Week 11: JS Part III

Agenda

- Overview
  - Previous Participation Questions/Responses
  - Midterm Reminder
  - What's Due?
- Working with the DOM (Lecture/Class Activity)
  - "this" Keyword
  - Targeting with JS
  - Events in JS
  - Class Example & Online Resources
- JS Challenges 3 (CW/HW)
- Participation Questions (4)
- Next Week

Overview

Last week, we learned more about JS and how we could use it to retrieve input from the page and use it in functions. This week, we will learn about different ways to use elements on our pages, so we can use them to build more interactive components.

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 tonight at midnight**. Please make sure you are close to submitting it. 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 Questions (4)
- JS Challenges 3 (CW/HW)
- Midterm (due tonight at midnight)

Working with the DOM (Lecture/Class Activity)

"this" Keyword

You can use the "this" keyword with the DOM to target an object that just had an event occur on it. Look at the following code snippet as an example. When the button is clicked, the color for the text and background will change.

HTML

```
<button type="submit" id="example">Example Button</button>
```

JavaScript

```
document.getElementById("example").onclick = function() {
        this.style.color = "white";
        this.style.backgroundColor = "red";
}
```

In the example above, "this" refers to the button that was clicked. When you use "this" inside of a function that is triggered by an event, it refers to the object the event is happening to (button, text, etc.). When you use this outside of a function, it typically refers to the Window object.

Targeting with JS

We've already seen how JavaScript can be used to execute functions and create basic output on our page. We have done this using the Document Object Model which consists of the objects located in the document. The main method we have used so far has been document.getElementById().

Using the document object's "getElementById" method, we can target any **one** HTML element based on its id value. Each id value on our page must be unique and can't be repeated in other elements. This limits us to targeting only one element at a time with this method.

Another useful method is getElementsByClassName. Instead of using the id value of an HTML element, you would provide the class name of an element or multiple elements. In the example below, I select all elements with a class name of "para" and store them in a variable called paras.

```
var paras = document.getElementsByClassName("para");
```

When using this method, we create a collection of all the elements which match the class name provided. A collection is very similar to an array in that it holds multiple values. Let's say we had the following code:

First sentence.
Second sentence.
Third sentence.

If we use var paras = document.getElementsByClassName("para");, we would have a collection of all three paragraphs above. To refer to the first sentence, we could use paras[0] since the first item in a collection has an index value of 0 (just like arrays). The last paragraph could be referenced with paras[2] since the index values are 0, 1, and 2.

We could loop through each member of the collection with the code below:

```
for(var x = 0; x < paras.length; x++) {
        paras[x].style.color = "blue";
}
```

In the second part of the for loop, the condition we check before executing the code inside the loop (x < paras.length), we specify that the loop should only run up to but not including the length of the collection. Since the length of the collection is 3, x < paras.length means x < 2. This makes sense because if we refer to each item in our collection individually, it will look like the code below:

paras[0] – first sentence
paras[1] – second sentence
paras[2] – third sentence

During each execution of the loop, we change the elements to have a color of blue by using paras[x]. x just refers to the number which is 0 through 2 in our case.

Another method which can be used to select elements on our page is the querySelector() method. It is like the getElementById method in that it targets only **one** element on the page. However, instead of targeting something by its id value, we provide a CSS selector between the quotation marks. Any valid CSS selectors can be used.

In this example, I target anything with a class of "redPara" and give it a color of red. The querySelector method only targets the very first match and so this red color style wouldn't apply to the second paragraph tag.

This paragraph will be red. - Receives red color
This paragraph will be red. - Doesn't get styled

```
document.querySelector(".redPara").style.color = "red";
```

If we only wanted to target the span, we could also do that using this method. We would just change the value provided to ".redPara span" so that it references all span tags inside of anything with a class of "redPara."

The last function we will cover is querySelectorAll(). Just like querySelector, it uses a provided CSS selector to target elements. The main difference is that a list of items is returned just like with getElementsByClassName(). We will reuse the code from the querySelector example as demonstrated below.

