**Chapter 1 :**

* [jQuery Intro](http://www.w3schools.com/jquery/jquery_intro.asp)
* [jQuery Syntax](http://www.w3schools.com/jquery/jquery_syntax.asp)
* [jQuery Selectors](http://www.w3schools.com/jquery/jquery_selectors.asp)
* [jQuery Events](http://www.w3schools.com/jquery/jquery_events.asp)

**Introduction :**

**jQuery History :**

jQuery was first released in January 2006 by **John Resig** at BarCamp NYC. It is currently

headed by Timmy Wilson and maintained by a team of developers.

Nowadays, jQuery is widely used technology. Most of the websites are using jQuery.

**jQuery Release History :**

Let's see the release dates of jQuery versions.

|  |  |
| --- | --- |
| **Version No.** | **Release Date** |
| 1.0 | 26,August,2006 |
| 1.1 | 14,January,2007 |
| 1.2 | 10, September, 2007 |
| 1.3 | 14, January, 2009 |
| 1.4 | 14, January, 2010 |
| 1.5 | 31, January, 2011 |
| 1.6 | 3, May, 2011 |
| 1.7 | 3, November, 2011 |
| 1.8 | 9, August, 2012 |
| 1.9 | 15, January, 2013 |
| 1.10 | 24,May, 2013 |
| 1.11 | 24, January, 2014 |
| 2.0 | 18, April, 2013 |
| 2.1 | 24, January, 2014 |

## What You Should Already Know :

## Before you start studying jQuery, you should have a basic knowledge of:

* HTML
* CSS
* JavaScript

## 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

## Why jQuery?

## There are lots of other JavaScript frameworks out there, but jQuery seems to be the most

## popular, and also the most extendable.

## Many of the biggest companies on the Web use jQuery, such as:

* Google
* Microsoft
* IBM
* Netflix

## Adding jQuery to Your Web Pages :

## There are several ways to start using jQuery on your web site. You can:

* Download the jQuery library from jQuery.com
* Include jQuery from a CDN, like Google

**Downloading jQuery :**

There are two versions of jQuery available for downloading:

* Production version - this is for your live website because it has been minified and compressed
* Development version - this is for testing and development (uncompressed and readable code)

Both versions can be downloaded from [jQuery.com](http://jquery.com/download/).

The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag

(notice that the <script> tag should be inside the <head> section):

<head>

<script src="jquery-1.11.3.min.js"></script>

</head>

## jQuery CDN :

If you don't want to download and host jQuery yourself, you can include it from

a CDN (Content Delivery Network).

Both Google and Microsoft host jQuery.

To use jQuery from Google or Microsoft, use one of the following:

Google CDN:

<head>

<script

src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

</head>

Microsoft CDN:

<head>

<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery 1.11.3.min.js"></script>

</head>

## 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

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

$(function(){  
  
   *// jQuery methods go here...*

});

## 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](http://www.w3schools.com/cssref/css_selectors.asp), 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:

$(document).ready(function(){

    $("button").click(function(){

        $("p").hide();

    });

});

## 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 HTML

element:

$("#test")

Example

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

$(document).ready(function(){

    $("button").click(function(){

       $("#test").hide();

    });

});

**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")

Example :

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

$(document).ready(function(){

$("button").click(function(){

$(".test").hide();

});

});

## 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

The term **"fires/fired"** is often used with events. Example: "The keypress event is fired, the

moment you press a key".

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 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!!

});

## Commonly Used jQuery Event Methods :

**$(document).ready()**

The $(document).ready() method allows us to execute a function when the

document is fully loaded. This event is already explained in the [jQuery](http://www.w3schools.com/jquery/jquery_syntax.asp)

[Syntax](http://www.w3schools.com/jquery/jquery_syntax.asp) chapter.

**click()**

The click() method attaches an event handler function to an HTML element.

The function is executed when the user clicks on the HTML element.

The following example says: When a click event fires on a <p> element; hide

the current <p> element:

**Example:**

$("p").click(function(){

    $(this).hide();

});

**dblclick()**

The dblclick() method attaches an event handler function to an HTML element.

The function is executed when the user double-clicks on the HTML element:

**Example:**

$("p").dblclick(function(){

$(this).hide();

});

**mouseenter() :**

The mouseenter() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer enters the HTML element:

**Example:**

$("#p1").mouseenter(function(){

    alert("You entered p1!");

});

**mouseleave() :**

The mouseleave() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer leaves the HTML element:

**Example:**

$("#p1").mouseleave(function(){

    alert("Bye! You now leave p1!");

});

**mousedown() :**

The mousedown() method attaches an event handler function to an HTML element.

The function is executed, when the left, middle or right mouse button is pressed down, while the

mouse is over the HTML element:

**Example:**

$("#p1").mousedown(function(){

    alert("Mouse down over p1!");

});

**mouseup() :**

The mouseup() method attaches an event handler function to an HTML element.

