jQuery

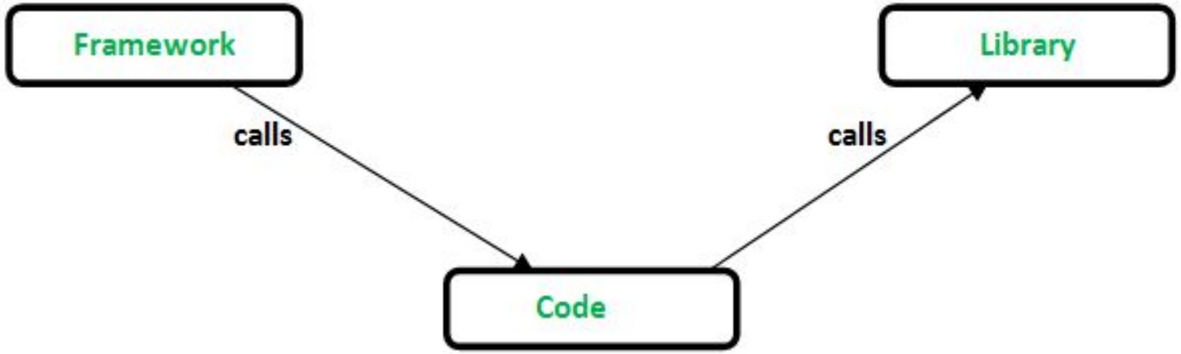
COEN 161

Libraries vs Frameworks

- Libraries
  - Provide helper functions/objects/modules to make coding easier and more reusable
  - Have a limited scope (e.g. Strings, DOM, etc.) making them less complex
  - Your code calls the library
- Framework
  - Provides open ended function definitions where you can write your own custom code
  - Can be made up of many libraries to extend functionality but increasing complexity
  - The framework later calls your code when it decides it is appropriate, this logic is usually built into the framework
  - This is called **inversion of control**

Libraries vs Frameworks

- With a library, we *control* the library functions from our code
- With a framework, the *framework* controls when your code is called



jQuery

- A JavaScript library made aimed at making it easier to use JavaScript in your website
- It is a "write less, do more" library that wraps a lot of complex JavaScript functionality into easy to use methods
  - For example, the syntax for DOM manipulation and AJAX* requests is a lot simpler
- Things jQuery can do
  - HTML/DOM manipulation
  - CSS manipulation
  - HTML event methods
  - Effects and animations
  - AJAX*
  - Utilities

***A**synchronous **J**avaScript **A**nd **X**ML, or AJAX is the idea of updating parts of a webpage by making XMLHttpRequest calls and using DOM methods to change the page based on the response

Why jQuery?

- Other than being really easy to learn and use…
- jQuery is used by a lot of big companies and is included in major JavaScript frameworks
- It is also cross-browser compliant, meaning you don’t have to worry about whether or not certain browsers support the JavaScript methods you are using, if it is a jQuery method, it will work on most major browsers

Adding jQuery

- You can download the jQuery `.js` file and add it to your page

```
<head>
    <script src="jquery-3.3.1.min.js"></script>
</head>
```

- You can include jQuery from a content delivery network, or CDN, by setting the `src` attribute of the `script` tag to the absolute path to the `.js` file

```
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
    </script>
</head>
```

jQuery Syntax

- When you include jQuery into your page, it adds two properties into the window object, `jQuery` and `$`
- Both properties have the same properties and methods, however, developers like to use `$` because it is shorter and faster to write
- The jQuery syntax is made for selecting

```
$(selector).action()
```

- `(selector)` is used to query your HTML just like `querySelectorAll()`
- `.action()` is a jQuery method to be performed on the selected HTML

jQuery Syntax

- The jQuery selector is CSS3 compliant, that means you can use any CSS selector to query your HTML
- 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

- In the world of DOM, we can use the following to run code once the page has loaded

```
window.onload = function() {
 alert( "welcome" );
};
```

- jQuery allows you to do something similar with its own syntax

```
$( document ).ready(function() {
    alert( "welcome" );
});
```

