jQuery

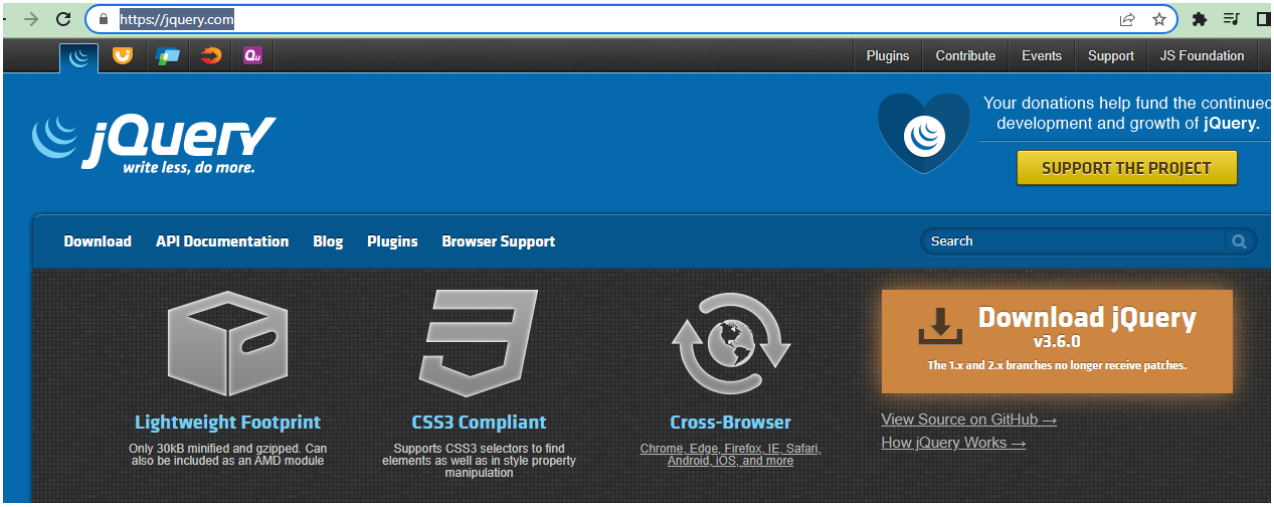
jQuery, one of the most popular and widely used JavaScript libraries on the web. JQuery makes all kinds of common scenarios in website and application development a lot easier. It provides a standardized, simplified way of extracting, manipulating, and creating webpage content. It makes operations like retrieving data via AJAX incredibly simple. This material covers the basic of jQuery. We will start by seeing how to use jQuery to extract information from webpages using the rules of CSS

Setting up the development environment

First, you will need a good code editor. You can use any text editor like Sublime Text, TextWrangler, Atom, Notepad++, or Visual Studio Code.

Most of the examples in this topic will not need a server, but, when we get to the section on AJAX, we will need a server because AJAX requests have to be sent from a server for most browsers to properly handle them.

