Week 9: Intro to JavaScript (JS)

Agenda

- Overview
  - Previous Participation Questions/Responses
  - Midterm Reminder
  - What's Due?
- Introduction to JavaScript (Lecture/Class Activity)
  - What is it?
  - How do we use it on our pages?
  - Basic Concepts
    - Commenting
    - Variables
    - Data Types
    - Outputting our Info
    - If/Else Statements
    - Loops in JavaScript
    - Functions
- JS Challenges (CW/HW)
- Participation Proof
- Next Week

Overview

Last week, we finished learning about positioning using floats and flexbox in CSS. This week, we will learn about JavaScript (JS) and how we can add it to our pages to provide some basic interactivity. At the end of this week, students should be able to link JS to their pages and create output to the page or console.

Previous Participation Questions/Responses

There were no participation questions from last week. Instead, you had to demonstrate that you made progress on an assignment or something related to the Final Project for this course.

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

Midterm

The **Midterm is due on October 28th at midnight**. Please make sure you have started working on it or start soon. Overall, you should be creating 3 pages of HTML that are linked together. You can do this with the <a> tag's href attribute. There should also be one page of externally placed CSS for all three pages of HTML. For more details, please review the "Final Project Description" document available on Blackboard → Final Project.

You don't need to have perfectly placed or finalized content. However, your pages should be mostly put together. Major headings should be used to describe content. If you haven't thought

of text to use, feel free to Google “Lorem Ipsum” for place holder text. You can also use place holder images by Googling those as well. The most important thing is that you have some sort of content placed for your Midterm.

The basic-page-example.html file provided earlier in the semester is a good an example of an acceptable page. Feel free to use some of the CSS I’ve provided (row class, float items to the left, give them a width) to help style your pages. The overall structure should be original and your own work though.

For the Midterm submission, you’ll submit 3 links in total. Each link should point to a different page of your midterm. When I check your work, I should be able to go from page to page easily by using links on your pages.

What’s Due?

- Participation Proof
- JS Challenges (CW/HW)
- Midterm

Introduction to JavaScript (Lecture/Class Activity)

What is it?

JavaScript is a programming language which can be used to add interactivity to any web page. JS serves as the third main technology to web development on the front-end side. It is its own language but there are many frameworks built on top of it which enhance its features and make coding easier. We can use it on the client side or server side (backbone.js, node.js, etc.), but for the purposes of this course, we will focus solely on the client-side applications of JS.

JS works based on the premise of **objects** and **functions**. In the lines below, I demonstrate some pseudo examples of objects along with some functions they could have.

Cat Object (Meow, Lick, Eat, Scratch, Purr)
Person Object (Talk, Run, Read, Write, High-Five)
Car Object (Accelerate, Brake, Turn, Lock, Park)

The contents in the parentheses are functions that can be performed by the objects listed on the left. Each of those functions can take what are known as **arguments** or **parameters**. If a cat was to Meow, some “arguments” could be the volume or the frequency of the meow. We could create pseudo code to represent this as show below. The first “argument” is the volume and the second one is the frequency. I’ve included other examples as well to demonstrate how real-life items can be seen as objects with functions and parameters. I use periods to separate the object from the function and then parentheses are used for the parameters.

Cat.Meow(“high”, “medium”); // A cat performing the Meow function with the volume argument set to high and the frequency argument set to medium

Person.Talk(“low”, “spanish”); // A person performing the Talk function with the volume argument set to low and the language argument set to Spanish

Car.Turn(“left”); // A car performing the Turn function with the direction argument set to left.

JavaScript has many functions that can be used with the page or objects on the page. These functions sometimes take in parameters and are usually trigged by some sort of **event** on the page. In the examples above, I have shown you objects that perform functions. In the one below, I show what kind of events can be used to trigger functions on a web page.

Objects on a Page (Clicked, Hovered Over, Changed, Loaded, Key Pressed, etc.)

When any of the actions in the parentheses are performed on or with an object on a page, we can trigger a function to create some action. We can provide arguments to the function to change the outcome or result. Before I can provide examples of some JS usage based on events, I will show you how to link it to your HTML pages.