The Document Ready Event

- It is generally good practice to wait for the document to be fully loaded before running any JavaScript code
- We used to do this by adding our script at the bottom of the body
- With the ready event, we can add our code before the body, in the head
- The ready event is so useful, the jQuery team created a quicker syntax for it

```
$(function() {

    alert( "welcome" );

});
```

jQuery Selectors

- The jQuery selector is used to "find" and return elements based on their name, id, classes, types, attributes, values of attributes, etc.
- Examples

```
$("p") - selects all <p> elements

$(".test") - selects all elements with class="test"

$("#test") - selects the element with id="test"
```

jQuery Selectors

- Some DOM methods, such as querySelectorAll, returned *collections*

```
window.onload = function() {
    var myCollection = document.getElementsByTagName("p");
    var i;
    for (i = 0; i < myCollection.length; i++) {
        myCollection[i].style.display = "none";
    }
};
```

jQuery Selectors

- jQuery also returns collections when selecting elements, however they are a little more complex
- What is actually returned is a jQuery Object that wraps the collection in an object that provides additional properties and methods

```
$(document).ready(function(){

    $("p").hide();

});
```

jQuery Events

- jQuery objects also define some methods that allow you to handle HTML events easily

Mouse Events	**Keyboard Events**	**Form Events**	**Document/Window Events**
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

jQuery Events

- The syntax is similar to DOM events, but again made simpler

```
$("p").click();
```

- The event method takes a function that defines what should happen when that event is fired

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

  // action goes here!!

});
```

jQuery Events

- Compared to DOM method, this syntax is a lot simpler

```
var myCollection = document.getElementsByTagName("p");

for (var i = 0; i < myCollection.length; i++) {

    myCollection[i].onclick = function () { // action };

}

// in jQuery

$("p").click(function(){ // action });
```

jQuery Events

- jQuery also has an equivalent to `addEventListener` call `on`
- Like `addEventListener`, `on` lets you add multiple event handlers to the selected elements

```
$("p").on("click", function(){

    $(this).hide();

});
```

jQuery Events

- The on method also let’s define multiple types of events at once

```
$("p").on({
    mouseenter: function(){
        $(this).css("background-color", "lightgray");
    },
    mouseleave: function(){
        $(this).css("background-color", "lightblue");
    },
    click: function(){
        $(this).css("background-color", "yellow");
    }
});
```

jQuery CSS

- jQuery also makes it really easy to manipulate the CSS of elements
- It has the following methods which let us quickly change properties
    - addClass() - Adds one or more classes to the selected elements
    - removeClass() - Removes one or more classes from the selected elements
    - toggleClass() - Toggles between adding/removing classes from the selected elements
    - css() - Sets or returns the style attribute

jQuery CSS

- Consider the following CSS styles

```
.important {
    font-weight: bold;
    font-size: xx-large;
}
.blue {
    color: blue;
}
```

jQuery CSS

- We can select as many elements as we want and classes to them to change their styles

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

    $("h1, h2, p").addClass("blue");

    $("div").addClass("important");

});
```

jQuery CSS

- We can also add as many classes as we want in one function call

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

    $("#div1").addClass("important blue");

});
```

jQuery CSS

- jQuery also makes it really easy to remove classes

```
$("button").click(function(){
    $("h1, h2, p").removeClass("blue");
});
```

- Using DOM methods it would look something like this, **for one element**

```
var element = document.getElementById("myDIV");
element.classList.remove("mystyle");
```

jQuery CSS

- The jQuery `css()` method is a quick way to get and set specific css values

```
// returns the value of background-color

$("p").css("background-color");

// sets the background-color to yellow

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

jQuery CSS

- The `css()` method also lets you set multiple styles with one call

```
$("p").css({
    "background-color": "yellow",
    "font-size": "200%"
});
```