You can download jQuery directly from **https://jquery.com/**



https://www.syncfusion.com/succinctly-free-ebooks/javascript

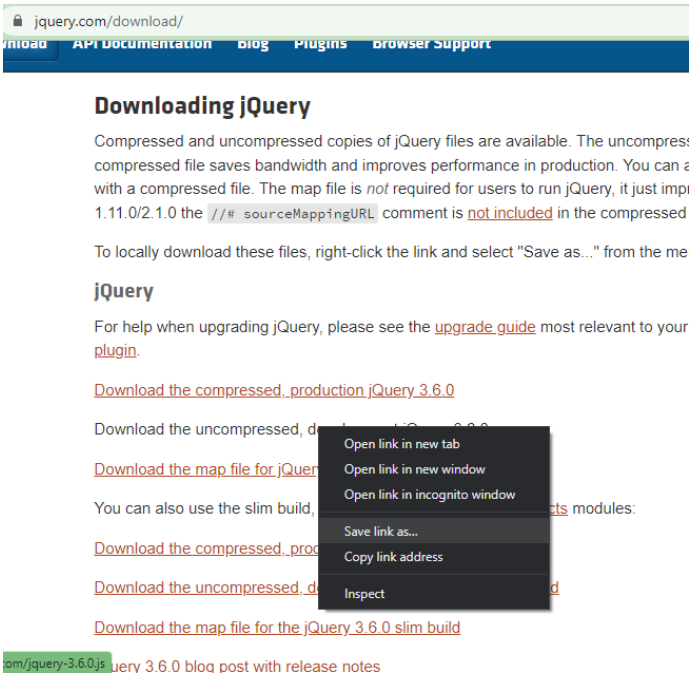
Version 3.6.0 is the latest branch of the library, and it does not support some older browsers, most notably, Internet Explorer version 6 through 8. If you need to support those older versions of IE, then you can continue to use the version 1.0 branch, which will continue to get bug fixes but will not be receiving new features.

At jquery.com, there are two distributions of jQuery, there is the compressed production version, and then there's the uncompressed development version. Typically, you download both,

and then deploy the production version to your server while using the development version to debug your code. Since we are just going to practice and be familiar with jQuery, we are going to be using the development version in this course. We can also have the URL link to jQuery development in our `<script>` https://code.jquery.com/jquery-3.6.0.js

```
<body>
<script src=" https://code.jquery.com/jquery-3.6.0.js "></script>
</body>
```

If you want to download the jQuery, either right click or control+click on the link for downloading jQuery, then choose **`save link as…`**, and then **save** jQuery into the root level of the exercise folder.



What is jQuery

JQuery is one of the most popular, widely-used, open-source free JavaScript libraries on the web. Using jQuery brings a whole host of benefits to your development work-flow. It greatly simplifies the tasks commonly involved with creating modern web applications that are highly interactive and responsive. Features like AJAX, dynamic content, rich animations, are all made easier by jQuery. JQuery works across all current modern browsers.

Selectors and filters

Selectors and filters work together to retrieve document content. Selectors are used to select content, as their name implies, and then filters are used to further refine the selected content. In this way, you can think of selectors and filters as the query part of jQuery.

The resulting content can then be manipulated by other jQuery or plain JavaScript functions. The result of using selectors and filters to retrieve content, is an array of objects that meet the specified criteria. Now it is important to understand that array that comes back is not a set of pure DOM elements. It is a collection of jQuery objects that are wrapped around each of the DOM elements. These jQuery objects provide a large number of functions and properties for further operating on the content.

Basic 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 name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.

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

jQuery's **`.ready()`** method checks that the page is ready for your code to work with.

```
$(document).ready(function(){
// here goes the jQuery lines
});
```

As with plain JavaScript, if the browser has not yet constructed the DOM tree, jQuery will not be able to select elements from it.

Basic jQuery selectors work by using a CSS-like syntax to select content from the page.

To select all the elements in a page with a particular `tagName`, we simply pass the name of the tag to the jQuery function, which will select all of those elements. Similarly, we can pass an identifier of a specific element, and jQuery will select the element with that ID attribute that has the same name as the identifier.

Working with css

Trying to build a modern Web application without CSS is a lot like trying to build a sandwich without bread. Ultimately, it is not going to work very well, and the result is going to be messy. On top of that, CSS is one of those areas that is notoriously difficult to work with across browsers. The good news is that jQuery provides a variety of CSS functions that smooth out the CSS work flow for working across browsers and for accomplishing things that would normally require you to write custom code.

The **`css()`** method sets or returns one or more style properties for the selected elements. To set a specified CSS property, use the following syntax:

Syntax

```
$("p").css('border','3px solid red');
```

jQuery selector — Assigning CSS properties

Set Multiple CSS Properties

To set multiple CSS properties, use the following syntax:

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

jQuery order are used in conjunction with selectors in order to further refine a result set that comes back from a selector expression.