How do we use it on our page?
Adding JS to a page is simple to do. Like CSS, we can add it **inline, internally, or externally**. I’ve provided examples of each type in the PowerPoint notes, but I will explain them as well below. There’s also a link below which demonstrates linking JS using all three methods.

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

Inline
When we use inline JS, we place actual JS code inside of an HTML elements attribute. We might use something like the onclick attribute and set it equal to a function we define in our JS code. **I wouldn’t recommend this as it isn’t good practice.** In this example, clicking the button produces a popup message that says “Hello World!”

Remember how I said we can trigger events on objects in our HTML pages? The button is the object and the onclick event is our trigger to the alert function. The alert function accepts what is known as an **argument** or **parameter**. In this case, the parameter is a string with a value of “Hello World!”. We will learn about strings later but just know that without some sort of value, the alert wouldn’t output anything.

```
<button onclick="alert('Hello World!');">Click Me!</button>
```

Internally
Internally placed code is located inside of <script> tags which can be placed anywhere in the document. With internally placed code, our onclick function and all related code would be placed inside of the <script> tags. We can target the button many ways but one of the easiest ways to do so in JS is with the document.getElementById function.

With document.getElementById, we are targeting the “document” object and using its “getElementById” function. getElementById searches the whole document for the provided id value. The id value is known as the parameter or argument that we supply to the function. Our retrieved element is then linked to a function which contains our alert function.

It does the exact same thing as the button onclick code from above, but it is linked differently. The id attribute value must match the value provided to getElementById or it will not work.

```
<button id="myButton">Click Me!</button>
<script>
      document.getElementById("myButton").onclick = function(){
         alert("Hello World!");
      }
</script>
```

Externally

Externally placed code would be in an external JS file entirely. You would use a script tag with a source attribute to link your JS/HTML. Like what you do with CSS or any other linked source, you need to make sure the folder and file structure match up in your reference to it. With external JS, the HTML id attributes are used to hook the JS and HTML.

HTML File

```
<button id="myButton">Click Me!</button>
<script src="script.js"></script>
```

Script.js file – Doesn't contain HTML, just JavaScript

```
document.getElementById("myButton").onclick = function(){
   alert("Hello World!");
}
```

Best practice dictates that we place our JS in a **separate file** and place the <script></script> tags directly **before the closing body tag**. The reason we do that is because our HTML needs to load first. If we use a button for a JS function and the JS loads before the button, our JS won't link to it correctly and users may not be able to use your page as expected. Separating our code from our structure also helps keep our code tidy and easier to maintain in the long run.

Basic Concepts

This week, we will cover some basic concepts such as comments, variables, data types, and functions so that you can build some very basic interactive components on your web pages. Most of the output from the topics mentioned below will be displayed in the console. If you choose to replicate this code, make sure you check that place out for your output.

Comments

Like HTML and CSS, you can leave comments in your JS code to explain what is happening. To create a single-line comment, you can use two forward slashes ("//") or an asterisk and forward slash for multi-line comments ("/* */").

```
// This is a one-line comment
/* This comment spans
Multiple lines */
```

Variables

Variables are the foundation of JS and without them, we can't accomplish more complicated things like form validation, modals, image carousels, etc. Think of variables as boxes that store

data or pieces of information inside of them. Each box holds a value and then we can use the value later for whatever we want.

They typically get assigned values and those values are used to add interactivity on our page or reference elements. You can use any name you want for a variable with some exceptions. It can't begin with a number or be a reserved word in JavaScript (bool, break, etc.)

```
var x = 5;
```

The above line is called a JavaScript statement and each statement should end in a semi-colon. We start by declaring a variable called x. When we declare a variable, we always use lowercase "var".

The equal sign doesn't mean that x equals 5. Instead, that is an assignment operator. Operators are used to perform some function or calculation. A single equal sign is used to assign values. The line above means we are assigning a value of 5 to a variable called x. We could also just say "var x;" which would declare the variable but not give it a value. Please note that variable names are case sensitive so "var x" and "var X" would refer to two different variables.