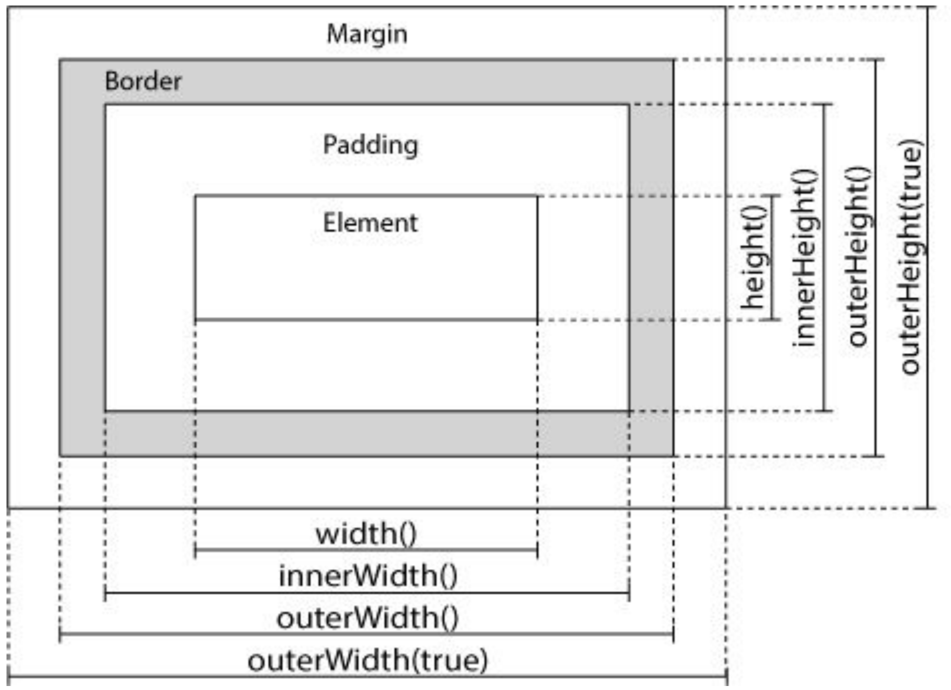
jQuery CSS

- The `css()` method is equivalent to the `getComputedStyle()` method, which returns the actual styles of the elements
- However, `getComputedStyle()` is not available in all browsers, but `css()` is available in the jQuery library regardless of browser

jQuery Dimensions

- jQuery has a set of methods to easily set the dimensions of elements
  - `width()`
  - `height()`
  - `innerWidth()`
  - `innerHeight()`
  - `outerWidth()`
  - `outerHeight()`

jQuery Dimensions



jQuery Dimensions

- The width and height methods sets or returns the width or height excluding padding, border and margin
- The innerWidth and innerHeight methods include padding
- The outerWidth and outerHeight methods include padding and border

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

    $("#div1").width(500).height(500);

});
```

jQuery Dimensions

- Example

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

    var txt = "";

    txt += "Document width/height: " + $(document).width();

    txt += "x" + $(document).height() + "\n";

    txt += "Window width/height: " + $(window).width();

    txt += "x" + $(window).height();

    alert(txt);

});
```

jQuery DOM Manipulation

- jQuery also comes with many methods for manipulating the DOM
- To get content from the document you can use the following methods:
    - text() - Sets or returns the text content of selected elements
    - html() - Sets or returns the content of selected elements (including HTML markup)
    - val() - Sets or returns the value of form fields

jQuery DOM Manipulation

- Examples

```
$("#btn1").click(function(){
    alert("Text: " + $("#test").text());
});

$("#btn2").click(function(){
    alert("HTML: " + $("#test").html());
});

$("#btn1").click(function(){
    alert("Value: " + $("#test").val());
});
```

jQuery DOM Manipulation

- To get attributes use the `attr()` method

```
$("button").click(function(){
    alert($("#w3s").attr("href"));
});
```

