JQuery Tutorial

1(a). Downloading jQuery

To get started, first download a copy of jQuery and include it in your document. There are two versions of jQuery available for downloading — compressed and uncompressed.

The uncompressed file is best suited for development or debugging; while, the minified and compressed file is recommended for production because it saves the precious bandwidth and improves the performance due to small file size.

You can download jQuery from here: https://jquery.com/download/ (right click, save as)

Once you've downloaded the jQuery file you can see it has .js extension, because the jQuery is just a JavaScript library, therefore you can include the jQuery file in your HTML document with the <script> element just like you include normal JavaScript files.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Simple HTML Document</title>
  <link rel="stylesheet" href="style.css">
  <script src="jquery-3.5.1.min.js"></script>
</head>
<body>
  <h1>Hello, World!</h1>
</body>
</html>
```

Always include the jQuery file before your custom scripts; otherwise, the jQuery APIs will not be available when your jQuery code attempts to access it.

As you can see the attribute type="text/javascript" has been omitted from inside the <script> tag in the above example. This is not required in HTML5. JavaScript is the default scripting language in HTML5 and in all modern browsers.

1(b). Including jQuery from CDN

Alternatively, you can include jQuery in your document through freely available CDN (Content Delivery Network) links, if you don't want to download and host jQuery yourself.

CDNs can offer a performance benefit by reducing the loading time, because they are hosting jQuery on multiple servers spread across the globe and when a user requests the file, it will be served from the server nearest to them.

This also offers an advantage that if the visitor to your webpage has already downloaded a copy of jQuery from the same CDN while visiting other sites, it won't have to be re-downloaded since it is already there in the browser's cache.

jQuery's CDN provided by StackPath

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

You can also include jQuery through Google[1] and Microsoft[2] CDN's.

2. Creating Your First jQuery Powered Web Page

So far you have understood the purposes of jQuery library as well as how to include this in your document, now it's time to put jQuery into real use.

In this section, we will perform a simple jQuery operation by changing the colour of the heading text from the default black colour to red.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My First jQuery Powered Web Page</title>
  <link rel="stylesheet" href="style.css">
  <script src="jquery-3.5.1.js"></script>
  <script>
    $(document).ready(function(){
      $("h1").css("color", red");
    });
  </script>
</head>
<body>
  <h1>Hello, World!</h1>
</body>
</html>
```

In the above example we've performed a simple jQuery operation by changing the colour of the heading i.e. the <h1> element using the jQuery element selector and css() method when the document is ready which is known as document ready event.

[1] https://developers.google.com/speed/libraries/#jquery
[2] https://docs.microsoft.com/en-us/aspnet/ajax/cdn/overview#jQuery_Releases_on_the_CDN_0

3. Standard jQuery Syntax

A jQuery statement typically starts with the dollar sign ($) and ends with a semicolon (;).

In jQuery, the dollar sign ($) is just an alias for jQuery. Let's consider the following example code which demonstrates the most basic statement of the jQuery.

```
<script>
  $(document).ready(function(){
    // Some code to be executed...
    alert("Hello World!");
  });
</script>
```

The above example simply displays an alert message "Hello World!" to the user.

Explanation of code

If you are completely new to the jQuery, you might think what that code was all about. OK, let's go through each of the parts of this script one by one.

- The <script> element — Since jQuery is just a JavaScript library, so the jQuery code can be placed inside the <script> element.

- The $(document).ready(handler); — This statement is typically known as ready event. Where the handler is basically a function that is passed to the ready() method to be executed safely as soon as the document is ready to be manipulated i.e. when the DOM hierarchy has been fully constructed.

The jQuery ready() method is typically used with an anonymous function. So, the above example can also be written in a shorthand notation like this:

```
<script>
  $(function(){
    // Some code to be executed...
    alert("Hello World!");
  });
</script>
```

You can use any syntax you like as both the syntax are equivalent. However, the document ready event is easier to understand when reading the code.