Syntax

```
$("#example p:last").css("background-color", "pink");
```

jQuery order selector — Assigning CSS properties

Example 1) Use jQuery selectors to add CSS attributes to elements

```
<body>
    <h1>Using Basic jQuery Selectors</h1>
    <section id="content">
       <p>Selectors are used to select content within a Web page.</p>
       <p>They use a very CSS-like syntax in order to do this.</p>
        <ul id="list1">
          <h3>Fruits and Veggies</h3>
            <li class="a">Apples</li>
            <li class="a">Grapes</li>
            <li class="b">Carrots</li>
            <li class="b">Tomatoes</li>
        </ul>
        <div id="example">
            <p class="a"> paragraph 1</p>
            <p id="para1"> paragraph 2</p>
            <p class="b"> paragraph 3</p>
            <p id="para4" lang="en-us"> paragraph 4</p>
            <p id="para5" lang="en-gb"> paragraph 5</p>
        </div>
      </section>

    <script type="text/javascript" src="jquery_3_6.js"></script>
    <script type="text/javascript" src="basicSelector.js"></script>
  </body>
```

js file basicSelector.js

```
$(document).ready(function(){
 $("#para1").css('border','3px solid red');
 $('.a').css({'color':'magenta', 'padding':'0.5em 1em','background-color':'yellow'});
 $('#example').css('background-color','lightblue');
});
```

Descendants Selectors

If you've ever worked with some of the more advanced CSS selectors, such as the parent-child or descendants selectors, these are going to look very familiar to you because they use the same syntax.

For example, the child selector "`parent > child`" selects "child" elements that are immediate descendants of the "parent"

Syntax

```
$("div > p").css("background-color", "green");
```

jQuery descendent selector — Assigning CSS properties

Example 2) use different jQuery selectors to select elements and add CSS attributes to elements

```
$("document").ready(function() {
  // The child selector "parent > child" selects "child" elements that are
immediate descendants of the "parent"
$("div > p:odd").css("background-color", "orange");
 // The descendant selector "ancestor descendant" selects "descendant"
elements as long as they have an "ancestor" element somewhere above them
      $("div p.a").css("border", "10px ridge olive"); // asking for a p
element that has the class name 'a' whick is also inside of the element 'div'

// The next adjacent selector "prev + next" selects the "next" element if it
is immediately preceded by a "prev" element
      $("ul + div ").css("border", "3px solid red"); // '+' means select
element 'div' that is right next (adjacent) to element 'ul'

// Next sibling selector "prev ~ siblings" selects all "siblings" elements
that come after a "prev" element
      $("ul + div ").css({"border": "10px dashed gray", 'margin':'1em
auto','width':'80%'});//The next adjacent selector says select, in this case
'p' as long as it has an element with para1 as its previous sibling.
});
```

Attribute Selectors

The attribute selectors in jQuery let you perform various tests to see if attributes are present and optionally contain certain values. In this first example, we will write a jQuery selector that's looking for paragraph tags and for attribute filters, we put the attributes inside square brackets.

Syntax

```
$("p[class]").css("border", "3px solid red");
```

Example 3) Use attribute selectors to select elements and add CSS attributes to elements

```
$("document").ready(function() {
  $("p[class]").css("border", "3px solid red");
  $("p[id=para1]").css("border", "3px solid red");
  $("p[id^=para]").css("border", "3px solid red"); // select all 'p' element
with id that 'starts ^' with para
  $("p[id^=para][lang*=en-]").css("border", "3px solid red");// select all
'p' element with id that 'starts ^' with para AND attribute lang that contain
'*=' the text (en-) somewehre
});
```

Advanced filters

Content filters let you filter the results of jQuery selectors by examining the content of the selectors themselves.