The function is executed, when the left, middle or right mouse button is

released, while the mouse is over the HTML element:

**Example:**

$("#p1").mouseup(function(){

    alert("Mouse up over p1!");

});

**hover() :**

The hover() method takes two functions and is a combination of the mouseenter() and

mouseleave() methods.

The first function is executed when the mouse enters the HTML element, and the second

function is executed when the mouse leaves the HTML element:

**Example:**

$("#p1").hover(function(){

    alert("You entered p1!");

},

function(){

    alert("Bye! You now leave p1!");

});

**focus()**

The focus() method attaches an event handler function to an HTML form field.

The function is executed when the form field gets focus:

**Example:**

$("input").focus(function(){

    $(this).css("background-color", "#cccccc");

});

**blur()**

The blur() method attaches an event handler function to an HTML form field.

The function is executed when the form field loses focus:

**Example:**

$("input").blur(function(){

    $(this).css("background-color", "#ffffff");

});

## The on() Method

The on() method attaches one or more event handlers for the selected

elements.

Attach a click event to a <p> element:

Attach multiple event handlers to a <p> element:

**Example:**

$("p").on({

mouseenter: function(){

        $(this).css("background-color", "lightgray");

    },

    mouseleave: function(){

        $(this).css("background-color", "lightblue");

    },

    click: function(){

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

    }

});

**Chapter 2 :**

**jQuery Effects:**

* jQuery Hide/show
* [jQuery Fade](http://www.w3schools.com/jquery/jquery_fade.asp)
* [jQuery Slide](http://www.w3schools.com/jquery/jquery_slide.asp)
* [jQuery Animate](http://www.w3schools.com/jquery/jquery_animate.asp)
* [jQuery stop()](http://www.w3schools.com/jquery/jquery_stop.asp)
* [jQuery Callback](http://www.w3schools.com/jquery/jquery_callback.asp)
* [jQuery Chaining](http://www.w3schools.com/jquery/jquery_chaining.asp)

## jQuery hide() and show() :

With jQuery, you can hide and show HTML elements with the hide() and show() methods:

**Syntax:**

$(selector).hide(speed,callback);  
  
$(selector).show(speed,callback);

The optional speed parameter specifies the speed of the hiding/showing, and can take the

following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the hide() or show()

method completes (you will learn more about callback functions in a later chapter).

### Example :

$("#hide").click(function(){

   $("p").hide();

});

$("#show").click(function(){

    $("p").show();

});

The following example demonstrates the speed parameter with hide():

**Example :**

$("button").click(function(){

    $("p").hide(1000);

});

## jQuery toggle() :

## With jQuery, you can toggle between the hide() and show() methods with the

## toggle() method.

**Syntax:**

$(*selector*).toggle(*speed,callback*);

The optional speed parameter can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after toggle() completes.

Shown elements are hidden and hidden elements are shown:

**Example:**

$("button").click(function(){

    $("p").toggle();

});

## jQuery Fading Methods :

With jQuery you can fade an element in and out of visibility.

jQuery has the following fade methods:

* fadeIn()
* fadeOut()
* fadeToggle()
* fadeTo()

## jQuery fadeIn() Method :

The jQuery fadeIn() method is used to fade in a hidden element.

**Syntax:**

$(selector).fadeIn(speed,callback);

The optional speed parameter specifies the duration of the effect. It can take the following

values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the fading completes.

The following example demonstrates the fadeIn() method with different parameters:

Example :

$("button").click(function(){

    $("#div1").fadeIn();

    $("#div2").fadeIn("slow");

    $("#div3").fadeIn(3000);

});

## jQuery fadeOut() Method :

The jQuery fadeOut() method is used to fade out a visible element.

**Syntax:**

$(selector).fadeOut(speed,callback);

The optional speed parameter specifies the duration of the effect. It can take the following

values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the fading completes.

The following example demonstrates the fadeOut() method with different parameters:

Example :

$("button").click(function(){

    $("#div1").fadeOut();

    $("#div2").fadeOut("slow");

    $("#div3").fadeOut(3000);

}):

## jQuery fadeToggle() Method :

The jQuery fadeToggle() method toggles between the fadeIn() and fadeOut() methods.

If the elements are faded out, fadeToggle() will fade them in.

If the elements are faded in, fadeToggle() will fade them out.

**Syntax:**

$(*selector*).fadeToggle(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following

values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the fading completes.

The following example demonstrates the fadeToggle() method with different parameters:

**Example :**

$("button").click(function(){

    $("#div1").fadeToggle();

    $("#div2").fadeToggle("slow");

    $("#div3").fadeToggle(3000);

});

## jQuery fadeTo() Method:

The jQuery fadeTo() method allows fading to a given opacity (value between 0 and 1).

