get() Method

The $.get() method requests data from the server with an HTTP GET request.

Syntax:

```
$.get(URL,callback);
```

The required URL parameter specifies the URL you wish to request.

The optional callback parameter is the name of a function to be executed if the request succeeds.

The following example uses the $.get() method to retrieve data from a file on the server:

Example

```
$("button").click(function(){
  $.get("demo_test.asp",function(data,status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

The first parameter of $.get() is the URL we wish to request ("demo_test.asp").

The second parameter is a callback function. The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

Tip: Here is how the ASP file looks like ("demo_test.asp"):

```
<%
response.write("This is some text from an external ASP file.")
%>
```

jQuery $.post() Method

The $.post() method requests data from the server using an HTTP POST request.

Syntax:

```
$.post(URL,data,callback);
```

The required URL parameter specifies the URL you wish to request.

The optional data parameter specifies some data to send along with the request.

The optional callback parameter is the name of a function to be executed if the request succeeds.

The following example uses the $.post() method to send some data along with the request:

Example

```
$("button").click(function(){
  $.post("demo_test_post.asp",
  {
    name:"Donald Duck",
    city:"Duckburg"
  },
  function(data,status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

The first parameter of $.post() is the URL we wish to request ("demo_test_post.asp").

Then we pass in some data to send along with the request (name and city).

The ASP script in "demo_test_post.asp" reads the parameters, process them, and return a result.

The third parameter is a callback function. The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

Tip: Here is how the ASP file looks like ("demo_test_post.asp"):

```
<%
dim fname,city
fname=Request.Form("name")
city=Request.Form("city")
Response.Write("Dear " & fname & ". ")
Response.Write("Hope you live well in " & city & ".")
%>
```

jQuery - The noConflict() Method

What if you wish to use other frameworks on your pages, while still using jQuery?

jQuery and Other JavaScript Frameworks

As you already know; jQuery uses the $ sign as a shortcut for jQuery.

What if other JavaScript frameworks also use the $ sign as a shortcut?

Some other popular JavaScript frameworks are: MooTools, Backbone, Sammy, Cappuccino, Knockout, JavaScript MVC, Google Web Toolkit, Google Closure, Ember, Batman, and Ext JS.

Some of the other frameworks also use the $ character as a shortcut (just like jQuery), and then you suddenly have two different frameworks using the same shortcut, which might result in that your scripts stop working.

The jQuery team have already thought about this, and implemented the noConflict() method.

The jQuery noConflict() Method

The noConflict() method releases the hold on the $ shortcut identifier, so that other scripts can use it.You can of course still use jQuery, simply by writing the full name instead of the shortcut:**Example**

```
$.noConflict();
jQuery(document).ready(function(){
  jQuery("button").click(function(){
    jQuery("p").text("jQuery is still working!");  });
});
```

You can also create your own shortcut very easily. The noConflict() method returns a reference to jQuery, that you can save in a variable, for later use. Here is an example:

Example

```
var jq = $.noConflict();
jq(document).ready(function(){
  jq("button").click(function(){
    jq("p").text("jQuery is still working!");
  });
});
```

If you have a block of jQuery code which uses the $ shortcut and you do not want to change it all, you can pass the $ sign in as a parameter to the ready method. This allows you to access jQuery using $, inside this function - outside of it, you will have to use "jQuery":

Example

```
$.noConflict();
jQuery(document).ready(function($){
  $("button").click(function(){
    $("p").text("jQuery is still working!");
  });
});
```

jQuery Misc Methods

Method	Description
data()	Attaches data to, or gets data from, selected elements
each()	Execute a function for each matched element
get()	Get the DOM elements matched by the selector
index()	Search for a given element from among the matched elements
$.noConflict()	Release jQuery's control of the $ variable
$.param()	Create a serialized representation of an array or object (can be used as URL query string for AJA
removeData()	Removes a previously-stored piece of data
size()	Deprecated in version 1.8. Return the number of DOM elements matched by the jQuery selector
toArray()	Retrieve all the DOM elements contained in the jQuery set, as an array

jQuery Examples

jQuery ExamplesjQuery QuizjQuery Certificate