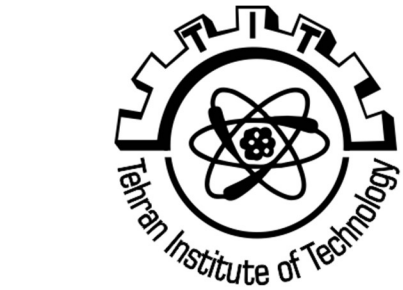


JQUERY & AJAX

Tehran Institute of Technology

Zahra Mansoori
z.mansoori@gmail.com
http://tarsimm.com

Contents

1. Introduction

jQuery is a JavaScript library that makes the use of JavaScript easier and simpler. There are many people who still think that jQuery is a software to be installed on the computer or a new programming language, but it is not. It is just a library that is very fast, light-weight and feature-rich. By using jQuery, you can implement much critical functionalities that you would have to write hundreds of lines of JavaScript with one or two lines of code. This will make coding so faster and also result in simple and cleaner code. However, this does not mean that jQuery is a replacement to JavaScript. It simply enhances the productivity of the developer by writing less and doing more.

1.1. jQuery Referencing

Being a JavaScript library, you need to reference jQuery before using. You can reference it either offline or online. If your Internet connection is not that stable, you can download jQuery from the official website

http://jquery.com/download/

and add reference to the required file in your .html or .js file you plan to use jQuery. If you visit the site, you could find two versions of jQuery, jQuery 1.x and jQuery 2.x. Though the core functionality of jQuery remains the same in both these versions, it is always better to download the latest version of jQuery. Obviously, the latest version will have more enhanced features.

Under each version, you could find a production version and a development version. These versions are exactly the same when it comes to functionalities. The only difference is in their performance. If you download both of these files and see the file size, you could find a huge difference. Suppose you download the production and development versions of jQuery 2.x, you could find that the file size of development version is 241 KB where as the size of production version is only 84 KB. Hence, if you are referencing the production version, your page will load lot faster than referencing development version.

Suppose you downloaded the jQuery file, say jquery.js, and saved it inside the folder D:\jQuery and you want to add a reference to it from D:\MySite\index.html file, then inside the <head> section of the index.html file, you should have the following line of code:

```
<script src="D:\jQuery\jquery.js"></script>
```

Of course, the path will be different based on where you have saved the jQuery file.

To reference it online, just add the following lines of code:

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
```

Example.1. Sample Code

```
<html>
	<head>
		<title>Sample</title>
		<script src="jquery.js"></script>
		<script>
		 $(document).ready(function() {
			$("#btnClick" ).click(function() {
				alert("You clicked me!!" );
			});
		});
		</script>
	</head>
	<body>
		 <button id="btnClick">Click Me!!</button>
	</body>
</html>
```

2. Basic Concepts

2.1. jQuery Basics

```
$(document).ready()
```

You cannot safely manipulate a page until the document is "ready." jQuery detects this state of readiness for you; code included inside $(document).ready() will only run once the page is ready for JavaScript code to execute.

Example. 2: A $(document).ready() block

```
$(document).ready(function() {
        console.log('ready!');
});
```

There is a shorthand for $(document).ready() that you will sometimes see; however, I recommend against using it if you are writing code that people who aren't experienced with jQuery may see.

Example. 3: Shorthand for $(document).ready()

```
$(function() {
        console.log('ready!');
});
```

You can also pass a named function to $(document).ready() instead of passing an anonymous function.

Example. 4: Passing a named function instead of an anonymous function

```
function readyFn() {
        // code to run when the document is ready
}
$(document).ready(readyFn);
```

2.2. Selecting Elements

The most basic concept of jQuery is to "select some elements and do something with them." jQuery supports most CSS3 selectors, as well as some non-standard selectors.

Selector	*Example*	*Selects*
*	$("*")	All elements

#id	$("#lastname")	The element with id="lastname"
.class	$(".intro")	All elements with class="intro"
.class, .class	$(".intro,.demo")	All elements with the class "intro" or "demo"
element	$("p")	All <p> elements
el1, el2, el3	$("h1,div,p")	All <h1>, and <p> elements
:first	$("p:first")	The first <p> element
:last	$("p:last")	The last <p> element
:even	$("tr:even")	All even <tr> elements
:odd	$("tr:odd")	All odd <tr> elements
:first-child	$("p:first-child")	All <p> elements that are the first child of their parent
:first-of-type	$("p:first-of-type")	All <p> elements that are the first <p> element of their parent
:last-child	$("p:last-child")	All <p> elements that are the last child of their parent
:last-of-type	$("p:last-of-type")	All <p> elements that are the last <p> element of their parent
:nth-child(n)	$("p:nth-child(2)")	All <p> elements that are the 2nd child of their parent
:nth-last-child(n)	$("p:nth-last-child(2)")	All <p> elements that are the 2nd child of their parent, counting from the last child
:nth-of-type(n)	$("p:nth-of-type(2)")	All <p> elements that are the 2nd <p> element of their parent
:nth-last-of-type(n)	$("p:nth-last-of-type(2)")	All <p> elements that are the 2nd <p> element of their parent, counting from the last child
:only-child	$("p:only-child")	All <p> elements that are the only child of their parent
:only-of-type	$("p:only-of-type")	All <p> elements that are the only child, of its type, of their parent
parent > child	$("div > p")	All <p> elements that are a direct child of a element
parent descendant	$("div p")	All <p> elements that are descendants of a element

element + next	$("div + p")	The <p> element that are next to each elements
element ~ siblings	$("div ~ p")	All <p> elements that appear after the element
*:eq(*index*)*	$("ul li:eq(3)")	The fourth element in a list (index starts at 0)
*:gt(*no*)*	$("ul li:gt(3)")	List elements with an index greater than 3
*:lt(*no*)*	$("ul li:lt(3)")	List elements with an index less than 3
*:not(*selector*)*	$("input:not(:empty)")	All input elements that are not empty
:header	$(":header")	All header elements <h1>, <h2> ...
:animated	$(":animated")	All animated elements
:focus	$(":focus")	The element that currently has focus
*:contains(*text*)*	$(":contains('Hello')")	All elements which contains the text "Hello"
*:has(*selector*)*	$("div:has(p)")	All elements that have a <p> element
:empty	$(":empty")	All elements that are empty
:parent	$(":parent")	All elements that are a parent of another element
:hidden	$("p:hidden")	All hidden <p> elements
:visible	$("table:visible")	All visible tables
:root	$(":root")	The document's root element
*:lang(*language*)*	$("p:lang(de)")	All <p> elements with a lang attribute value starting with "de"
*[*attribute*]*	$("[href]")	All elements with a href attribute
*[*attribute=value*]*	$("[href='default.htm']")	All elements with a href attribute value equal to "default.htm"
*[*attribute*!=*value*]*	$("[href!='default.htm']")	All elements with a href attribute value not equal to "default.htm"
*[*attribute*$=*value*]*	$("[href$='.jpg']")	All elements with a href attribute value ending with ".jpg"

[attribute|=value]	$("[title|='Tomorrow']")	All elements with a title attribute value equal to 'Tomorrow', or starting with 'Tomorrow' followed by a hyphen
[attribute^=value]	$("[title^='Tom']")	All elements with a title attribute value starting with "Tom"
[attribute~=value]	$("[title~='hello']")	All elements with a title attribute value containing the specific word "hello"
[attribute=value]*	$("[title*='hello']")	All elements with a title attribute value containing the word "hello"
:input	$(":input")	All input elements
:text	$(":text")	All input elements with type="text"
:password	$(":password")	All input elements with type="password"
:radio	$(":radio")	All input elements with type="radio"
:checkbox	$(":checkbox")	All input elements with type="checkbox"
:submit	$(":submit")	All input elements with type="submit"
:reset	$(":reset")	All input elements with type="reset"
:button	$(":button")	All input elements with type="button"
:image	$(":image")	All input elements with type="image"
:file	$(":file")	All input elements with type="file"
:enabled	$(":enabled")	All enabled input elements
:disabled	$(":disabled")	All disabled input elements
:selected	$(":selected")	All selected input elements
:checked	$(":checked")	All checked input elements

Following are a few examples of common selection techniques.

Example. 5: Selecting elements by ID