Further, inside an event handler function you can write the jQuery statements to perform any action following the basic syntax, like: $(selector).action();

Where, the $(selector) basically selects the HTML elements from the DOM tree so that it can be manipulated and the action() applies some action on the selected elements such as changes the CSS property value, or sets the element's contents, etc. Let's consider another example that sets the paragraph text after the DOM is ready:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery Document Ready Demo</title>
  <link rel="stylesheet" href="style.css">
  <script src="jquery-3.5.1.js"></script>
  <script>
    $(document).ready(function(){
      $("p").text("Hello World!");
    });
  </script>
</head>
<body>
  <p>Not loaded yet.</p>
</body>
</html>
```

In the jQuery statement of the example above the p is a jQuery selector which select all the paragraphs i.e. the <p> elements in the document, later the text() method set the paragraph's text content to "Hello World!" text.

The paragraph text in the example above is replaced automatically when the document is ready. But what if we want the user to perform some action before executing the jQuery code to replace the paragraph text. Let's consider one last example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery Click Handler Demo</title>
  <link rel="stylesheet" href="style.css">
  <script src="jquery-3.5.1.js"></script>
  <script>
    $(document).ready(function(){
      $("button").click(function(){
        $("p").text("Hello World!");
      });
    });
  </script>
</head>
<body>
  <p>Not loaded yet.</p>
  <button type="button">Replace Text</button>
</body>
</html>
```

In the above example the paragraph text is replaced only when a click event occurs on the "Replace Text" <button> that simply means when a user click this button.

4. jQuery Selectors

Selecting Elements with jQuery
JavaScript is most commonly used to get or modify the content or value of the HTML elements on the page, as well as to apply some effects like show, hide, animations etc. But before you can perform any action you need to find or select the target HTML element.

Selecting the elements through a typical JavaScript approach could be very painful, but the jQuery works like a magic here. The ability of making the DOM elements selection simple and easy is one of the most powerful features of jQuery.

jQuery supports almost all the selectors defined in the latest CSS3 specifications, as well as it has its own custom selectors. These custom selectors greatly enhance the capabilities selecting the HTML elements on a page.

Selecting Elements by ID
You can use the ID selector to select a single element with the unique ID on the page.

For example, the following jQuery code will select and highlight an element having the ID attribute id="mark", when the document is ready to be manipulated.

```
<script>
  $(document).ready(function(){
    // Highlight element with id mark
    $("#mark").css("background", "yellow");
  });
</script>
```

In the example above, the $(document).ready() is an event that is used to manipulate a page safely with the jQuery. Code included inside this event will only run once the page DOM is ready.

Selecting Elements by Class Name
The class selector can be used to select the elements with a specific class.

For example, the following jQuery code will select and highlight the elements having the class attribute class="mark", when the document is ready.

```
<script>
  $(document).ready(function(){
    // Highlight elements with class mark
    $(".mark").css("background", "yellow");
  });
</script>
```

Selecting Elements by Name
The element selector can be used to select elements based on the element name.

For example, the following jQuery code will select and highlight all the paragraph i.e. the <p> elements of the document when it is ready.

```
<script>
  $(document).ready(function(){
    // Highlight paragraph elements
    $("p").css("background", "yellow");
  });
</script>
```

Selecting Elements by Attribute
You can use the attribute selector to select an element by one of its HTML attributes, such as a link's target attribute or an input's type attribute, etc.

For example, the following jQuery code will select and highlight all the text inputs i.e. <input> elements with the type="text", when the document is ready.

```
<script>
  $(document).ready(function(){
    // Highlight paragraph elements
    $('input[type="text"]').css("background", "yellow");
  });
</script>
```

Selecting Elements by Compound CSS Selector
You can also combine the CSS selectors to make your selection even more precise.

For instance, you can combine the class selector with an element selector to find the elements in a document that has certain type and class.

```
<script>
  $(document).ready(function(){
    // Highlight only paragraph elements with class mark
    $("p.mark").css("background", "yellow");
    // Highlight only span elements inside the element with ID mark
    $("#mark span").css("background", "yellow");
    // Highlight li elements inside the ul elements
    $("ul li").css("background", "red");
    // Highlight li elements only inside the ul element with id mark
    $("ul#mark li").css("background", "yellow");
    // Highlight li elements inside ul element with class mark
    $("ul.mark li").css("background", "green");
    // Highlight all anchor elements with target blank
    $('a[target="_blank"]').css("background", "yellow");
  });
</script>
```

jQuery Custom Selector
In addition to the CSS defined selectors, jQuery provides its own custom selector to further enhancing the capabilities of selecting elements on a page.

```
<script>
  $(document).ready(function(){
    // Highlight table rows appearing at odd places
    $("tr:odd").css("background", "yellow");
    // Highlight table rows appearing at even places
    $("tr:even").css("background", "orange");
    // Highlight first paragraph element
    $("p:first").css("background", "red");
    // Highlight last paragraph element
    $("p:last").css("background", "green");
    // Highlight all input elements with type text inside a form
    $("form :text").css("background", "purple");
    // Highlight all input elements with type password inside a form
    $("form :password").css("background", "blue");
    // Highlight all input elements with type submit inside a form
    $("form :submit").css("background", "violet");
  });
</script>
```