Syntax

```
$("p:contains('3')").css("border", "3px solid red");
```

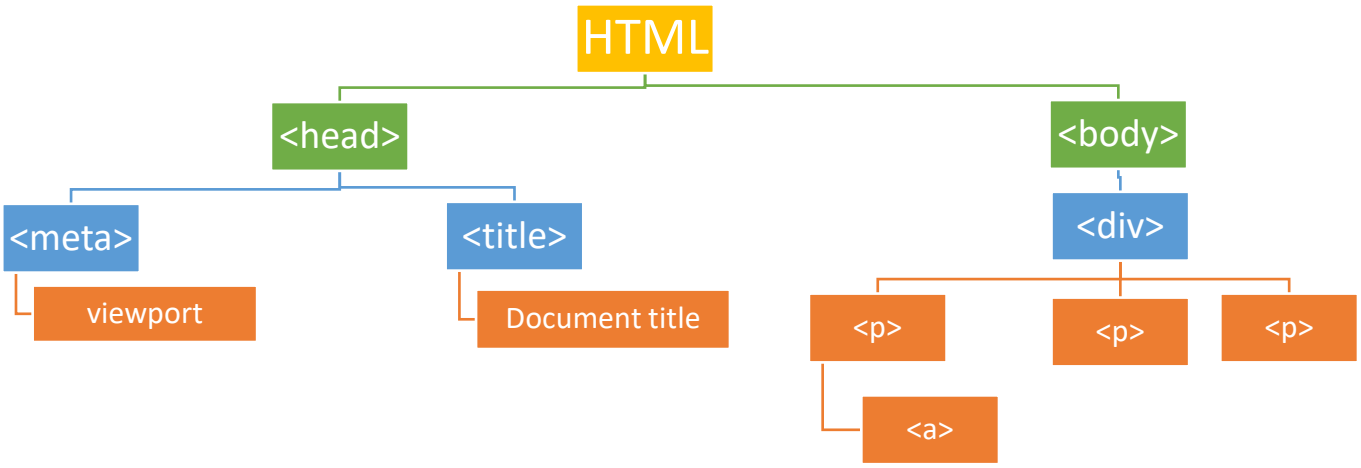
Example 4) Use advanced filter selector to select elements

```
$("document").ready(function() {
  $("p:contains('3')").css("border", "3px solid red"); // select paragraph
that contains string 3
  $("p:parent").css("border", "3px solid red"); // parent filter :  Content
filters let you filter the results of jQuery selectors by examining the
content of the selectors themselves. So, for the first example, I'll use the
contains filter. So, I'll write the selector to look for paragraph tags and
then use the css trick that we've been using.
  $("div:has(p[class=a])").css("border", "3px solid red");// select div that
has p with class a
  $("div p:first-child").css("border", "3px solid red");
  $("div p:last-of-type").css("border", "3px solid red"); // look for the
last p that is inside of a div
  $("div p:nth-child(3)").css("border", "3px solid red"); //look for the 3rd
p that is inside of a div
  $("div p:nth-child(2n)").css("border", "3px solid red"); // 2n ==> select
every multiple of 2,
});
```

You can also check more jQuery selector at
https://www.w3schools.com/jquery/jquery_selectors.asp

Traversing documents with jQuery

The relationships between elements in the page within this tree structure can be described using family-like terms. For example, the HTML tag is said to be at the root of the document tree, and is the parent of the head and body tags. In turn, the head and body tags are also said to be children of the HTML tag and siblings of each other. This tree structure and the relationships between the elements are described using a standard API called the Document Object Model, or DOM.



From the DOM tree above, consider a simple HTML page, represented by the tree diagram you see here. Now, imagine that we have a reference to this paragraph element in the tree. The paragraph that follows this one is called the next sibling. There is a function to get it called next. The paragraph before this one is called the previous sibling, and there is a function to get that called prev.

