Week 12: JS Part IV/JQuery

Agenda

- Overview
  - Previous Participation Questions/Responses
  - What's Due?
- Wrapping up JS (Lecture/Class Activity)
- Intro to JQuery
  - What is it?
  - How do we use it on our pages?
  - JQuery Syntax
- JQuery Challenges (CW/HW)
- Participation Questions (4)
- Next Week

Overview

Last week, we learned more about targeting elements on a page with JS. We covered things like getElementsByClassName, querySelector, and querySelectorAll. This week, we will do a final review of the concepts we've learned so far and then go over JQuery and how it can be used on a page.

Previous Participation Questions/Responses
https://forms.gle/q58pbSjL2hqPjCc67

This link has the responses to the fun questions from previous weeks. Feel free to look over the responses from myself and your peers.
https://docs.google.com/spreadsheets/d/1QLoQzbO-Er7ARIECl0aPvY1lKUyZTdYeuwsn4g0XEAk/edit?usp=sharing

What's Due?

- Participation Questions (4)
- JQuery Challenges (CW/HW)

Wrapping up JS (Lecture/Class Activity)

Over the past 3 weeks, we have covered several concepts in JavaScript. I will go over some of them briefly again as a quick review before you start learning about JQuery.

JavaScript is the third major component of client-side web development. When used correctly, it can greatly enhance the experience for users. We can add it to our pages in three ways:

Inline – The JavasScript code is placed inside of the opening tag of an HTML element. A typical example is a button which uses an onclick attribute and then provides a function as the value of that attribute (https://www.w3schools.com/tags/ev_onclick.asp).

Internally – Code is placed inside of <script></script> tags like internal CSS when we place styles inside of <style></style> tags in our document.

Externally – Code is placed inside of a separate file with a ".js" extension. To link to this file, we would still use the <script></script> tags but we must include a "src" attribute in the opening script tag. This src attribute behaves like the href attribute for the link tag when linking CSS.

<script src="javascript-file.js"></script>

The **best practice** for implementing JavaScript is to link to it externally. It is always good to separate HTML, CSS, and JS from each other, so we can maintain them more easily.

Commenting
Like HTML and CSS, we can add comments to our JS to help explain things to anyone reading our code. To add a comment, you can do it in one of two ways:

```
// This is a single-line comment

/* This is another comment
Which can span multiple lines
*/
```

Variables
You can think of these as boxes which contain values. When we create a variable, we can **assign** it a value. Later in our code, we can make a reference to that variable's name and the value we assigned to it would be used.

```
var myName = "Chris";
alert(myName);
```

To create a variable, we use the "var" keyword which signifies that we are creating a new variable, followed by the name of the variable. Afterward, we can use a single equal sign (assignment operator) to say that the value provided on the right is being assigned to the variable name on the left.

In the example above, I am creating a variable called myName and assigning it a value of a string with the text "Chris" in it. I then use the alert function to output the myName variable. This code would create a pop-up box which says "Chris" inside of it.

Variables can have multiple types, but three very common ones are strings, numbers, and Booleans. Strings are just text surrounded by quotation marks such as "Chris" in the previous example. Numbers aren't surrounded by quotes and are simply typed out as the number itself. Booleans can only have a value of either "true" or "false."

If/Else Statements
These are statements which allow us to execute only certain pieces of code based on some sort of condition provided. The if part is the first condition we check and whatever we type into the parenthesis must evaluate to either true or false. If it is true, the code inside of the first set of

curly brackets is executed. If it isn't true (else), we execute some other code. We can also use elseif statements to add as many extra conditions as we want.

In the example below, I create a variable called myAge and assign it a value of 30. I then use an if/else statement to check if that value is greater than or equal to 18. If the age is greater than or equal to 18, the first set of code is executed. If it isn't, then we execute the code inside of the else portion. Since myAge is 30 and is greater than or equal to 18, an alert saying "You can vote!" would appear.

```
var myAge = 30;
If (myAge >= 18) {
        alert("You can vote!");
} else {
        alert("You cannot vote, sorry.");
}
```

Using Loops to Repeat Code

If we want to perform some action several times, we could type the same line of code over and over again in our JS. However, this would be very inefficient, and loops help us accomplish this with a lot less code.

Every loop requires 3 components to it which can be simplified as shown below:

1. Where to start
2. What is the condition to check again (where to end)
3. How much we increase/decrease our loop value

```
for(var x = 0; x <= 5; x++) {
        console.log(x);
}
```

In the for loop above, the first part of the for loop describes where we start (x = 0). The second part describes what condition we are checking against before we execute our loop code (is x less than or equal to 5). The third part of the loop determines how much we increase our value each time the loop is executed (x++ is shorthand for saying increase by 1 every time).

```
var x = 0;
while(x <= 5) {
        console.log(x);
        x++;
}
```

Both loops above perform the exact same function. However, the main difference is the placement of the information for where to start and how much to increase/decrease our loop value.