- Compared to the native DOM methods

```
window.onload = function() {
    alert(document.getElementById("w3s").getAttribute());
};
```

jQuery DOM Manipulation

- jQuery also lets you **set** content using the same methods used for get
  - text() - Sets or returns the text content of selected elements
  - html() - Sets or returns the content of selected elements (including HTML markup)
  - val() - Sets or returns the value of form fields

jQuery DOM Manipulation

- Examples

```
$("#btn1").click(function(){
    $("#test1").text("Hello world!");
});

$("#btn2").click(function(){
    $("#test2").html("<b>Hello world!</b>");
});

$("#btn3").click(function(){
    $("#test3").val("Dolly Duck");
});
```

jQuery DOM Manipulation

- Alternatively, these set methods also take what is called a **callback function**
- Callback functions are a function expression passed as an argument that gets invoked by the library method
- These functions are commonly expected to return a value that the library function will then use to perform some action

```
$("#btn1").click(function(){
    $("#test1").text(function(i, origText){
        return "Old text: " + origText + " New text: Hello world!
        (index: " + i + ")";
    });
});
```

jQuery DOM Manipulation

- For the set methods, the callback function takes two parameters, the index of the current element, and the original text of the element.
- The callback returns the new value to set for the element

```
$("#btn2").click(function(){

    $("#test2").html(function(i, origText){

        return "Old html: " + origText + " New html: Hello <b>world!</b>

        (index: " + i + ")";

    });

});
```

jQuery DOM Manipulation

- To set attributes we can just pass a second parameter to `attr()` with the value we want to set for the attribute

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

    $("#w3s").attr("href", "https://www.w3schools.com/jquery/");

});
```

jQuery DOM Manipulation

- The attr() method also lets you set multiple attributes at once by passing an object as a parameter.

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

    $("#w3s").attr({

        "href" : "https://www.w3schools.com/jquery/",

        "title" : "W3Schools jQuery Tutorial"

    });

});
```

jQuery DOM Manipulation

- You can also pass a callback to the attr() method that, just like the other set methods, takes two parameters, the index of the element and the original value, and returns the new value for that attribute

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

    $("#w3s").attr("href", function(i, origValue){

        return origValue + "/jquery/";

    });

});
```

jQuery DOM Manipulation

- Adding new content to the document can be done by using the following methods
    - append() - Inserts content **at the end** of the selected elements
    - prepend() - Inserts content **at the beginning** of the selected elements
    - after() - Inserts content **after** the selected elements
    - before() - Inserts content **before** the selected elements

jQuery DOM Manipulation

- Examples

```
$("p").append("Some appended text.");

$("p").prepend("Some prepended text.");

$("img").after("Some text after");

$("img").before("Some text before");
```

jQuery DOM Manipulation

- We can insert more than just text using these methods, in fact we can insert an infinite number of elements generated using text/HTML, jQuery, or even DOM methods

```
function appendText() {

    var txt1 = "<p>Text.</p>";               // Create element with HTML

    var txt2 = $("<p></p>").text("Text.");   // Create with jQuery

    var txt3 = document.createElement("p");  // Create with DOM

    txt3.innerHTML = "Text.";

    $("body").append(txt1, txt2, txt3);      // Append the new elements
}
```

jQuery DOM Manipulation

- Examples

```
function afterText() {

    var txt1 = "<b>I </b>";                          // Create element with HTML

    var txt2 = $("<i></i>").text("love ");     // Create with jQuery

    var txt3 = document.createElement("b");    // Create with DOM

    txt3.innerHTML = "jQuery!";

    $("img").after(txt1, txt2, txt3);         // Insert new elements after <img>

}
```

jQuery DOM Manipulation

- The two main methods for removing elements with jQuery are:
  - remove() - Removes the selected element (and its child elements)

    ```
    $("#div1").remove();
    ```

  - empty() - Removes the child elements from the selected element

    ```
    $("#div1").empty();
    ```

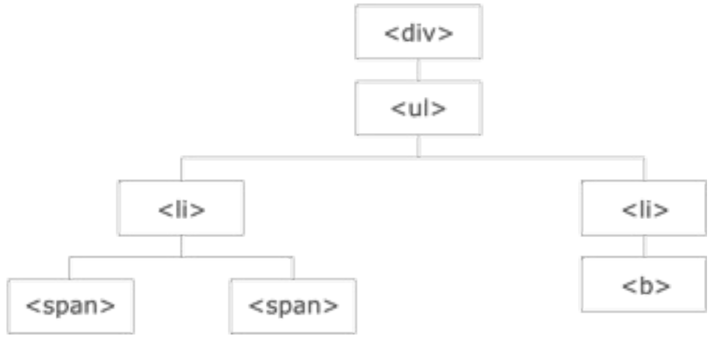
jQuery DOM Manipulation

- The `remove()` method can also accept one parameter, a selector, that lets you filter out which elements should be removed from the original selection