The `div` tag that contains all these paragraphs is called their parent. Moreover, not surprisingly, there's a function to get that as well, called parent. In fact, there's a function to get all the parents of a given tag, as a set called parents. For example, from the tree structure above, you can see that the `div`, `body`, and `HTML` tags are all parents of this paragraph. There is even a function to get the parents of an element, but only until a particular tag is reached. That is called `parentsUntil`. In this case, you can see that `parentsUntil`, when we pass in the HTML tag, only retrieves the `div` and the `body`. It stops and doesn't include the one that we passed in as the argument

Example 5) using the HTML from example 1, apply DOM tree ancestor selectors to **para1**

```
<script>
	$("document").ready(function() {
	// The children() function retrieves the immediate (that is, first-level
	down) child elements of the matched set, excluding text nodes.
	let elem = $("#para1");
	elem.prev().css("border", "10px solid red"); // previus element in para1
	elem.next().css("border", "10px dotted blue");// nextelement in para1
	elem.parents().css("border", "10px ridge gray"); // all parents that
	includes para1 ==> event flow = capturing event
	elem.parentsUntil($("body")).css("border", "10px dashed green");// all
	parents of the para1 until it reaches body and the body itself is not
	include
	// use the find function to locate content within particular elements
	$("#example").find("#para4").css("border", "10px onset purple");//starts
	in #example and find the if para4
	// use the each function to iterate over a set of elements and operate on
	them
	});
</script>
```

Manipulating page content

Once you have used the selectors and filters to get the content out of the web page, typically you are going to want to do something with it. In addition to working with existing content, there are times when you are going to want to create new content and add it into the page. For these purposes, you use jQuery's content creation and manipulation functions.

As you will see in this section, there are functions for creating, copying, deleting, and moving content around. In addition to manipulating content, jQuery provides some really good cross browser support for working with CSS style information including positioning and sizing.

Inserting page content

jQuery provides some rich flexibility for inserting content into webpages, and that's what we'll examine in this lesson. There are two ways to insert content into pages. You can insert content inside of other elements, and you can insert content relative to the outside of other elements, and we'll examine both ways.

append and appendTo function

The **append** function allows us to take content and add that content to the inside of the end of the matched elements returned by a jQuery selector expression. In this case the content is a string, but it could also be HTML content, or another jQuery object, or even an array of jQuery objects. This jQuery selector retrieves all paragraph tags and appends the content to the end of the inside of the paragraphs.

Syntax

```
$('destination').append('new content');
```

Example 6) Use **append** method to add content to the end of `<p class="p_append">`

```
<p class="p_append">Original paragraph</p>


$('.p_append').append('New Content!');
```

Output ➔ Append paragraphNew Content!

To insert content inside a set of elements at the beginning position, you can use either prepend, or the **prependTo** functions. The end result of this would be the content inserted at the beginning. To insert content adjacent to other elements, that is, either before or after, as opposed to inside, you use the before and after functions.

```
$('.p_append').prepend('New content using prepend');
```

Output ➔ New content using prependOriginal paragraph

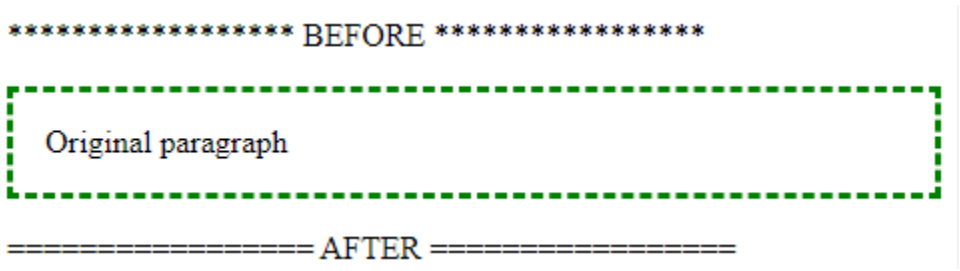
before and insertBefore function

To insert content adjacent to other elements, that is, either before or after, as opposed to inside, you use the **before** and **after** method. The **before** method takes content and will insert it before the outside of the matched set of elements. While the after method places the content after the outside of the matched set of elements.