Arrays

These can be thought of as a box which holds other boxes inside of it. A plain variable can be used to store one value at a time. With arrays, we can store multiple values of varying types in one container.

Arrays use square brackets to contain all of the values. The brackets hold the values, and the commas separate them. Each value can be referred to by its location in the array which is known as an **index number**. The first item in an array always has an index number of 0.

var theFourElements = [“water”, “earth”, “fire”, “air”];

In our array above, we have stored 4 string values. To reference any value in the array, you have to write out the name of the array followed by square brackets. The content inside of the square brackets must evaluate to a number so that it can pull up the specific item in the array.

theFourElements[0]; // Refers to the first item in the array
theFourElements[theFourElements.length – 2] // References “fire” because the length of the array is 4…4-2 equals 2 so it evalutes to theFourElements[2] which is fire.

Functions
These are blocks of code which are placed inside of a container. To execute that code, we would need to reference the name of the container (the function name).

```
var fullName = “Chris J. Velez”;
var shoutMyName = function() {
        alert(fullName);
}
shoutMyName();
```

In the code above, we create a variable called fullName and assign it a value of “Chris J. Velez.” After this we create a variable called shoutMyName and assign it to a function. Inside of that function, we use an alert to output the fullName variable. After the closing curly bracket for the function, we call/reference the function by typing out the name of it followed by parenthesis.

Functions generally don’t execute on their own which is why we call/reference them. Functions can take in arguments so that we can increase the functionality of them.

```
var addTwoNumbers = function(firstNum, secondNum) {
        var sum = firstNum + secondNum;
        alert(sum);
}
addTwoNumbers(5, 10);
```

In the example above, we create another function called addTwoNumbers. This function takes in two parameters/arguments which are called firstNum and secondNum. Inside of the function, we use those two variables to do some basic math by adding them together and assigning that value to a variable called sum. After this, we use an alert to output the sum.

If we wanted to execute this function, we would call/reference it just like the first one. However, we would need to provide two values since the function utilizes arguments. This is why we reference the function and provide the numbers 5 and 10. The result of this would be an alert box appearing with the value of 15.

Working with the DOM
JavaScript works based on the idea of a Document Object Model. Each tag on your page can be considered an object. These objects have various properties/functions we can use to manipulate them. If we had a paragraph with an id of "para," we could use that id to target the para tag and do things with it.

This is a sentence inside of a para tag.

var para = document.getElementById("para");

In the line above, we store the HTML Object (p tag) inside of a variable called para. Once we have this variable, we can do things like change the text inside of it, add styles, remove it, etc.

para.innerHTML = "New content."; // This would replace the text
para.style.color = "red"; // This would change the color of the text to red

With the objects on the page, we can use **events** to trigger some sort of function. Instead of creating a variable for a function, we can assign that function to an event on an element on our page. In the example below, we use an onclick event to trigger a function which uses an alert inside of it.

```
<button id="btn">Click Me!</button>
document.getElementById("btn").onclick = function() {
        alert("The button was clicked!");
}
```

Other events include double-clicking, hovering, and key presses. You can combine various events on elements with the concepts explained earlier to create robust interactive components on your page.

Intro to JQuery (Lecture/Class Activity)

What is it?
JQuery is a library based on JavaScript that allows users to implement the same features they would in JavaScript but with less code. Things like DOM manipulation, event handling, animation, and more can be done more easily.

It comes in multiple flavors such as JQuery core, JQuery UI (for widgets like accordions, tabbed layouts, etc.), and JQuery Mobile (for use with creating responsive websites). The one we will focus on for this course is JQuery core, but you can use any of them if you'd like except for JQuery Mobile (your responsive design should be done with media queries).

How do we use it on our pages?
You can use JQuery on your page by either downloading a copy of the JQuery code and linking it like regular external JS or by using JQuery hosted on a Content Delivery Network (CDN). For the purposes of this course, you can do either. The main advantage to using JQuery hosted by Google or some other platform is that a user won't have to download the code again if it was already downloaded from another website they visited.

One downside to JQuery is that it adds a bit of bloat to your code. Usually, to use any one feature, you still must download the whole library which can add a lot of unnecessary code to your page. In most instances, you can build the feature with plain JS, but if it's more complex, JQuery can be helpful. Despite the code bloat, JQuery is still a very small library and with modern bandwidth limits, it might not be an issue to have a local copy.