```
$('#myId');      // note IDs must be unique per page
```

Example. 6: Selecting elements by class name

```
$('div.myClass');      // performance improves if you specify element type
```

Example. 7: Selecting elements by attribute

```
$('input[name=first_name]');      // beware, this can be very slow
```

Example. 8: Selecting elements by compound CSS selector

```
$('#contents ul.people li');
```

Example. 9: Pseudo-selectors

```
$('a.external:first');
$('tr:odd');
$('#myForm :input');     // select all input-like elements in a form
$('div:visible');
$('div:gt(2)');     // all except the first three divs
$('div:animated');     // all currently animated divs
```

2.3. Saving Selections

Every time you make a selection, a lot of code runs, and jQuery doesn't do caching of selections for you. If you've made a selection that you might need to make again, you should save the selection in a variable rather than making the selection repeatedly.

Example. 10: Storing selections in a variable

```
var $divs = $('div');
```

Example. 11: Pseudo-selectors

```
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
<script src="js/jquery-3.5.1.min.js"></script>

</head>
<body>
	<form action="targetPage.php">
		<label>Name:</label>
		<input type="text" id="name1" value="test1">
		<br>
		<label>Married</label>
		<input type="checkbox" id="maritalStatus1" checked>
		<br>
		<label>Employed</label>
		<input type="checkbox" id="employmentStatus1" checked>
		<br>
		<input type="button" id="button1" value="Click on me">
	</form>

	<script language="javascript">
		$(document).ready(function(){
			$name = $("#name1");
			$maritalStatus = $("#maritalStatus1");
			$AllCheckedCheckBoxes = $(":checked");
			$("#button1").click(function(){
				alert("Name: " + $name.val() + " & Marital Status: " +
				$maritalStatus.val() );
				$AllCheckedCheckBoxes.each(function() {
					alert( "The person is: " + $(this).prev().text() );
				  });
			});
		});
	</script>

</body>
</html>
```

2.4. Traversing

In jQuery, traversing means moving through or over the HTML elements to find, filter or select a particular or entire element.
Based on the traversing purposes following methods are Categorized as follows:

2.4.1. Tree Traversing

Ancestors:

parent()

it gives parent element of specified selector

```
$(selector).parent();
```

Example.12: parent()

```
$(document).ready(function(){
        $("span"). parent().css({"color": "red", "border": "2px solid red"});
});
```

parents()

it gives all ancestor elements of the specified selector.

```
$(selector).parents();
```

Example.13: parents()

```
$(document).ready(function(){
        $("span").parents().css({"color": "red", "border": "2px solid red"});
});
```

parentsUntil()

it gives all ancestor elements between specified selector and arguments.

```
$(selector).parentsUntil(selector, filter element)

$(selector).parentsUntil(element, filter element)
```

Example.14: ParentUntil()

```
$(document).ready(function(){
        $("span").parentsUntil("div").css({"color": "red", "border": "2px solid red"});
});
```

offsetParent()

it gives the first positioned parent element of specified selector.

Tip: An element can be positioned with jQuery, or with the CSS position property (relative, absolute, or fixed).

```
$(selector).offsetParent();
```

Example.15: Set the background color of the closest positioned parent element of the <p> element:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("p").offsetParent().css("background-color", "red");
		  });
		});
		</script>
	</head>
	<body>

		<button>Set background-color</button>

		<div style="border:1px solid black; width:70%; position:absolute;
		left:10px; top:50px">
		<div style="border:1px solid black; margin:50px; background-
		color:yellow">

		<p>Click button to set the background color of the first positioned
		parent element of this paragraph.</p>

		</div></div>

	</body>
</html>
```

closest()

it gives the first ancestor of the specified selector.

```
$(selector).closest(selector);
$(selector).closest(selector, context);
$(selector).closest(selection);
$(selector).closest(element);
```

Example.16: closest()

```
$(document).ready(function(){
        $("span").closest("ul").css({"color": "red", "border": "2px solid red"});
});
```

Descendants:

children()

it gives the children of each selected elements, optionally filtered by a selector.

```
$(selector).children();
```

Example.17: children()

```
$(document).ready(function(){
        $("ul").children().css({"color": "red", "border": "2px solid red"});
});
```

find()

it gives descendant elements of specified elements, filtered by a selector, jQuery object, or element.

```
$(selector).find('selector to find');
```

Example.18: find()

```
$(document).ready(function(){
        $("ul").find("span").css({"color": "red", "border": "2px solid red"});
});
```