Data Types

In that example, the value of 5 has a "type" which all values have. There are six "primitive types" in JS and they are as follows: Boolean, null, undefined, number, string, and symbol. The most common ones are Boolean, number, and string. Everything else besides that is an object and objects have methods to work with them (arrays, functions, etc.).

A **Boolean** value evaluates to either true or false. In this example, we have created two separate variables and assigned a value of true to one of them and a value of false to the other one.

```
var catsAreCool = true;
var mosquitosAreCool = false;
```

A **Number** type is straight-forward and just reflects an numerical value. You can use **mathematical operators** with numbers to add, subtract, multiply, or perform other mathematical functions.

```
var answerToEverything = 42;
```

A **String** is a bit of text and is always wrapped inside of quotation marks. String objects have properties or functions such as length, search, split, and indexOf.

In the first line of my example below, I declare a variable called myName and assign it a value of "Christopher." In the next line, I create another variable and assign it the value of myName.length. The length is property that can be used on any string to find its length. We use the period to separate an object and its property or function.

```
var myName = "Christopher";
var myNameLength = myName.length; // This value would be 11 (number type)
```

Unlike some other programming languages, JavaScript doesn't allow you to declare a variable's type. Instead, the browser engine determines what it is and goes from there which can lead to weird behaviors. If you were to try and add a string to a number type, it would treat both as a string and combine them. This is called "type-coercion" and you must be careful with your variable types when using them if you don't want unexpected behaviors to occur.

With these different types, we have operators which we can use to manipulate them. Some of the most common ones are the mathematical operators (*, /, +, -, %) and the concatenation operator ('+' which is used to combine strings). In the example below, I use the concatenation operator to combine strings, but I also add a space because our strings don't include them automatically.

```
var firstName = "Chris";
var lastName = "Velez";

var fullName = firstName + " " + lastName; // This would be "Chris Velez"
```

Outputting our Information

You can output information to the page in a number of ways. For this week, we will focus on using alerts and console.log(). Like the document object, the console is another object which is useful for doing testing on an HTML page. Log() is one of the methods of the console object and it can accept many things as a parameter.

For example:

```
console.log("My name is Chris Velez.");  // Outputs whatever is in the string
console.log("My name is " + fullName); // Outputs the same as the line above because we created a variable called fullName earlier.

var myAge = 30;
console.log(myAge); // outputs 30

alert(myAge); // Would create a popup with the value of the myAge variable. Alert is the function and the thing inside of the parenthesis (myAge) is the argument/parameter we provide.
```

If/Else Statements

Having variables is great but sometimes we need to check their values to have certain things occur on our page. Let's say we have a modal that appears which asks for a user's age. If they are under 21, they can't gain access to the website. If they are 21 or older, then they can enter (common for alcohol distribution websites). We can use **if/else** statements to do this.

```
var myAge = 30;
If (myAge < 21) {
        alert("Sorry, you are not old enough!");
} else {
        alert("You may enter!");
}
```

The if() portion checks to see if myAge < 21 is true. If it's true, then the code inside of the curly brackets is executed. If it is false, it goes to the next else statement (or else if assuming there are

multiple options). If the next statement is true, that will be executed. If none of them are true, none of the code gets used. The first part is in parenthesis and whatever is placed inside of there must evaluate to true or false. Each code block that we want to execute should be separate by curly brackets.

Loops in JavaScript

Another important feature of JavaScript is the ability to perform loops. If we wanted to list the numbers 1-100 in the console, we could say console.log(1); console.log(2); and so on. However, that isn't very efficient and will result in code bloat. We can accomplish the same result with loops. The two main types of loops are For and While loops. They both contain the same features but are syntactically different.