The CDN I recommend/will be using in my class examples comes from Google and is displayed below. You would add this directly before your own script tag to use it as depicted.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script src="script.js"></script>
```

JQuery Syntax
One major difference in JQuery is how you can select items on the page. Rather than use the standard document.getElementById or document.querySelector methods, we can simply use a dollar sign and specify the html element, class, id, attribute, etc.

With regular JS, we might use the following code to add a click event to a button:

```
<button id="btn">Submit</button>
document.getElementById("btn").onclick = function() {// function code }
```

In JQuery, we can simply say:

```
$("#btn").click(function() {//function code});
```

The dollar sign is built in as a selector for all types of stuff. The only thing you need to do is specify by element, class name, id name, etc. The selectors work the same as they do in CSS so if you want to select a class, you must identify it with a period. The same goes for ids with the pound sign (#). There are two basic examples below but JQuery is very robust with the criteria it can use to select elements.

Selecting a class of para inside a div: $("div.para")
Selecting a list item inside of an unordered list: $("ul li")

When using the general selector for JQuery, it will behave similarly to the document.querySelectorAll() method in that it will return a list of elements if there are multiple matches to your query. You could then loop through that list if you needed to do so.

There's so much more to JQuery but rather than explain the technical details here, I've provided links to various resources that will demonstrate some of its capabilities. You don't have to read through all of them, feel free to skim through them to see what features are available. The class example contains my own code in addition to more explanations that will be relevant for this week's assignment.

Class Example & Online Resources
This example contains content which demonstrates how to use JQuery on a web page. It shows how to link it to our HTML and then provides examples of basic functions and click events.

https://www.albany.edu/~cv762525/cinf201/examples/jquery-example.html

To view all other class examples, visit this link and click whatever you want to see. The js folder contains the various JS files used for the examples.

https://www.albany.edu/~cv762525/cinf201/examples/

You **do not** have to read through every single link below. They are to be used as needed if you want additional coded examples. JavaScript has many features, and we will only scratch the very surface of them in this course. However, digging deeper is good if you want to implement advanced features on your pages.

Online Resources

- https://jquery.com/ (General JQuery Website)
- https://api.jquery.com/ (Technical details for JQuery core)
- https://learn.jquery.com/ (Learn about JQuery core)
- https://www.tutorialspoint.com/jquery/jquery-overview.htm (Multiple sections)

JQuery Challenges (CW/HW)

Download the "JQuery-Challenges.zip" folder from Blackboard under today's Lecture Notes or the Assignment folder. Extract everything from inside of this zip folder to some place where you will be able to find them easily. Inside of the zip folder will be an HTML file.

Your task is to read the three challenges displayed on the HTML page and add JQuery-based code as needed to complete them. Underneath all challenges (before closing body tag), there will be one <script></script> tag with some code already in it. Place your code for the challenges between the curly brackets of the code so that your assignment works properly.

Review the example from this week (listed above) as it contains code that can be used for these challenges.

To view all other class examples, visit this link and click whatever you want to see. The js folder contains the various JS files used for the examples.

https://www.albany.edu/~cv762525/cinf201/examples/

But Professor, I Don't Know Where to Start
If you're not sure where to start at all, this section was made for you! I've listed out basic steps that you can take below for your convenience. Follow them carefully and you should be able to get the assignment started.

- Download the "Week-12-Lecture-Notes.zip" folder from Blackboard

- Take the “jquery-challenges.html” file and place it in a folder you can find.
    - Create a “Week-12” folder in your “Documents” folder or some other folder you can access easily
    - Place your html file in this newly created “Week-12” folder
- Open the “jquery-challenges.html” file with a text editor of your choice
- Open the “jquery-challenges.html” file with a web browser and read the instructions
- Review the 5 challenges listed on the page and add code in the <script></script> tags of the HTML file to complete the challenges
- I would start with setting up each function individually. In total, there are 4 functions triggered by clicks, and one function triggered by a keypress
- Make sure to pay close attention to the tips I’ve provided in the comments of the HTML file

Your web page is **due November 18th at 6pm.** To submit the work for this exercise, go to Blackboard → Course Materials → Lecture Notes for this week’s class. You can also visit the “Assignments” folder and the submission area will be titled “JQuery Challenges” **You must submit a live link to your web page**. The work submitted will be evaluated based on the rubric explained below.

JQuery Challenges – 10pts

- Live link submitted – 1pt
- Your code is neatly organized – 1pt
- Each challenge is complete – 8pts total or 1.6pts for each challenge

Participation Questions 11/11

To receive credit for participation in today’s class, please answer all the questions in the Google Form linked below. If you don’t want to answer the final question (fun), simply put N/A (not applicable) for your answer and that will count.

This Week’s Participation Questions:
https://forms.gle/hQ8u8b9tWKaMWm4q9

Previous Participation Questions/Responses
https://forms.gle/q58pbSjL2hqPjCc67

This link has the responses to the fun question from previous weeks. Feel free to look over the responses from myself and your peers.
https://docs.google.com/spreadsheets/d/1QLoQzbO-Er7ARIECl0aPvY1lKUyZTdYeuwsn4g0XEAk/edit?usp=sharing

Next Week

We will briefly go over responsive design and how it can be used to enhance a user’s experience on a website. Specifically, we will learn about the viewport meta tag and media queries. If you’d like to get ahead on the material, feel free to skim through the links below.

1. https://www.pewresearch.org/internet/fact-sheet/mobile/
2. https://developers.google.com/web/fundamentals/design-and-ux/responsive
3. https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

Besides this, we will begin to choose slots for the Final Project Review on the last day of class. You will choose one slot by the end of class and anyone not present will be randomly assigned to the remaining time slots.