Siblings:

siblings()

it gives all siblings of the specified selector.

```
$(selector).siblings();
```

Example.19: siblings()

```
$(document).ready(function(){
        $("li.start").siblings().css({"color": "red", "border": "2px solid red"});
});
```

next()

it gives the next sibling element of the specified selector.

```
$(selector).next();
```

Example.20: next()

```
$(document).ready(function(){
        $("li.start").next().css({"color": "red", "border": "2px solid red"});
});
```

nextAll()

it gives all next sibling elements of the specified selector.

```
$(selector).nextAll();
```

Example.21: nextAll()

```
$(document).ready(function(){
        $("li.start").nextAll().css({"color": "red", "border": "2px solid red"});
});
```

nextUntil()

it gives all next sibling elements between specified selector and arguments.

```
$(selector).nextUntil();
```

Example.22: nextUntil()

```
$(document).ready(function(){
        $("li.start").nextUntil("li.stop").css({"color": "red", "border": "2px solid red"});
});
```

prev()

it gives the previous sibling element of the specified selector.

```
$(selector).prev(selector);
$(selector).prev()
```

Example.23: prev()

```
$(document).ready(function(){
        $("li.start").prev().css({"color": "red", "border": "2px solid red"});
});
```

prevAll()

it gives all previous sibling elements of the specified selector.

```
$(selector).prevAll(selector, filter element)
$(selector).prevAll(element, filter element)
```

Example.24: prevAll()

```
$(document).ready(function(){
        $("li.start").prevAll().css({"color": "red", "border": "2px solid red"});
});
```

prevUntil()

it gives all previous sibling elements between specified selector and arguments.

```
$(selector).prevUntil(selector, filter element)
$(selector).prevUntil(element, filter element)
```

Example.25: prevUntil()

```
$(document).ready(function(){
        $("li.start").prevUntil("li.stop").css({"color": "red", "border": "2px solid red"});
});
```

2.4.2. Filtering

first()

it gives the first element of the specified selector.

```
$(selector).first();
```

Example.26: first(), Select the first <p> element inside the last element:

```
$("div p").first()
```

last()

it gives the last element of the specified selector.

```
$(selector).last();
```

Example.27: last(), Select the last <p> element inside the last element:

```
$("div p").last()
```

eq()

it gives an element with a specific index number of the specified selector.

```
$(selector).eq(index);
$(selector).eq( indexFromEnd );
```

Example.28: eq(), Select the second <p> element (index number 1)::

```
$("p").eq(1).css("background-color", "yellow");
```

filter()

it remove/detect an elements that are matched with specified selector.

```
.filter( selection )
```

Example.29: filter(),Change the color of all divs; then add a border to those with a "middle" class.

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>filter demo</title>
		<style>
			div {
			  width: 60px;
			  height: 60px;
			  margin: 5px;
			  float: left;
			  border: 2px white solid;
			}
		</style>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>

		<div></div>
		<div class="middle"></div>
		<div class="middle"></div>
		<div class="middle"></div>
		<div class="middle"></div>
		<div></div>

		<script>
		$(document).ready(function(){
			$( "div" )
			  .css( "background", "#c8ebcc" )
			  .filter( ".middle" )
			  .css( "border-color", "red" );
		});
		</script>

	</body>
</html>
```

has()

it gives all elements that have one or more elements within, that are matched with specified selector.

```
$(selector).has(selector);
```

Example.30: Return all <p> elements that have a element inside of them:

```
$("p").has("span")
```

is()

Returns true or false, not an object. it checks if one of the specified selectors is matched with arguments.

```
.is( selector )
.is( function )
.is( selection )
.is( elements )
```

Example.31: If the parent of <p> is a element, alert some text:

```
if ($("p").parent().is("div")) {
        alert("Parent of p is div");
}
```

slice()

it selects a subset of specified selector based on its argument index or by start and stop value.

```
$(selector).slice(start, end );
$(selector).slice(start);
```

Example.32: Start the selection of <p> elements from the <p> element with index number 2 to the end:

```
$("p").slice(2).css("background-color", "yellow");
```

2.4.3. Miscellaneous Traversing

add()

it add all elements to set of matched elements to manipulate them at the same time.

```
$(selector).add(selector to add);
```

Example.33: Add <p> and elements to an existing group of elements (<h1>):

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		>
		</script>
		<script>
		$(document).ready(function(){
			$("h1").add("p").add("span").css("background-color",
			"yellow");
		});
		</script>

	</head>
	<body>
		<h1>Welcome</h1>
		<p>A p element.</p>
		<p>Another p element.</p>

		<span>A span element.</span>
		<span>A span element.</span>
		<br><br>

		<div>This example adds the same css style for both p and span
		elements, as the existing h1 element.</div>
	</body>
</html>
```

contents()

it gives all direct children, including text and comment nodes, of the specified selector.

```
$(selector).contents();
```

Example.34: Find all the text nodes inside a element and wrap them with a <b> element:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		>
		</script>
		<script>
		$(document).ready(function(){
			$("button").click(function(){
				$("div").contents().filter("em").wrap("<b/>");
			});
		});
		</script>
	</head>
	<body>
		<div><em>Hello world! What a beautiful day!</em></div>

		<p>In this example, by clicking on the button, we search for all the text
		nodes inside the div element and wrap them with a b element.</p>

		<button>Find all text nodes in div and wrap them</button><br>
	</body>
</html>
```

not()

it gives all elements that do not match with specified selector.

```
$(selector).not(selector);
```

Example.35: Return all <p> elements that do **not** have the class name "intro":

```
$("p").not(".intro")
```

2.4.4. Collection Manipulation

each()

it iterates over the DOM elements and execute call back function

Example.37: Iterate over three divs and sets their color property:

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>each demo</title>
		<style>
			div {
			  color: red;
			  text-align: center;
			  cursor: pointer;
			  font-weight: bolder;
			  width: 300px;
			}
		</style>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>

		<div>Click here</div>
		<div>to iterate through</div>
		<div>these divs.</div>

		<script>
		$( document.body ).click(function() {
			$( "div" ).each(function( i ) {
				if ( $(this).style.color !== "blue" ) {
					$(this).style.color = "blue";
				} else {
					$(this).style.color = "";
				}
			});
		});
		</script>

	</body>
</html>
```

3. Working with Selections

Once you have a selection, you can call methods on the selection. Methods generally come in two different flavors: getters and setters. Getters return a property of the first selected element; setters set a property on all selected elements.