This paragraph will be red.
This paragraph will be red.

var redParas = document.querySelectorAll(".redPara span");

Once we create a variable to represent the list of all items which match our CSS selector (span tags inside of elements with the redPara class), we can loop through it to change the style for each individual element.

```
for(var x = 0; x < redParas.length; x++) {
        redParas[x].style.color = "red";
}
```

After executing this code, all span tags within those paragraphs would have that red color applied. We could also refer to just one of those elements inside of the list by referring to the index value (same as arrays or collections).

redParas[0].style.color = "red"; - Only the very first paragraph in our list gets styled with this

Events in JavaScript
Most features built into web pages that use JavaScript rely on some sort of event. For example, if there is a drop-down menu which only appears after some element is clicked, we could say that there is a click event being used.

Another example of an event can be found on Twitter or some other platforms which limit characters with posts. As you type into an input box for a status update, there might be a counter present which tells you how many characters you have typed or have left. In this instance, there is a keypress event being used.

Some of the most popular JavaScript events are listed below:

- Click – A user clicks on an element
- Dblclick – A user double-clicks on an element
- Mouseover – When the pointer is moved onto an element or one of its children
- Keypress – When a user presses a key
    - Keydown – When the user is pressing a key
    - Keyup – When a user releases a key
- Onload – When an HTML object has loaded
- Focus – When an element gets focus

To add an event to an element, you can use the "on event" syntax as demonstrated below. In this example, we have a button with a class of "btn." We target that button with querySelector and then add a "onclick" event to it.

<button class="btn" type="submit">Click me</button>

document.querySelector(".btn").onclick = function () {//Some code in here }

We could change "onclick" to "onmouseover", "ondblclick" and other events as we want. After we specify the event to trigger our function, we define the function which is execute once the event is triggered. We would do the same thing for all other events.

Class Example & Online Resources
This example contains content which demonstrates how to use the various document methods mentioned above (getElementById, getElementsByClassNames, etc.). It also shows how some basic events work.

https://www.albany.edu/~cv762525/cinf201/examples/dom-manipulation.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://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
- https://www.w3schools.com/js/js_htmldom.asp
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events
- https://www.tutorialspoint.com/javascript/javascript_events.htm
- https://www.w3schools.com/jsref/met_element_addeventlistener.asp

JS Challenges 3 (CW/HW)

Download the "JS-Challenges-3.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 and some image files which depict how the page should look if the challenges are done correctly.

Your task is to read the challenges displayed on the HTML page and add JS as needed to complete them. Underneath each challenge description is a set of <script></script> tags where you will place your JavaScript.

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-11-Lecture-Notes.zip" folder from Blackboard
- Take the "js-challenges-3.html" file and the image files and place them in a folder you can find.
    - Create a "Week-11" folder in your "Documents" folder or some other folder you can access easily
    - Place both files in this newly created "Week-11" folder
- Open the "js-challenges-3.html" file with a text editor of your choice
- Open the "js-challenges-3.html" file with a web browser and read the instructions
- After reading the instructions, look at the provided images (if you didn't see my explanation in class) to understand what output I expect
- Review the 3 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 the onclick part of your function. All challenges start with the click of a button, so attach an onclick event to each button.
- 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 11th 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 3" **You must submit a live link to your web page**. The work submitted will be evaluated based on the rubric explained below.

JS Challenges 3 – 10pts

- Live link submitted – 1pt
- Your code is neatly organized – 1pt
- Each challenge is complete – 8pts total or 2.66pts for each challenge

Participation Questions 11/4

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/q58pbSjL2hqPjCc67

Previous Participation Questions/Responses
There were no participation questions for last week.

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 learn about JQuery, which is a library that is based on JavaScipt. It's easy to add to our pages and makes coding a little bit easier. Feel free to look through the two links below to get a glimpse of what JQuery is and how it works.

- https://jquery.com/
- https://www.w3schools.com/jquery/jquery_intro.asp