Example 7) Use **before** and **after** method to add content to the beginning and end, respectly, of `<p class="originalP">` element

```
<p class="originalP" style="border:dashed green; padding:1em;">Original
paragraph</p>

$("<p>New Para after</p>").insertAfter(".p_append");
$("<p>New Para before</p>").insertBefore(".p_append");
```

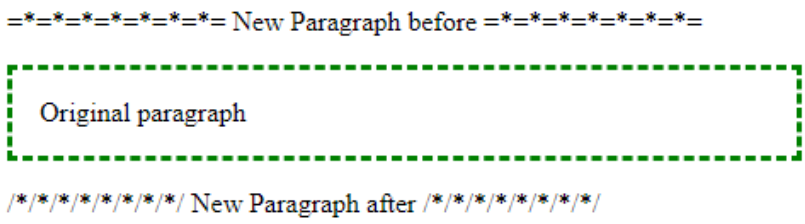


An **`insertBefore`** version that specifies the new content first, and the selector second. Here, my jQuery selector expression is selecting a page as paragraph tags, and inserting the content before the tags. As you might imagine, there are after, and **`insertAfter`** functions which do the same thing, except they insert the content on the outside end of the matched elements.

Example 8) apply **`insertBefore`** and **`insertAfter`** method to insert content before and after, respectly, `<p class="originalP">` element

```
<p class="originalP" style="border:dashed green; padding:1em;"> Original
paragraph</p>

$("<p>/*/*/*/*/*/*/*/*/ New Paragraph after /*/*/*/*/*/*/*/*/
</p>").insertAfter(".originalP");
        $("<p>=*=*=*=*=*=*=*=*= New Paragraph before =*=*=*=*=*=*=*=*=
</p>").insertBefore(".originalP");
```



Altering page content

jQuery allows user to alter content that is already in the page by using the methods:

- `wrap("content");` wraps content around existing element or content element
- `wrapAll("content");` wraps the parent element with the given content
- `empty();` clear all content or element of the selected object
- `$("new element").replaceAll("selected element");` Replace all selected element with the new element.
- `$("selected element").replaceWith("new element");` Replace all selected element with the new element.

Example 9) apply different methods to alter page content

```
  <div id="example">
     <p class="a">This is paragraph 1</p>
     <p id="para1">This is paragraph 2</p>
     <p class="b">This is paragraph 3</p>
     <p id="para4" >This is paragraph 4</p>
     <p id="para5" >This is paragraph 5</p>
   </div>
$("document").ready(function() {
    $("#example p").wrap("<div style='color:red'/>"); // wrap function wraps
content around existing
    $("#example p").wrapAll("<div style='color: red'/>"); // wrap the parent
elements of #example p into the given css
    $("#example").empty(); // clear all element in #example
    $("#example p.a, #example p.b").remove();
    $("#example p.a, #example p.b").detach(); // detach temporary remove the
data
    $("<div>replaced</div>").replaceAll("#example p[id]"); // replace all p
elements in #example that has id with the new given div
    $("#example p[id]").replaceWith("<div>replaced</div>"); // replace from
left to right
});
```

jQuery Events

jQuery Event Handling Features

Building any kind of modern web application today means that you have to handle all kinds of user generated events. Events are the foundation of the kinds of features that users expect from great web apps today. Drag and drop, shortcut keys, rich animations, etc. In the old days before browsers settled on using the modern W3C standard event API, this was a hard problem to solve because different browsers used different event APIs.

jQuery provides a mechanism for working with events that's simpler than relying on the document object model, DOM, and the W3C APIs, and because jQuery works with sets of elements by default, it's very easy to write code that assigns event handlers to groups of objects just by using the results of the selectors and filters that we've already learned about.