3.1. Chaining

If you call a method on a selection and that method returns a jQuery object, you can continue to call jQuery methods on the object without pausing for a semicolon.

Example. Chaining

```
$('#content').find('h3').eq(2).html('new text for the third h3!');
```

If you are writing a chain that includes several steps, you (and the person who comes after you) may find your code more readable if you break the chain over several lines.

Example.38: Formatting chained code

```
$('#content')
.find('h3')
.eq(2)
.html('new text for the third h3!');
```

3.2. Getters & Setters

jQuery “overloads” its methods, so the method used to set a value generally has the same name as the method used to get a value. When a method is used to set a value, it is called a setter. When a method is used to get (or read) a value, it is called a getter. Setters affect all elements in a selection; getters get the requested value only for the first element in the selection.

Example.39: Setter Function

```
$('h1').html('hello world');
```

Example.40: Getter Function

```
$('h1').html();
```

Setters return a jQuery object, allowing you to continue to call jQuery methods on your selection; getters return whatever they were asked to get, meaning you cannot continue to call jQuery methods on the value returned by the getter.

3.3. CSS, Styling, & Dimensions

jQuery includes a handy way to get and set CSS properties of elements.

Example.41: Getting CSS properties

```
$('h1').css('fontSize'); // returns a string such as "19px"
```

Example.42: Setting CSS properties

```
$('h1').css('fontSize', '100px'); // setting an individual property
$('h1').css({ 'fontSize' : '100px', 'color' : 'red' }); // setting multiple properties
```

3.3.1. Using CSS Classes for Styling

As a getter, the $.fn.css method is valuable; however, it should generally be avoided as a setter in production-ready code, because you don't want presentational information in your JavaScript.

Instead, write CSS rules for classes that describe the various visual states, and then simply change the class on the element you want to affect.

Classes can also be useful for storing state information about an element, such as indicating that an element is selected.

addClass()

adds one or more class names to the selected elements

Example.43: Addclass() example

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></s
		cript>
		<script>
			$(document).ready(function(){
			  $("button").click(function(){
			    $("p:first").addClass("intro");
			  });
		});
		</script>

		<style>
		.intro {
		  font-size: 150%;
		  color: red;
		}
		</style>

	</head>
	<body>

		<h1>This is a heading</h1>

		<p>This is a paragraph.</p>
		<p>This is another paragraph.</p>

		<button>Add a class name to the first p element</button>

	</body>
</html>
```

removeClass()

adds one or more class names to the selected elements

Example.44: removeClass() example

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></s
		cript>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("p").removeClass("intro");
		  });
		});
		</script>

		<style>
		.intro {
		  font-size: 120%;
		  color: red;
		}
		</style>

	</head>
	<body>

		<h1>This is a heading</h1>

		<p class="intro">This is a paragraph.</p>
		<p class="intro">This is another paragraph.</p>

		<button>Remove the "intro" class from all p elements</button>

	</body>
</html>
```

toggleClass()

toggles between adding and removing one or more class names from the selected elements.

Example.45: toggleClass() example

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></s
		cript>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("p").toggleClass("main");
		  });
		});
		</script>

		<style>
		.main {
		  font-size: 120%;
		  color: red;
		}
		</style>

	</head>
	<body>

		<button>Toggle class "main" for p elements</button>

		<p>This is a paragraph.</p>
		<p>This is another paragraph.</p>
		<p><b>Note:</b> Click the button more than once to see the toggle
		effect.</p>

	</body>
</html>
```

3.3.2. Dimensions

jQuery offers a variety of methods for obtaining and modifying dimension and position information about an element.

Example.46: Working with classes

```
$('h1').width('50px'); // sets the width of all H1 elements
$('h1').width();          // gets the width of the first H1

$('h1').height('50px'); // sets the height of all H1 elements
$('h1').height();          // gets the height of the first H1

$('h1').position();      // returns an object containing position
                         // information for the first H1 relative to
                         // its "offset (positioned) parent"
```

3.4. Manipulating Elements

Once you've made a selection, the fun begins. You can change, move, remove, and clone elements. You can also create new elements via a simple syntax.

3.4.1. Getting and Setting Information about Elements

There are many ways to change an existing element. Among the most common tasks is changing the inner HTML or attribute of an element. jQuery offers simple, cross-browser methods for these sorts of manipulations. You can also get information about elements using many of the same methods in their getter incarnations. For more information on getters and setters, see the Working with Selections section. Here are a few methods you can use to get and set information about elements:

html()

Get or set the HTML contents

```
$(selector).html("string"); //setter function
$(selector).html();         //getter function
```

Example.47: Html() Getter Function

```
<script>
$(document).ready(function(){
   // Get HTML contents of first selected paragraph
   $(".btn-one").click(function(){
      var str = $("p").html();
      alert(str);
   });

   // Get HTML contents of an element with ID container
   $(".btn-two").click(function(){
      var str = $("#container").html();
      alert(str);
   });
});
</script>
```

Example.48: Html() Setter Function

```
<script>
$(document).ready(function(){
   // Set HTML contents for document's body
   $("button").click(function(){
      $("body").html("<p>Hello World!</p>");
   });
});
</script>
```

text()

Get or set the text contents; HTML will be stripped.

Example.49: Text() Getter Function

```
<script>
$(document).ready(function(){
   // Get combined text contents of all paragraphs
   $(".btn-one").click(function(){
      var str = $("p").text();
      alert(str);
   });

   // Get text contents of the first paragraph
   $(".btn-two").click(function(){
     var str = $("p:first").text();
     alert(str);
   });
});
</script>
```

Example.50: text() Setter Function

```
<script>
$(document).ready(function(){
   // Set text contents of all paragraphs
   $(".btn-one").click(function(){
      $("p").text("This is demo text.");
   });

   // Set text contents of the first paragraph
   $(".btn-two").click(function(){
      $("p:first").text("This is another demo text.");
   });
});
</script>
```

attr()

Get or set the value of the provided attribute

Example.51: attr() Getter Function

```
<script>
$(document).ready(function(){
   // Get href attribute value of first selected hyperlink
   $(".btn-one").click(function(){
      var str = $("a").attr("href");
      alert(str);
   });