5. jQuery Events

What are Events

Events are often triggered by the user's interaction with the web page, such as when a link or button is clicked, text is entered into an input box or textarea, selection is made in a select box, key is pressed on the keyboard, the mouse pointer is moved etc. In some cases, the Browser itself can trigger the events, such as the page load and unload events.

jQuery enhances the basic event-handling mechanisms by offering the events methods for most native browser events, some of these methods are ready(), click(), keypress(), focus(), blur(), change(), etc. For example, to execute some JavaScript code when the DOM is ready, you can use the jQuery ready() method, like this:

```
<script>
  $(document).ready(function(){
    // Code to be executed
    alert("Hello World!");
  });
</script>
```

The $(document).ready() is an event that is used to manipulate a page safely with the jQuery. Code included inside this event will only run once the page DOM is ready i.e. when the document is ready to be manipulated.

In general, the events can be categorised into four main groups — mouse events, keyboard events, form events and document/window events.

Mouse Events

A mouse event is fired when the user click some element, move the mouse pointer etc. Here're some commonly used jQuery methods to handle the mouse events.

The click() Method

The jQuery click() method attaches an event handler function to the selected elements for "click" event. The attached function is executed when the user clicks on that element. The following example will hide the <p> elements on a page when they are clicked.

```
<script>
  $(document).ready(function(){
    $("p").click(function(){
      $(this).slideUp();
    });
  });
</script>
```

The this keyword inside the jQuery event handler function is a reference to the element where the event is currently being delivered.

The dblclick() Method

The jQuery dblclick() method attach an event handler function to the selected elements for "dblclick" event. The attached function is executed when the user double-clicks on that element. The following example will hide the <p> elements when they are double-clicked.

```
<script>
  $(document).ready(function(){
    $("p").dblclick(function(){
      $(this).slideUp();
    });
  });
</script>
```

The hover() Method

The jQuery hover() method attach one or two event handler functions to the selected elements that is executed when the mouse pointer enters and leaves the elements. The first function is executed when the user place the mouse pointer over an element, whereas the second function is executed when the user removes the mouse pointer from that element.

The following example will highlight <p> elements when you place the cursor on it, the highlighting will be removed when you remove the cursor.

```
<script>
  $(document).ready(function(){
    $("p").hover(function(){
      $(this).addClass("highlight");
    }, function(){
      $(this).removeClass("highlight");
    });
  });
</script>
```

You can consider the hover() method is a combination of the jQuery mouseenter() and mouseleave() methods.

The mouseenter() Method
The jQuery mouseenter() method attach an event handler function to the selected elements that is executed when the mouse enters an element. The following example will add the class highlight to the <p> element when you place the cursor on it.

```
<script>
  $(document).ready(function(){
    $("p").mouseenter(function(){
      $(this).addClass("highlight");
    });
  });
</script>
```

The mouseleave() Method
The jQuery mouseleave() method attach an event handler function to the selected elements that is executed when the mouse leaves an element. The following example will remove the class highlight from the <p> element when you remove the cursor from it.

```
<script>
  $(document).ready(function(){
    $("p").mouseleave(function(){
      $(this).removeClass("highlight");
    });
  });
</script>
```

For more event methods, please check out the jQuery Events Reference[3].

Keyboard Events
A keyboard event is fired when the user press or release a key on the keyboard. Here're some commonly used jQuery methods to handle the keyboard events.

The keypress() Method
The jQuery keypress() method attach an event handler function to the selected elements (typically form controls) that is executed when the browser receives keyboard input from the user. The following example will display a message when the kypress event is fired and how many times it is fired when you press the key on the keyboard.

[3] https://api.jquery.com/category/events/

```
<script>
  $(document).ready(function(){
    var i = 0;
    $('input[type="text"]').keypress(function(){
      $("span").text(i += 1);
      $("p").show().fadeOut();
    });
  });
</script>
```

Note: The keypress event is similar to the keydown event, except that modifier and non-printing keys such as Shift, Esc, Backspace or Delete, Arrow keys etc. trigger keydown events but not keypress events.