Even though today's browsers have standardized around the W3 event API, the properties that are passed to event handler functions differ between browsers. jQuery solves that problem by creating a unified event object that exposes the properties and methods that you use most commonly in a cross-browser way.

Binding and Unbinding Events

There are several functions that you can use to work on Events. Now, some of these have been deprecated, so you might come across older jQuery code that uses functions like bind and live and die and so on and so forth.

But the only functions that we're going to concern ourselves right now are ***on*** and ***off***. These are the modern ways, as of jQuery 1.8, to attach and remove event handlers from elements.

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

Syntax

```
$(selector).off(event,selector,function(eventObj),map)
```

Parameter	Description
event	Required. Specifies one or more events or namespaces to remove from the selected element(s). Multiple event values are separated by a space. Must be a valid event
selector	Optional. A selector which should match the one originally passed to the on() method when attaching event handlers
function(eventObj)	Optional. Specifies the function to run when the event occurs
map	Specifies an event map ({event:function, event:function, ...}) containing one or more event to attach to the elements, and functions to run when the events occur

click() event

The click event occurs when an element is clicked.

The **click()** method triggers the click event, or attaches a function to run when a click event occurs.

Syntax

`Trigger the click event for the selected elements:`

```
$(selector).click()
```

Attach a function to the click event:

```
$(selector).click(function)
```

toggleClass( )

The *toggleClass( )* method toggles between adding and removing one or more class names from the selected elements.

This method checks each element for the specified class names. The class names are added if missing, and removed if already set - This creates a toggle effect.

However, by using the "switch" parameter, you can specify to only remove, or only add a class name.

Syntax

```
$(selector).toggleClass(classname,function(index,currentclass),switch)
```

Parameter	Description
classname	Required. Specifies one or more class names to add or remove. To specify several classes, separate the class names with a space
Function *(index,currentclass)*	Optional. Specifies a function that returns one or more class names to add/remove • *index* - Returns the index position of the element in the set • *currentclass* - Returns current class name of selected elements
switch	Optional. A Boolean value specifying if the class should only be added (true), or only be removed (false)

Example 10) use click event to add CSS properties and add, remove, and toggle class to elements

CSS

```
.pClass {color: green; background-color: lightgray;}
button {margin: 1em 1em 0em 0em;}
```

HTML

```
<h1>CSS Properties and Class Manipulation</h1>
<section id="content">
   <p> jQuery functions</p>
   <p>CSS methods()</p>
   <ul>
     <li><code>css()</code>:set CSS properties on the matched elements</li>
     <li><code>hasClass(className)</code>: check if an element has a certain class</li>
     <li><code>addClass(className | function)</code>: add the given CSS class to the
elements in the matched set</li>
     <li><code>removeClass(className | function)</code>: remove the given CSS class from
the elements in the matched set</li>
     <li><code>toggeClass(className | function)</code>: add or remove the given CSS
class to the elements in the matched set depending on whether it is already there</li>
</ul>
   <div id="example">
     <p class="a">This is paragraph 1</p>
     <p id="para1">This is paragraph 2</p>
     <p class="b">This is paragraph 3</p>
     <p id="para4" lang="en-us">This is paragraph 4</p>
     <p id="para5" lang="en-gb">This is paragraph 5</p>
   </div>
   <button id="setProp">Set Property</button>
   <button id="setProps">Set Properties</button>
   <button id="addCl">Add Class</button>
   <button id="rmCl">Remove Class</button>
   <button id="toggleCl">Toggle Class</button>
</section>
```

JS

```
$("document").ready(function() {
  // #setProp on click
    $("#setProp").click(function() {
        $("#example p").css("text-decoration", "overline")
                       .css("font-size", "+=1pt"); });
//
    $("#setProps").click(function() {
        $("#example p").css({
            "font-weight" : "bold",
            "color" : "red",
            "text-decoration" : "underline" });
//
          $("#addCl").click(function() {
              $("#example p").addClass("pClass");
    });
//
    $("#rmCl").click(function() {
        $("#example p").removeClass("pClass");
    });
//
    $("#toggleCl").click(function() {
        $("#example p").toggleClass("pClass");
    });
});
```

Example 11) Create a program that when a mouse moves over the a division, the background color changes. Also, when you click on a division, the event will turn off.