   // Get alt attribute value of an image with ID sky
   $(".btn-two").click(function(){
      var str = $("img #sky").attr("alt");
      alert(str);
   });
});
</script>
```

Example.52: attr() Setter Function

```
<script>
$(document).ready(function(){
   // Check all the checkboxes
   $("button").click(function(){
      $('input[type="checkbox"]').attr("checked", "checked");
   });
});
</script>
```

Example.53: attr() Data Function

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>jQuery Get the data-id Attribute</title>
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<script>
$(document).ready(function(){
   $(".gallery li").on("click", function(){
      var dataId = $(this).attr("data-id");
      alert("The data-id of clicked item is: " + dataId);
   });
});
</script>
</head>
<body>
   <ul class="gallery">
      <li data-id="1">
         Li Id 1
      </li>
      <li data-id="2">
         Li Id 2
      </li>
      <li data-id="3">
         Li Id 3
      </li>
      <li data-id="4">
         Li Id 4
      </li>
   </ul>
</body>
</html>
```

width()

Get or set the width in pixels of the first element in the selection as an integer

Example.53: width() Getter Function

```
<!DOCTYPE html>
<html>
	<head>
	<script
	src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
	</script>

	<script>
		$(document).ready(function(){
			$("button").click(function(){
				alert("Width of div: " + $("div").width());
			});
		});
	</script>
	</head>
	<body>

		<div style="height:100px; width:300px; padding:10px; margin:3px;
		border:1px solid blue; background-color:lightblue;"></div><br>

		<button>Display the width of div</button>

	</body>
</html>
```

Example.54: width() Setter Function

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>width demo</title>
		<style>
		div {
		  width: 70px;
		  height: 50px;
		  float: left;
		  margin: 5px;
		  background: red;
		  cursor: pointer;
		}
		.mod {
		  background: blue;
		  cursor: default;
		}
		</style>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>

		<div>d</div>
		<div>d</div>
		<div>d</div>
		<div>d</div>
		<div>d</div>

		<script>
		$(document).ready(function(){
			$modWidth = 50;
			$( "div" ).one( "click", function() {
				$( this ).width( modWidth ).addClass( "mod" );
				$modWidth -= 8;
			});
		});
		</script>

	</body>
</html>
```

height()

Get or set the height in pixels of the first element in the selection as an integer

Example.55: height() Getter Function

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>height demo</title>
		<style>
			body {
			  background: yellow;
			}
			button {
			  font-size: 12px;
			  margin: 2px;
			}
			p {
			  width: 150px;
			  border: 1px red solid;
			}
			div {
			  color: red;
			  font-weight: bold;
			}
		</style>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>

		<button id="getp">Get Paragraph Height</button>
		<button id="getd">Get Document Height</button>
		<button id="getw">Get Window Height</button>

		<div> </div>
		<p>
		 Sample paragraph to test height
		</p>

		<script>
		$(document).ready(function(){
			function showHeight( element, height ) {
				$( "div" ).text( "The height for the " + element + " is "
				+ height + "px." );
			}
			$( "#getp" ).click(function() {
				showHeight( "paragraph", $( "p" ).height() );
			});
```

```
                $( "#getd" ).click(function() {
                        showHeight( "document", $( document ).height() );
                });
                $( "#getw" ).click(function() {
                        showHeight( "window", $( window ).height() );
                });
        });
        </script>

    </body>
</html>
```

Example.56: height() Setter Function

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>height demo</title>
		<style>
		div {
		  width: 50px;
		  height: 70px;
		  float: left;
		  margin: 5px;
		  background: rgb(255,140,0);
		  cursor: pointer;
		}
		</style>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>

		<div></div>
		<div></div>
		<div></div>
		<div></div>
		<div></div>

		<script>
		$(document).ready(function(){
			$( "div" ).one( "click", function() {
				$( this )
				.height( 30 )
				.css({
				   cursor: "auto",
				   backgroundColor: "green"
				 });
			});
		});
		</script>

	</body>
</html>
```

position()

Get an object with position information for the first element in the selection, relative to its first positioned ancestor. This is a getter only.

Example.57: position() Setter Function

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
			$("button").click(function(){
				var x = $("p").position();
				alert("Top position: " + x.top + " Left position: " +
				x.left);
			});
		});
		</script>
	</head>
	<body>

		<p>This is a paragraph.</p>
		<button>Return the top and left position of the p element</button>

	</body>
</html>
```

val()

Get or set the value of form elements.

Example.58: val() Getter Function

```
<script>
$(document).ready(function(){
   // Get value of a text input with ID name
   $("button.get-name").click(function(){
      var name = $('input[type="text"]#name').val();
      alert(name);
   });

   // Get value of a textarea with ID comment
   $("button.get-comment").click(function(){
      var comment = $("textarea#comment").val();
      alert(comment);
   });

   // Get value of a select box with ID city
   $("button.get-city").click(function(){
      var city = $("select#city").val();
      alert(city);
   });
});
</script>
```

Example.59: val() Setter Function

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("input:text").val("Glenn Quagmire");
		  });
		});
		</script>
	</head>
	<body>

		<p>Name: <input type="text" name="user"></p>

		<button>Set the value of the input field</button>

	</body>
</html>
```

3.5. Moving, Copying, and Removing Elements

In this tutorial you learn how to manipulate existing elements in the page.

Moving

wrap()

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

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. A copy of this structure will be wrapped around each of the elements in the set of matched elements. This method returns the original set of elements for chaining purposes.

Example.60: wrap() a new div around all of the paragraphs.

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>wrap demo</title>
		<style>
		div {
		  border: 2px solid blue;
		}
		p {
		  background: yellow;
		  margin: 4px;
		}
		</style>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>

		<p>Hello</p>
		<p>cruel</p>
		<p>World</p>

		<script>
		$(document).ready(function(){
			$( "p" ).wrap( "<div></div>" );
		});
		</script>

	</body>
</html>
```

Removing

empty()

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

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:

Example.61: empty() contents of a paragraphs.