The keydown() Method
The jQuery keydown() method attach an event handler function to the selected elements (typically form controls) that is executed when the user first presses a key on the keyboard. The following example will display a message when the keydown event is fired and how many times it is fired when you press the key on the keyboard.

```
<script>
  $(document).ready(function(){
    var i = 0;
    $('input[type="text"]').keydown(function(){
      $("span").text(i += 1);
      $("p").show().fadeOut();
    });
  });
</script>
```

The keyup() Method
The jQuery keyup() method attach an event handler function to the selected elements (typically form controls) that is executed when the user releases a key on the keyboard. The following example will display a message when the keyup event is fired and how many times it is fired when you press and release a key on the keyboard.

```
<script>
  $(document).ready(function(){
    var i = 0;
    $('input[type="text"]').keyup(function(){
      $("span").text(i += 1);
      $("p").show().fadeOut();
    });
  });
</script>
```

Keyboard events can be attached to any element, but the event is only sent to the element that has the focus. That's why the keyboard events generally attached to the form controls such as text input box or textarea.

Form Events
A form event is fired when a form control receives or loses focus or when the user modifies a form control value such as by typing text in a text input, select any option in a select box etc. Here're some commonly used jQuery methods to handle the form events.

The change() Method
The jQuery change() method attach an event handler function to the <input>, <textarea> and <select> elements that is executed when its value changes. The following example will display an alert message when you select any option in dropdown select box.

```
<script>
  $(document).ready(function(){
    $("select").change(function(){
      var selectedOption = $(this).find(":selected").val();
      alert("You have selected - " + selectedOption);
    });
  });
</script>
```

For select boxes, checkboxes, and radio buttons, the event is fired immediately when the user makes a selection with the mouse, but for the text input and textarea the event is fired after the element loses focus.

The focus() Method
The jQuery focus() method attach an event handler function to the selected elements (typically form controls and links) that is executed when it gains focus. The following example will display a message when the text input receive focus.

```
<script>
  $(document).ready(function(){
    $("input").focus(function(){
      $(this).next("span").show().fadeOut("slow");
    });
  });
</script>
```

The blur() Method
The jQuery blur() method attach an event handler function to the form elements such as <input>, <textarea>, <select> that is executed when it loses focus. The following example will display a message when the text input loses focus.

```
<script>
  $(document).ready(function(){
    $("input").blur(function(){
      $(this).next("span").show().fadeOut("slow");
    });
  });
</script>
```

The submit() Method
The jQuery submit() method attach an event handler function to the <form> elements that is executed when the user is attempting to submit a form. The following example will display a message depending on the value entered when you try to submit the form.

```
<script>
  $(document).ready(function(){
    $("form").submit(function(event){
      var regex = /^[a-zA-Z]+$/;
      var currentValue = $("#firstName").val();
      if(regex.test(currentValue) == false){
        $("#result").html('<p class="error">Not
valid!</p>').show().fadeOut(1000);
        // Preventing form submission event.preventDefault();
      }
    });
  });
</script>
```

A form can be submitted either by clicking a submit button, or by pressing Enter when certain form elements have focus.

Document/Window Events
Events are also triggered in a situation when the page DOM (Document Object Model) is ready or when the user resize or scrolls the browser window, etc. Here're some commonly used jQuery methods to handle such kind of events.

The ready() Method
The jQuery ready() method specify a function to execute when the DOM is fully loaded.

The following example will replace the paragraphs text as soon as the DOM hierarchy has been fully constructed and ready to be manipulated.

```
<script>
  $(document).ready(function(){
    $("p").text("The DOM is now loaded and can be manipulated.");
  });
</script>
```

The resize() Method
The jQuery resize() method attach an event handler function to the window element that is executed when the size of the browser window changes.

The following example will display the current width and height of the browser window when you try to resize it by dragging its corners.

```
<script>
  $(document).ready(function(){
    $(window).resize(function() {
      $(window).bind("resize", function(){
        $("p").text("Window width: " +
        $(window).width() + ", " + "Window height: " +
        $(window).height());
      });
    });
  });
</script>
```

The scroll() Method
The jQuery scroll() method attach an event handler function to the window or scrollable iframes and elements that is executed whenever the element's scroll position changes.

The following example will display a message when you scroll the browser window.

```
<script>
  $(document).ready(function(){
    $(window).scroll(function() {
      $("p").show().fadeOut("slow");
    });
  });
</script>
```