Each loop requires three parts: Where we are starting, where we are finishing/the condition to check against, and how much we are incrementing/decrementing our number by. In the for loop below, the first part of the loop is where we start (0), the second part is where we go up to (less than or equal to 10), and the third part is what we increment our number by (++ is a shortcut for saying +1).

```
for(var x = 0; x <=10; x++) {
        console.log(x);
}
```

We start with 0 and check the second part of the loop to see if it's true. Since 0 is less than or equal to 10 (evaluates to true), we will go inside the for loop and execute our code. After we execute our console.log line, we increase x by 1. The loop continues and since 1 is less than or equal to 10, it will run again. This continues until we reach the number 11. Since 11 is neither equal to, or less than 10, the loop will exit.

```
var x = 0;
while(x <= 10) {
        console.log(x);
        x++;
}
```

The main difference between this loop (while) and the for loop is that the initial number and incrementing part are in different parts of the loop. You initialize the number before the loop and increment it inside of the loop. While loops are useful when you aren't sure when your loop needs to end. For loops are more for when you know exactly when the loop should end. Always remember to give your loop a way to close itself. If we removed x++ from any of the loop examples above, x would always be less than or equal to 10. This would cause the loop to run forever and crash the browser.

Functions

The last thing I want to talk about is functions. Functions are blocks of code that are wrapped up in a container. The code inside of that container (curly brackets) isn't executed unless the function is called.

In the example below, I've created a function called "squareMyNum" which will take in one number as a parameter/argument and then it will return the square of that number (number times itself). After creating the function, I call it by using the name of the function and providing a value for the argument. The result is logged to the console since that was inside of our function. The console.log() method is useful because it allows us to test our values without putting anything on our page.

```
var squareMyNum = function(num) {
        var result = num * num;
        console.log(result);
}

squareMyNum(5);  // This would output 25 but only viewable in the console.
```

Functions don't need arguments to work but, in some cases, you'll find that arguments can help you with creating output or manipulating input. The content between the parentheses is where you can place your arguments/parameters. The area between the curly bracket is where you place the code you want to be executed when the function is called. You can pair functions with if/else if/else statements and other JS objects to create robust applications.

Another example of a function is shown below. We create a variable called likesCats and assign it a value of true. We then create a function called catLikerChecker which takes in one parameter called "feelings." Inside of this function, we use an if/else statement to use if the parameter provided is true. If it is, we alert the user to say "You like cats!!" and if it's false, we output "You don't like cats!!"

After creating the function, we call/reference the function by its name. We provide the value of "likesCats" as an argument and since it's true, "You like cats!!" would be the output.

```
var likesCats = true;
var catLikerChecker = function(catFeelings) {
        if (feelings = true) {
                alert("You like cats!!");
        } else {
                alert("You don't like cats!!");
        }
}
catLikerChecker(likesCats);
```

Online Resources

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.

- https://htmldog.com/guides/javascript/beginner/makingstuffhappen/ (connecting JS)
- https://javascript.info/types (data types)

- https://www.w3schools.com/js/js_string_methods.asp (string methods/functions)
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators (operators)
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/if...else (if/else statements)
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Loops_and_iteration (all kinds of loops)
- https://htmldog.com/guides/javascript/beginner/functions/ (functions)
- https://eloquentjavascript.net/ (Free eBook on JavaScript - super comprehensive – I would check out parts 1-3 for this week)
- https://www.w3schools.com/js/default.asp (Tutorials on everything – use with caution)
- https://www.javascript.com/ (More tutorials with examples)

JS Challenges (CW/HW)

Download the "JS-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. This week, there will be an HTML file, a "js" folder, and a JS file inside of this folder. Make sure you don't change the folder's structure so that the HTML/JS are linked correctly.

Your task is to add relevant JS to complete the challenges listed underneath the assignment description. Some of the challenges will display results in the browser (alert/prompt) while others will produce results in the console (Inspect Element and then go to Console tab). Make sure you check both places to ensure you have completed the assignment. The challenges should be completed exactly as I have them described below. Helpful code can be found in this week's JavaScript example page.

Please note that the JS and HTML are already linked for you. All you need to do is add JavaScript to the JS file. When you place them on FileZilla, you'll need to make sure the src attribute matches up with your folder structure. If you choose to place the HTML/JS files in the same folder, change your src attribute value so that it only references the JS file.