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>empty demo</title>
		<style>
		p {
		  background: yellow;
		}
		</style>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>
		<p>
		 Hello, <span>Person</span> <em>and person</em>.
		</p>

		<button>Call empty() on above paragraph</button>

		<script>
			$(document).ready(function(){
				$( "button" ).click(function() {
					$( "p" ).empty();
				});
			});
		</script>
	</body>
</html>
```

remove()

Remove the set of matched elements from the DOM.

Similar to .empty(), the .remove() method takes elements out of the DOM. Use .remove() when you 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. To remove the elements without removing data and events, use .detach() instead.

Example.62: empty() contents of a paragraphs.

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>remove demo</title>
		<style>
		p {
		  background: yellow;
		  margin: 6px 0;
		}
		</style>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>
		<p>Hello</p>
		how are
		<p>you?</p>
		<button>Call remove() on paragraphs</button>

		<script>
		$(document).ready(function(){
			$( "button" ).click(function() {
				$( "p" ).remove();
			});
		});
		</script>
	</body>
</html>
```

detach()

Remove the set of matched elements from the DOM.

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.

Example.63: detach() function

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>detach demo</title>
		<style>
		p {
		  background: yellow;
		  margin: 6px 0;
		}
		p.off {
		  background: black;
		}
		</style>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>
		<p>Hello</p>
		  how are
		  <p>you?</p>
		  <button>Attach/detach paragraphs</button>

		  <script>
		  $(document).ready(function(){
			$( "p" ).click(function() {
				$( this ).toggleClass( "off" );
			});
			var p;
			$( "button" ).click(function() {
				if ( p ) {
					p.appendTo( "body" );
					p = null;
				} else {
					p = $( "p" ).detach();
				}
			});
		  });
		  </script>

	</body>
</html>
```

unwrap()

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

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

Example.64: detach() function

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>unwrap demo</title>
		<style>
		div {
		  border: 2px solid blue;
		}
		p {
		  background: yellow;
		  margin: 4px;
		}
		</style>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>
		<button>wrap/unwrap</button>
		<p>Hello</p>
		<p>cruel</p>
		<p>World</p>
		<script>
		$(document).ready(function(){
			$pTags = $( "p" );
			$( "button" ).click(function() {
				if ( $pTags.parent().is( "div" ) ) {
					$pTags.unwrap();
				} else {
					$pTags.wrap( "<div></div>" );
				}
			});
		});
		</script>

	</body>
</html>
```

Replacing

replaceWith()

Replace each element in the set of matched elements with the provided new content and return the set of elements that was removed.

The .replaceWith() method removes content from the DOM and inserts new content in its place with a single call.

Example.65: detach() function

```
<!doctype html>
<html lang="en">
	<head>
	 <meta charset="utf-8">
	 <title>replaceWith demo</title>
	 <style>
	 button {
	  display: block;
	  margin: 3px;
	  color: red;
	  width: 200px;
	 }
	 div {
	  color: red;
	  border: 2px solid blue;
	  width: 200px;
	  margin: 3px;
	  text-align: center;
	 }
	 </style>
	 <script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>

	<button>First</button>
	<button>Second</button>
	<button>Third</button>

	<script>
	$(document).ready(function(){
		$( "button" ).click(function() {
			$( this ).replaceWith( "<div>" + $( this ).text() + "</div>" );
		});
	});
	</script>

	</body>
</html>
```

replaceAll()

Replace each target element with the set of matched elements.

Example.66: replaceAll() function

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>replaceAll demo</title>
		<script src="https://code.jquery.com/jquery-3.5.0.js"></script>
	</head>
	<body>

		<p>Hello</p>
		<p>cruel</p>
		<p>World</p>

		<script>
		$(document).ready(function(){
			$( "<b>Paragraph. </b>" ).replaceAll( "p" );
		});
		</script>

	</body>
</html>
```

3.6. Adding New Elements

append()

Inserts content at the end of the selected elements

Example.67: append() function

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		 $("#btn1").click(function(){
		   $("p").append(" <b>Appended text</b>.");
		 });

		 $("#btn2").click(function(){
		   $("ol").append("<li>Appended item</li>");
		 });
		});
		</script>
	</head>
	<body>

		<p>This is a paragraph.</p>
		<p>This is another paragraph.</p>

		<ol>
		 <li>List item 1</li>
		 <li>List item 2</li>
		 <li>List item 3</li>
		</ol>

		<button id="btn1">Append text</button>
		<button id="btn2">Append list items</button>

	</body>
</html>
```

prepend()

inserts content AT THE BEGINNING of the selected HTML elements.

Example.68: prepend() function

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		 $("#btn1").click(function(){
		  $("p").prepend("<b>Prepended text</b>. ");
		 });
		 $("#btn2").click(function(){
		  $("ol").prepend("<li>Prepended item</li>");
		 });
		});
		</script>
	</head>
	<body>

		<p>This is a paragraph.</p>
		<p>This is another paragraph.</p>

		<ol>
		 <li>List item 1</li>
		 <li>List item 2</li>
		 <li>List item 3</li>
		</ol>

		<button id="btn1">Prepend text</button>
		<button id="btn2">Prepend list item</button>

	</body>
</html>
```

after()

inserts content AFTER the selected HTML elements.

before()

inserts content BEFORE the selected HTML elements.

Example.69: before() and after() function

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
			$("#btn1").click(function(){
				$("img").before("<b>Before</b>");
			});

			$("#btn2").click(function(){
				$("img").after("<i>After</i>");
			});
		});
		</script>
	</head>
	<body>

		<img src="/images/w3jquery.gif" alt="jQuery" width="100"
		height="140"><br><br>

		<button id="btn1">Insert before</button>
		<button id="btn2">Insert after</button>

	</body>
</html>
```

4. Events

In most web applications, the user does some action to perform an operation. For example, user clicks on save button to save the edited data in a web page. Here, clicking on the button is a user's action, which triggers click event and click event handler (function) saves data.

4.1. Event Methods

The jQuery library provides methods to handle DOM events. Most jQuery methods correspond to native DOM events.

The following table lists all jQuery methods and corresponding DOM events.

Category	*jQuery Method*	*DOM Event*
Form events	blur	onblur
	onchange	
	onfocus	
	onfocusin	
	onselect	
	onsubmit	
	keydown	onkeydown
	onkeypress	
	onkeyup	
Mouse events	click	onclick
	ondblclick	
	onmousedown	
	onmouseenter	
	onmouseleave	
	onmousemove	
	onmouseout	
	onmouseover	
	onmouseup	
Browser events	Error:	onerror()
	onresize	
	onscroll	
Document loading	Load:	onload
	onunload	

Example.70: Event handling

```
<!DOCTYPE html>
<html>
<head>
	<script
	src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js">
	</script>
	<script>
		$(document).ready(function () {

				$('#saveBtn').click(function () {
					alert('Save button clicked');
				});

		});
	</script>
</head>
<body>
	<h1>Demo: jQuery click() method</h1>

	<input type="button" value="Save" id="saveBtn" />
</body>
</html>
```

4.2. Attach an event handler

on()

Attach an event handler function for one or more events to the selected elements.

