Introduction

Welcome to Module 8! This week, we will learn about jQuery. jQuery is a fast, small and feature-rich Javascript library. jQuery enables developers to perform tasks with minimal JavaScript code. It allows HTML document traversal, event handling, and Ajax to be performed with little code and a reduced learning curve.

Topics and Learning Objectives

Topics

- JavaScript
- DOM (Document Object Model) traversal
- AJAX

Learning Objectives

By the end of the module, you should be able to:

1. Analyze the front end and the back end interact over HTTP.
2. Describe distributed applications.
3. Use jQuery for making HTTP requests.
4. Consume a Flask-based API endpoint from a JavaScript web application.

Readings & Resources

Reading

Required

- React Documentation (2022)
  - Advanced Guides
  - AJAX and APIs
- jQuery

jQuery

- jQuery is a JavaScript Library.
- jQuery greatly simplifies JavaScript programming.
- jQuery is easy to learn.

- Query is a lightweight, "write less, do more", JavaScript library.
- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.
- The jQuery library contains the following features:
    - HTML/DOM manipulation
    - CSS manipulation
    - HTML event methods
    - Effects and animations
    - AJAX
    - Utilities
- The principles of developing with jQuery are:
    - Separation of JavaScript and HTML: the jQuery library provides simple syntax for adding event handlers to the DOM using JavaScript, rather than adding HTML event attributes to call JavaScript functions. Thus, it encourages developers to completely separate JavaScript code from HTML markup.
    - Brevity and clarity: jQuery promotes brevity and clarity with features like “chainable” functions and shorthand function names.
    - Elimination of cross-browser incompatibilities: the JavaScript engines of different browsers differ slightly, so JavaScript code that works for one browser may not work for another. Like other JavaScript toolkits, jQuery handles all these cross-browser inconsistencies and provides a consistent interface that works across different browsers.
    - Extensibility: new events, elements, and methods can be easily added and then reused as a plugin.

Source: Wikipedia

jQuery Example

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

Downloading jQuery

- There are two versions of jQuery available for downloading:
    - Production version – this is for your live website because it has been minified and compressed

- Development version – this is for testing and development (uncompressed and readable code)

- Both versions can be downloaded from jQuery.com.

Using jQuery from CDNs

- If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).
- Both Google and Microsoft host jQuery.
- To use jQuery from Google or Microsoft, use one of the following:

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
```

```
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-
3.2.1.min.js"></script>
```

- Many users already have downloaded jQuery from Google or Microsoft when visiting another site. As a result, it will be loaded from cache when they visit your site, which leads to faster loading time. Also, most CDNs will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.

jQuery Syntax

- The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).
- Basic syntax is: $(selector).action()
  - A $ sign to define/access jQuery
  - A (selector) to "query (or find)" HTML elements
  - A jQuery action() to be performed on the element(s)
- jQuery Syntax Examples
  - $(this).hide() – hides the current element.
  - $("p").hide() – hides all <p> elements.
  - $(".test").hide() – hides all elements with class="test".
  - $("#test").hide() – hides the element with id="test".

The Document Ready Event

- You might have noticed that all jQuery methods in our examples are inside a document ready.

```
$(document).ready(function(){
   // jQuery methods go here...
});  event:
```

- This is to prevent any jQuery code from running before the document is finished loading (is ready).
- It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.
- The jQuery team has also created an even shorter method for the document ready event:

```
$(function(){
   // jQuery methods go here...
});
```

jQuery Selectors

- jQuery selectors allow you to select and manipulate HTML element(s).
- jQuery selectors are used to “find” (or select) HTML elements based on their 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: $().

The Element Selector

- The jQuery element selector selects elements based on the element name.
- You can select all <p> elements on a page like this:
  - $("p")
- See examples of jQuery selectors in Table 8.1 below.

Table 8.1. JQuery Selector Examples.

Selector Syntax	Explanation of Selector Syntax
`$("*")`	Selects all elements
`$(this)`	Selects the current HTML element
`$("p.intro")`	Selects all <p> elements with class="intro"
`$("p:first")`	Selects the first <p> element

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

Selector Examples

Click on each of the following to reveal example code:

Click-n-reveal: **Element selector example**

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>

<script>

$(document).ready(function(){

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

    $("p").hide();

     });

});

</script>

</head>
```

```
<body>


<h2>This is a heading</h2>

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

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

<button>Click me to hide paragraphs</button>


</body>

</html>
```

Click-n-reveal: **Id selector example**

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>

<script>

$(document).ready(function(){

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

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

          });

});

</script>

</head>

<body>


<h2>This is a heading</h2>

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

```
<p id="test">This is another paragraph.</p>


<button>Click me</button>


</body>

</html>
```

Click-n-reveal: **Class selector example**

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>

<script>

$(document).ready(function(){

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

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

          });

});