**Syntax:**

$(selector).fadeTo(speed,opacity,callback);

The required speed parameter specifies the duration of the effect. It can take the following

values: "slow", "fast", or milliseconds.

The required opacity parameter in the fadeTo() method specifies fading to a given opacity

(value between 0 and 1).

The optional callback parameter is a function to be executed after the function completes.

The following example demonstrates the fadeTo() method with different parameters:

**Example :**

$("button").click(function(){

    $("#div1").fadeTo("slow", 0.15);

    $("#div2").fadeTo("slow", 0.4);

    $("#div3").fadeTo("slow", 0.7);

});

# jQuery Effects – Sliding :

## jQuery Sliding Methods:

With jQuery you can create a sliding effect on elements.

jQuery has the following slide methods:

* slideDown()
* slideUp()
* slideToggle()

## jQuery slideDown() Method :

The jQuery slideDown() method is used to slide down an element.

**Syntax:**

$(selector).slideDown(speed,callback);

The optional speed parameter specifies the duration of the effect. It can take the following

values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the sliding completes.

The following example demonstrates the slideDown() method:

**Example:**

$("#flip").click(function(){

$("#panel").slideDown();

});

## jQuery slideUp() Method :

The jQuery slideUp() method is used to slide up an element.

**Syntax:**

$(*selector*).slideUp(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following

values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the sliding completes.

The following example demonstrates the slideUp() method:

**Example :**

$("#flip").click(function(){

   $("#panel").slideUp();

});

## jQuery slideToggle() Method :

The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.

If the elements have been slid down, slideToggle() will slide them up.

If the elements have been slid up, slideToggle() will slide them down.

$(*selector*).slideToggle(*speed,callback*);

The optional speed parameter can take the following values: "slow", "fast", milliseconds.

The optional callback parameter is a function to be executed after the sliding completes.

The following example demonstrates the slideToggle() method:

**Example :**

$("#flip").click(function(){

    $("#panel").slideToggle();

});

# jQuery Effects – Animation :

## jQuery Animations - The animate() Method :

The jQuery animate() method is used to create custom animations.

**Syntax:**

$(*selector*).animate({*params*}*,speed,callback*);

The required params parameter defines the CSS properties to be animated.

The optional speed parameter specifies the duration of the effect. It can take the following

values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the animation completes.

The following example demonstrates a simple use of the animate() method; it moves a

<div> element to the right, until it has reached a left property of 250px:

**Example:**

$("button").click(function(){

    $("div").animate({left: '250px'});

});

## jQuery animate() - Manipulate Multiple Properties :

Notice that multiple properties can be animated at the same time:

**Example:**

$("button").click(function(){

    $("div").animate({

        left: '250px',

        opacity: '0.5',

        height: '150px',

        width: '150px'

    });

});

## jQuery animate() - Using Relative Values :

It is also possible to define relative values (the value is then relative to the element's

current value). This is done by putting += or -= in front of the value:

**Example :**

$("button").click(function(){

    $("div").animate({

        left: '250px',

        height: '+=150px',

        width: '+=150px'

    });

});

## jQuery animate() - Using Pre-defined Values :

You can even specify a property's animation value as "show", "hide", or "toggle":

**Example:**

$("button").click(function(){

    $("div").animate({

        height: 'toggle'

    });

});

## jQuery animate() - Uses Queue Functionality :

By default, jQuery comes with queue functionality for animations.

This means that if you write multiple animate() calls after each other, jQuery creates an

"internal" queue with these method calls. Then it runs the animate calls ONE by ONE.

So, if you want to perform different animations after each other, we take advantage of the queue

functionality:

**Example:**

$("button").click(function(){

    var div = $("div");

    div.animate({height: '300px', opacity: '0.4'}, "slow");

    div.animate({width: '300px', opacity: '0.8'}, "slow");

    div.animate({height: '100px', opacity: '0.4'}, "slow");

    div.animate({width: '100px', opacity: '0.8'}, "slow");

});

# jQuery Stop Animations :

## jQuery stop() Method

The jQuery stop() method is used to stop an animation or effect before it is finished.

The stop() method works for all jQuery effect functions, including sliding, fading and custom

animations.

**Syntax:**

$(*selector*).stop(*stopAll,goToEnd*);

The optional stopAll parameter specifies whether also the animation queue should be cleared or

not. Default is false, which means that only the active animation will be stopped, allowing any

queued animations to be performed afterwards.

The optional goToEnd parameter specifies whether or not to complete the current animation

immediately. Default is false.

So, by default, the stop() method kills the current animation being performed on the selected

element.

The following example demonstrates the stop() method, with no parameters:

**Example:**

$("#stop").click(function(){

    $("#panel").stop();

});

# jQuery Callback Functions :

## 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*);**

**Example:**

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");

    });

});

### Example without Callback:

### $("button").click(function(){

### $("p").hide(1000);

### alert("The paragraph is now hidden");

### });

# jQuery – Chaining :

## 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);