Example.71: On() method

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("p").on("click", function(){
		    alert("The paragraph was clicked.");
		  });
		});
		</script>
	</head>
	<body>

		<p>Click this paragraph.</p>

	</body>
</html>
```

one()

The one() method attaches one or more event handlers for the selected elements, and specifies a function to run when the event occurs.

When using the one() method, the event handler function is only run **ONCE** for each element.

Example.72: One() method

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("p").one("click", function(){
		    $(this).animate({fontSize: "+=6px"});
		  });
		});
		</script>
	</head>
	<body>

		<p>This is a paragraph.</p>
		<p>This is another paragraph.</p>
		<p>Click any p element to increase its text size. The event will only
		trigger once for each p element.</p>

	</body>
</html>
```

off()

The off() method is most often used to remove event handlers attached with the on() method.

Example.73: Off() Remove the click event for all <p> elements:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		 $("p").on("click", function(){
		   $(this).css("background-color", "pink");
		 });



		 $("button").click(function(){
		  $("p").off("click");
		 });
		});
		</script>
	</head>
	<body>

		<p>Click this paragraph to change its background color.</p>
		<p>Click the button below and then click on this paragraph (the click
		event is removed).</p>

		<button>Remove the click event handler</button>

	</body>
</html>
```

5. Effects

animate()

Runs a custom animation on the selected elements

Example.74: animate() function:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("div").animate({left: '250px'});
		  });
		});
		</script>
	</head>
	<body>

		<button>Start Animation</button>

		<p>By default, all HTML elements have a static position, and cannot be
		moved. To manipulate the position, remember to first set the CSS
		position property of the element to relative, fixed, or absolute!</p>

		<div style="background:#98bf21; height:100px; width:100px;
		position:absolute;"></div>

	</body>
</html>
```

fadeIn()

Fades in the selected elements

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

Example.75: fadeIn() example

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("#div1").fadeIn();
		    $("#div2").fadeIn("slow");
		    $("#div3").fadeIn(3000);
		  });
		});
		</script>
	</head>
	<body>

		<p>Demonstrate fadeIn() with different parameters.</p>

		<button>Click to fade in boxes</button><br><br>

		<div id="div1" style="width:80px; height:80px; display:none;
		background-color:red;"></div><br>
		<div id="div2" style="width:80px; height:80px; display:none;
		background-color:green;"></div><br>
		<div id="div3" style="width:80px; height:80px; display:none;
		background-color:blue;"></div>

	</body>
</html>
```

fadeOut()

Fades out the selected elements

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

Example.76: fadeOut() example:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("#div1").fadeOut();
		    $("#div2").fadeOut("slow");
		    $("#div3").fadeOut(3000);
		  });
		});
		</script>
	</head>
	<body>
		<p>See the fadeOut() method example with different parameters.</p>
		<button>Click to fade out boxes</button><br><br>
		<div id="div1" style="width:80px; height:80px; background-
		color:red;"></div><br>
		<div id="div2" style="width:80px; height:80px; background-
		color:green;"></div><br>
		<div id="div3" style="width:80px; height:80px; background-
		color:blue;"></div>
	</body>
</html>
```

fadeTo()

Fades in/out the selected elements to a given opacity

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).

Example.77: fadeTo() example:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("#div1").fadeTo("slow", 0.15);
		    $("#div2").fadeTo("slow", 0.4);
		    $("#div3").fadeTo("slow", 0.7);
		  });
		});
		</script>
	</head>
	<body>

		<p>Demonstrate fadeTo() with different parameters.</p>

		<button>Click to fade boxes</button><br><br>

		<div id="div1" style="width:80px; height:80px; background-
		color:red;"></div><br>
		<div id="div2" style="width:80px; height:80px; background-
		color:green;"></div><br>
		<div id="div3" style="width:80px; height:80px; background-
		color:blue;"></div>

	</body>
</html>
```

fadeToggle()

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.

Example.78: fadeToggle() example:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("#div1").fadeToggle();
		    $("#div2").fadeToggle("slow");
		    $("#div3").fadeToggle(3000);
		  });
		});
		</script>
	</head>
	<body>

		<p>Demonstrate fadeToggle() with different speed parameters.</p>

		<button>Click to fade in/out boxes</button><br><br>

		<div id="div1" style="width:80px; height:80px; background-
		color:red;"></div>
		<br>
		<div id="div2" style="width:80px; height:80px; background-
		color:green;"></div>
		<br>
		<div id="div3" style="width:80px; height:80px; background-
		color:blue;"></div>

	</body>
</html>
```

hide()

Hides the selected elements

show()

Shows the selected elements

Example.79: hide() and show() examples:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $(".btn1").click(function(){
		    $("p").hide();
		  });
		  $(".btn2").click(function(){
		    $("p").show();
		  });
		});
		</script>
	</head>
	<body>

		<p>This is a paragraph.</p>

		<button class="btn1">Hide</button>
		<button class="btn2">Show</button>

	</body>
</html>
```

slideDown()

Slides-down (shows) the selected elements

slideUp()

Slides-up (hides) the selected elements

Parameter	*Description*
speed	Optional. Specifies the speed of the slide effect. Default value is 400 milliseconds Possible values: • milliseconds • "slow" • "fast"
easing	Optional. Specifies the speed of the element in different points of the animation. Default value is "swing" Possible values: • "swing" - moves slower at the beginning/end, but faster in the middle • "linear" - moves in a constant speed Tip: More easing functions are available in external plugins.

Example.80: slideDown() and slideUp() examples:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $(".btn1").click(function(){
		    $("p").slideUp();
		  });
		  $(".btn2").click(function(){
		    $("p").slideDown();
		  });
		});
		</script>
	</head>
	<body>
		<p>This is a paragraph.</p>

		<button class="btn1">Slide up</button>
		<button class="btn2">Slide down</button>
	</body>
</html>
```

slideToggle()