</script>

</head>

<body>


<h2 class="test">This is a heading</h2>

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

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


<button>Click me</button>
```

```
</body>

</html>
```

Click-n-reveal: **p.Intro selector example**

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>

<script>

$(document).ready(function(){

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

          $("p.intro").hide();

          });

});

</script>

</head>

<body>


<h2 class="intro">This is a heading</h2>

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

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


<button>Click me</button>


</body>
```

```
</html>
```

AJAX and jQuery

- AJAX = Asynchronous JavaScript and XML.
- In short; AJAX is about loading data in the background and display it on the webpage, without reloading the whole page.
- Examples of applications using AJAX: Gmail, Google Maps, Youtube, and Facebook tabs.
- jQuery provides several methods for AJAX functionality.
- With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post – And you can load the external data directly into the selected HTML elements of your web page!
- The jQuery load() method is a simple, but powerful AJAX method.
- The load() method loads data from a server and puts the returned data into the selected element.
- `$(selector).load(URL,data,callback);`
- The required URL parameter specifies the URL you wish to load.
- The optional data parameter specifies a set of querystring key/value pairs to send along with the request.
- The optional callback parameter is the name of a function to be executed after the load() method is completed.

AJAX Examples

AJAX Example Context:

- Add some text to a file named “demo_test.txt” in order to make this example work.
- Usually the HTML div is empty and AJAX is used to populate it after a document successfully loads. You can see this in the code below.
- Save the code below to an html file name “demo_ajax.html”.

Click on each of the following to reveal example code:

Click-n-reveal: **AJAX example**

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>

<script>
```

```
$(document).ready(function(){
     $("button").click(function(){
    $("#div1").load("demo_test.txt");
     });
});
</script>
</head>
<body>

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

</body>
</html>
```

Click-n-reveal: **A nice AJAX example**

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
     $("button").click(function(){
    $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
          if(statusTxt == "success")
               alert("External content loaded successfully!");
          if(statusTxt == "error")
```

```
                alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
     });
});


</script>
</head>
<body>


<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>


<button>Get External Content</button>


</body>
</html>
```

jQuery GET, POST

- Two commonly used methods for a request-response between a client and server are: GET and POST.
  - GET – Requests data from a specified resource
  - POST – Submits data to be processed to a specified resource
- GET is basically used for just getting (retrieving) some data from the server. Note: the GET method may return cached data.
- POST can also be used to get some data from the server. However, the POST method NEVER caches data, and is often used to send data along with the request.

POST Example

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
```

```
<script>
$(document).ready(function(){
          $("button").click(function(){
          $.post("demo_test.jsp",
          {
          name: "Donald Duck",
          city: "Duckburg"
          },
          function(data,status){
           alert("Data: " + data + "\nStatus: " + status);
          });
          });
});
</script>
</head>
<body>
```

jQuery and Other JavaScript Frameworks

- jQuery uses the $ sign as a shortcut for jQuery.
- There are many other popular JavaScript frameworks, like: Angular, Backbone, Ember, Knockout, and more.
- What if other JavaScript frameworks also use the $ sign as a shortcut?
- If two different frameworks are using the same shortcut, one of them might stop working.
- The jQuery team have already thought about this, and implemented the noConflict() method.
- Below is an example of using noConflict()

noConflict() Example

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
<script>
$.noConflict();
jQuery(document).ready(function(){
          jQuery("button").click(function(){
          jQuery("p").text("jQuery is still working!");
          });
});
</script>
</head>
<body>
```

```
<p>This is a paragraph.</p>
<button>Test jQuery</button>

</body>
</html>
```

Video

Please watch the following video on Introduction to JavaScript and jQuery. This video exhibits the usage of jQuery to retrieve information from RSS feeds and display it on a web page.

Introduction to JavaScript and JQuery
TMU Video

Summary

jQuery is a fast, small and feature-rich Javascript library. jQuery enables developers to perform tasks with minimal JavaScript code.

Assessments

Each week, you'll find here reminders of assignments or labs that you should be working on. For Week 8:

- Lab 8: Integrating rich UI's with jQuery/JavaScript and RSS feeds (5% of the final grade) is due in 7 days, i.e., the end of day Friday of Week 8.

Reminder

You have a weekly Zoom session. The date/timing for your Zoom session can be found in the Course Schedule in the Course Outline. The link to the Zoom session will be provided separately to you by your instructor. The Zoom session will consist of 20–30 minutes of lecture, followed by questions and answers (Q&A), followed by a lab period. You may ask questions about the previous or current week's content or labs.

Please bring any questions to the weekly Zoom session. However, if you have any questions during the week outside of the Zoom session, you may post them in the Discussion Board. Your instructor may respond in the Discussion board or during the weekly Zoom session.

Click-n-reveal: **Where is the Discussion board?**

Click on the "Communication" area at the top main course menu, and select Discussions from the dropdown menu.

References

None for this week.