```
// remove all <p> elements with class="test"

$("p").remove(".test");

// remove all <p> elements with class="test" and class="demo"

$("p").remove(".test, .demo");
```

jQuery Traversing

- Traversing, in the context of a DOM, means to “move through” elements based on their relation to other elements
- jQuery lets you follow any of the DOM tree relationships in the native DOM methods, such as parents, children, and siblings



jQuery Traversing

- The following methods can be used to select any ancestors for the selected elements
  - parent()
  - parents()
  - parentsUntil()

jQuery Traversing

- The parent method returns the direct parent of all the selected elements

```
$(document).ready(function(){

    $("span").parent();

});
```

jQuery Traversal

- The parents method returns all of the ancestors all the way to the root element (<html>)

```
$(document).ready(function(){

    $("span").parents();

});
```

- You can also filter which ancestors get returned by passing a selector

```
$(document).ready(function(){

    $("span").parents("ul");

});
```

jQuery Traversal

- The parentsUntil method returns all the ancestors between the selected element and the element specified by the selector passed into the method

```
$(document).ready(function(){

    $("span").parentsUntil("div");

});
```

jQuery Traversal

- To select the children of an element, you can use the following methods:
    - children() - returns the immediate children of the selected elements

```
$(document).ready(function(){

    $("div").children();

});
```

    - find() - returns all descendants of the given element all the way down to the last descendant

```
$(document).ready(function(){

    $("div").find("*");

});
```

jQuery Traversal

- You can filter out children and descendants using a selector

```
$(document).ready(function(){
    $("div").children("p.first");
});
$(document).ready(function(){
    $("div").find("*");
});
```

jQuery Traversal

- To access siblings, jQuery provides the following methods
  - siblings()
  - next()
  - nextAll()
  - nextUntil()
  - prev()
  - prevAll()
  - prevUntil()

jQuery Traversal

- The siblings method returns all the sibling of the selected element

```
$(document).ready(function(){

    $("h2").siblings();

});
```

- You can also filter the results

```
$(document).ready(function(){

    $("h2").siblings("p");

});
```

jQuery Traversal

- The next and prev methods return the next or previous sibling respectively

```
$(document).ready(function(){
    $("h2").next();
});
$(document).ready(function(){
    $("h2").prev();
});
```

jQuery Traversal

- The nextAll and prevAll returns all the elements following or preceding the selected elements, respectively

```
$(document).ready(function(){

    $("h2").prevAll();

});

$(document).ready(function(){

    $("h2").prevAll();

});
```

jQuery Traversal

- The nextUntil and prevUntil return all the elements between the selected elements and either after or before until the second selected element is reached

```
$(document).ready(function(){

    $("h2").nextUntil("h6");

});

$(document).ready(function(){

    $("h2").nextUntil("h6");

});
```

jQuery Traversal

- We’ve seen filtering built into some of the jQuery methods
- However, if a method doesn’t support filter, or you want to filter out a collection, the most basic filtering methods are `first()`, `last()` and `eq()`, which allow you to select a specific element based on its position in a group of elements.
- Additionally methods like `filter()` and `not()` let you specify some criteria to determine if elements should be kept or not

jQuery Traversal

- Examples

```
$(document).ready(function(){
    $("div").first(); // selects the first <div> element
});

$(document).ready(function(){
    $("div").last(); // selects the last <div> element
});