Toggles between the slideUp() and slideDown() methods

Example.81: slideToggle() example:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("p").slideToggle();
		  });
		});
		</script>
	</head>
	<body>

		<p>This is a paragraph.</p>

		<button>Toggle slideUp() and slideDown()</button>

	</body>
</html>
```

stop()

Stops the currently running animation for the selected elements

Example.82: stop() example:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("#flip").click(function(){
		    $("#panel").slideDown(5000);
		  });
		  $("#stop").click(function(){
		    $("#panel").stop();
		  });
		});
		</script>
		<style>
		#panel, #flip {
		  padding: 5px;
		  font-size: 18px;
		  text-align: center;
		  background-color: #555;
		  color: white;
		  border: solid 1px #666;
		  border-radius: 3px;
		}

		#panel {
		  padding: 50px;
		  display: none;
		}
		</style>
	</head>
	<body>

		<button id="stop">Stop sliding</button>

		<div id="flip">Click to slide down panel</div>
		<div id="panel">Hello world!</div>

	</body>
</html>
```

toggle()

Toggles between the hide() and show() methods

Example.83: toggle() example:

```
<!DOCTYPE html>
<html>
	<head>
		<script
		src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
		></script>
		<script>
		$(document).ready(function(){
		  $("button").click(function(){
		    $("p").toggle();
		  });
		});
		</script>
	</head>
	<body>

		<p>This is a paragraph.</p>

		<button>Toggle between hide() and show()</button>

	</body>
</html>
```

6. AJAX Introduction

AJAX is about updating parts of a web page, without reloading the whole page.

Before you continue you should have a basic understanding of the following:

- HTML
- JavaScript

6.1. What is AJAX?

AJAX is a technique for creating fast and dynamic web pages.

Stands for:

AJAX = Asynchronous JavaScript and XML

Hint: AJAX is a misleading name. You don't have to understand XML to use AJAX.

AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.

Hint: Examples of applications using AJAX: Google Maps, Gmail, YouTube, and Facebook.

6.2. History

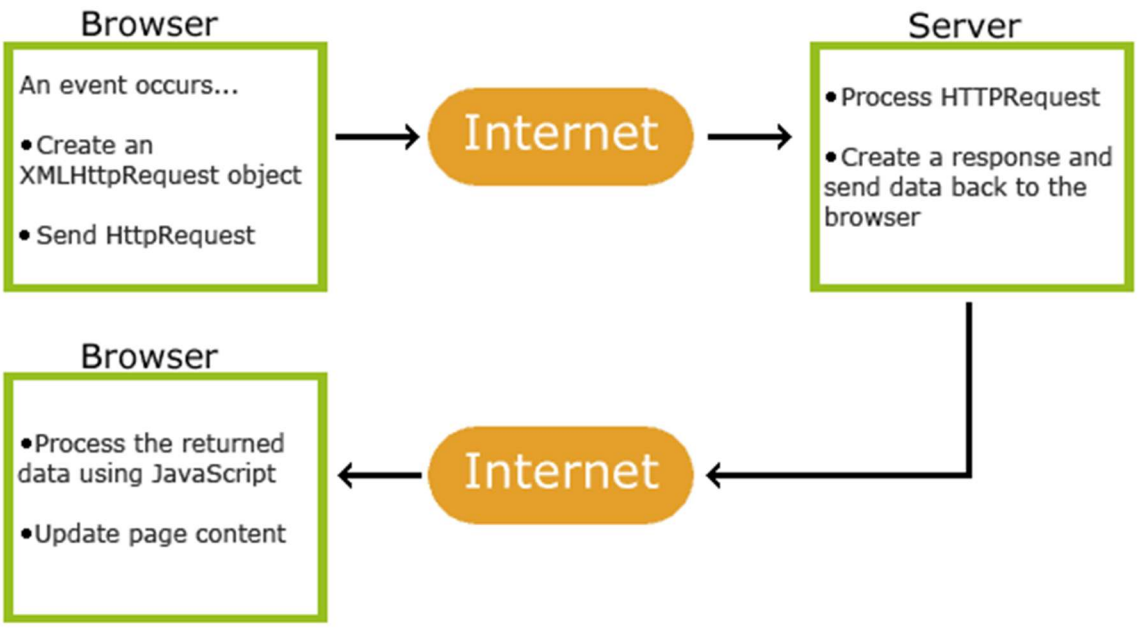
AJAX was made popular in 2005 by Google, with Google Suggest.

Google Suggest is using AJAX to create a very dynamic web interface: When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a list of suggestions.

6.3. How AJAX Works

AJAX is Based on Internet Standards:

- XMLHttpRequest object (to retrieve data from a web server)
- JavaScript/DOM (to display/use the data)



6.3.1. The XMLHttpRequest Object

All modern browsers (Chrome, IE7+, Firefox, Safari, and Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

```
variable = new XMLHttpRequest();
```

6.3.2. Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Method	*Description*
open(method, url, async)	Specifies the type of request method: the type of request: GET or POST url: the server (file) location async: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(string)	Sends the request to the server (used for POST)

6.3.3. GET or POST?

GET is simpler and faster than POST, and can be used in most cases.

However, always use POST requests when:

- A cached file is not an option (update a file or database on the server).
- Sending a large amount of data to the server (POST has no size limitations).
- Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

Post Method	*Get Method*
xhttp.open("POST", "demo_post.asp", true); xhttp.send();	xhttp.open("GET", "demo_get.asp", true); xhttp.send();

Example. 84: AJAX

ajax_info.txt

```
This is a text file for loading through AJAX Example
```

test.html

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Let AJAX change this text.</p>

<button type="button" onclick="loadDoc()">Change Content</button>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "ajax_info.txt", false);
  xhttp.send();
  document.getElementById("demo").innerHTML = xhttp.responseText;
}
</script>

</body>
</html>
```

```
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
<script src="js/jquery-3.5.1.js"></script>
</head>

<body>
<p id="demo"></p>

<button >OPen</button>
```

```
        <script>
            $(document).ready(function(){
                $("button").on("click", function(){
                    var xhttp = new XMLHttpRequest();
                    xhttp.open("GET", "ajax_info.txt", false);
                    xhttp.send();

                    $("#demo").html(xhttp.responseText);
                });
            });
        </script>
    </body>
    </html>
```