Challenges

1. Replicate the following code in your JS file and add operators between the numbers to make them evaluate to true. You should only be typing "console.log()" inside of your JS file. The numbers should go inside of the parentheses as I've depicted below. You'll need to use mathematical or comparison operators to get them to evaluate to true.
    a. console.log(4 5);
    b. console.log(6 3);
    c. console.log(0 0);
2. Create a for loop that logs the numbers 1-100 to the console. If the number is a multiple of 5, it should say "Multiple of 5" instead of the number. To do this one, you'll need to

use a for loop, an if/else statement, console.log(), and the remainder operator (%). To check if a number if divisible by another one, you can use the following:

if (x % 2 == 0) {if the number is divisible by 2, execute this code}

The output would look like this:

1
2
3
4
Multiple of 5
6…and so on

3. Convert the for loop in the previous challenge to a while loop and make it do the same thing. The output should look the same as the for loop output.
4. Create a variable called myAge and another called myName. Set their values to your age and your name respectively. Create an alert with your name and age using string concatenation. "My name is ___ and I am ___ years old" is what should be display in a popup on the page. Your name and age will replace the underlines.
5. Use a prompt which asks for the age of the user. If the user is above 21, an alert should appear which says "You are old enough to drink." If the user is under 21 (else), an alert should appear saying "No alcohol for you!" You will need to use a prompt, if/else statement, and an alert. https://www.w3schools.com/jsref/met_win_prompt.asp is a link that should help you with the prompt part.
6. Create a function called numberAdder that will add the numbers 1-100 together and log the result to the console. The answer will be 5050 if this challenge is done correctly. I would solve this by creating a variable and assigning it a value of 0. I'd then create a for or while loop which went up until 100. Each time the loop executes, you want to add the number in the loop to the variable outside of the loop.

numberAdder(); // This would output "5050" or "Adding the numbers 1-100 together results in 5050."

7. Create a function called boxVolume that will calculate the volume of a 3-dimensional box. The formula for volume is length x width x height. Your function will need to take in three arguments and then perform some math. The function should log the volume of the box to the console.

boxVolume(3, 4, 5); // This would output ("Volume of your box is 60.");

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-9-Lecture-Notes.zip" folder from Blackboard

- Take the “js-challenges.html” file and “js” folder and place them in a folder you can find.
    - Create a “Week-9” folder in your “Documents” folder or some other folder you can access easily
    - Place the HTML file and “js” folder in this newly created “Week-9” folder
- Open the “js-challenges.html” file with a text editor of your choice
- Open the page in the browser and inspect it
- After inspecting the page, go to the tab labeled “Console” so you can check your output
- Review the 7 challenges listed above and add comments in your JS file to distinguish between each code block as depicted below
- Add code underneath each comment as I’ve described above. I would review the class example as it contains useful code for this assignment.

// Challenge 1

// Challenge 2

Your web page is **due October 28th 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 “JS Challenges” **You must submit a live link to your web page**. The work submitted will be evaluated based on the rubric explained below.

JS Challenges – 10pts

- Live link submitted – 1.5pt
- JS is externally placed and organized/easy to read – 1.5pts
- Each challenge is complete – 7pts total or 1pt for each challenge

Participation Proof 10/21

There are no participation questions for this week. Instead, you will work on one of the following items:

- Any class assignment (today’s assignment preferably)
- Final Project

To receive credit for participation today, you must show me what you worked on during class. I will go around the class after explaining the assignment for today and ask what work you plan on doing. Later in the class, I will come back around to check your progress. Once you have shown me enough work (will depend on what you’re working on), I will write your name down, and then you are free to go.

Previous Participation Questions/Responses

There were no participation questions for last week.

This link has the responses to the fun question from last week. 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 continue learning about JavaScript and how to use it on our pages. Specifically, we will go over arrays, the DOM, and more objects. The links below cover these items. Make sure to read them to gain a better understanding of how JS can make a page more interactive.

1. https://www.htmldog.com/guides/javascript/ (Objects, Arrays, DOM)
2. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events
3. https://javascript.info/function-basics