$(document).ready(function(){
    $("p").eq(1); // selects the second <p> element
});
```

jQuery Traversal

- Examples

```
$(document).ready(function(){

    // returns all <p> elements with class name "intro"

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

});

$(document).ready(function(){

    // returns all <p> elements without class name "intro"

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

});
```

jQuery Effects

- With jQuery, you can `hide()` or `show()` elements

```
$("#hide").click(function(){
    $("p").hide();
});

$("#show").click(function(){
    $("p").show();
});
```

jQuery Effects

- Hide and show take optional parameters that let you control the speed at which an element is hidden ("slow", "fast", or milliseconds), and a callback function that gets called after the element is hidden

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

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

- Example
  ```
  $("button").click(function(){
      $("p").hide(1000);
  });
  ```

jQuery Effects

- jQuery also lets you `toggle()` between hidden and showing

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

    $("p").toggle();

});
```

- Toggle also takes the same optional parameters

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

jQuery Effects

- jQuery has a series of methods that let you fade elements
- They all take the same optional parameters as hide and show*

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

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

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

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

jQuery Effects

- The method `fadeTo()` let’s fade until a given opacity
- The first two parameters are required, and the callback is optional

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

    $("#div1").fadeTo("slow", 0.15);

    $("#div2").fadeTo("slow", 0.4);

    $("#div3").fadeTo("slow", 0.7);

});
```

jQuery Effects

- The slide methods let you "slide" an element in and out of the page
- They all take the same optional parameters as hide and show

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

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

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

jQuery Effects

- You can write your own custom animations using the `animate()` method

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

- Example

```
$("button").click(function(){
    $("div").animate({left: '250px'});
});
```

jQuery Effects

- You can animate using multiple values
- Example

```
$("button").click(function(){
    $("div").animate({
        left: '250px',
        opacity: '0.5',
        height: '150px',
        width: '150px'
    });
});
```

jQuery Effects

- Relative values

```
$("button").click(function(){
    $("div").animate({
        left: '250px',
        height: '+=150px',
        width: '+=150px'
    });
});
```

jQuery Effects

- And predefined values like "show", "hide", or "toggle"

```
$("button").click(function(){
    $("div").animate({
        height: 'toggle'
    });
});
```

jQuery Chaining

- jQuery supports chaining, meaning you can call multiple jQuery methods after one another

```
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
```

jQuery AJAX

- AJAX = Asynchronous JavaScript and XML
- There are several AJAX methods in jQuery

```
$(selector).load(URL,data,callback);

$.get(URL,callback);

$.post(URL,data,callback);
```

jQuery AJAX

- The load method loads data from the server and puts the response in the selected element
- The URL is required, but data and callback are optional

```
$("#div1").load("demo_test.txt");

// loads the following text

<h2>jQuery and AJAX is FUN!!!</h2>

<p id="p1">This is some text in a paragraph.</p>
```

jQuery AJAX

- The callback takes the following parameters
  - responseTxt - contains the resulting content if the call succeeds
  - statusTxt - contains the status of the call
  - xhr - contains the XMLHttpRequest object

```
$("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);
    });
});
```

jQuery AJAX

- The get and post methods work similarly to load, except they don’t operate on a selected element
- The get method uses a GET HTTP request to load data from the server
- The post method uses a POST HTTP request to submit data to the server

jQuery AJAX

- Example - `$.get()`

```
$("button").click(function(){
    $.get("demo_test.asp", function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

jQuery AJAX

- Example - `$.post()`

```
$("button").click(function(){
    $.post("demo_test_post.asp",
    {
        name: "Donald Duck",
        city: "Duckburg"
    },
    function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});
```

Resources

https://www.geeksforgeeks.org/software-framework-vs-library/

https://learn.jquery.com/about-jquery/how-jquery-works/

https://www.w3schools.com/jquery/default.asp

https://www.w3schools.com/jquery/jquery_selectors.asp

https://www.w3schools.com/jquery/jquery_ref_effects.asp

https://www.w3schools.com/jquery/jquery_ref_ajax.asp