CSS

```
*{box-sizing: border-box;}
.target_div{height: 200px; width: 200px; background-color: olive; color:
lightyellow;padding: 1em;}
.changeBG{background-color: orange;}
```

HTML

```
<section class="container">
  <p>jQuery makes events easy to apply using the <code>on()</code> and
     <code>off()</code> methods </p>
   <div class="target_div">Mouse over this div to see the effect. You can also
     click to unbind the mouseover</div>
</section>
```

JS

```
$('.target_div').on('mouseover mouseleave',mouseEffect);
$('.target_div').on('click',function(){
      $('.target_div').off('mouseover mouseleave', mouseEffect);
      $('.target_div').removeClass('changeBG')
      $('.target_div').html('<h2>The mouse event was removed!</h2>')
})

function mouseEffect(){
      $('.target_div').toggleClass('changeBG')
}
```

If we save this program and we run it, you can see that when you move the mouse over and off the element that the class is being added and removed and it is just simply making the background orange.

Now, let us take a look at on the click event. When you click on the division **`.target_div`**, the event will just simply display a message that the **`on`** event will turn **`off`** using the `html` method. So inside the event handler, it is an anonymous JavaScript function, It will remove the highlight event handler for the mouse over and mouse leave by using the off function. Next, the function will replace the **`html`** inside the event target with a heading saying "The mouse event was removed!"

The last line in the function will remove the highlighted class if it's already there.

hover( ) event

One to the jQuery event method is ***hover***. The hover event pretty much does the same thing that my previous ***on*** and ***off*** event.

The ***hover( )*** method specifies two functions to run when the mouse pointer hovers over the selected elements.

This method triggers both the *mouseenter* and *mouseleave* events.

Note: If only one function is specified, it will be run for both the mouseenter and mouseleave events.

Syntax

```
$(selector).hover(inFunction,outFunction)
```

dblclick( ) event

The `dblclick` event occurs when an element is double-clicked. The dblclick() method triggers the dblclick event, or attaches a function to run when a dblclick event occurs.

Tip: The dblclick event also generates a click event. This can cause problems if both events are applied to the same element.

Syntax

Trigger the dblclick event for the selected elements:

```
$(selector).dblclick()
```

resize( ) event

The resize event occurs when the browser window changes size. The `resize()` method triggers the resize event, or attaches a function to run when a resize event occurs.

Syntax

Trigger the resize event for the selected elements:

```
$(selector).resize()
```

Example 12) Create a program to practice hover, click, dbclick, one, and resize events.

CSS

```
.changeBG{background-color: orange;}

.circle{background-color: purple; height: 200px; width: 200px; border-radius:
50%; padding: 3em; text-align: center;}
```

HTML

```
<div class="circle" id='circle'>
   Mouse over to see the hover event. Also try to doble-click
</div>
```

JS

```
$(function(){
   $('#circle').hover(hoverCircle);
   $('#circle').dblclick(fnDoubleClick);
   $(window).resize(fnResize);
});
function hoverCircle(){
   $('#circle').toggleClass('changeBG')
   $('#circle').html('<h3>hover event</h3>')
}
function fnDoubleClick(){
   $('#circle').html('<h3>Double-clicked!</h3>')
}
function fnResize(){
   $('#circle').html('<h3>Window was resize!</h3>')
}
```

one( ) event

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.

Syntax

```
$(selector).one(event,data,function)
```

JS

```
$('#circle').one('click', function(){
  $(this).css({border:'10px inset gray', cursor:'zoom-in'})
})
```