WESTCLIFF
UNIVERSITY
Educate. Inspire. Empower.

Module 1

Week 4 Day 3

JavaScript Essentials Review

Agenda

Review: JavaScript Essentials

- More Array Methods
- How to generate random numbers using the random method
- Document Object Model (DOM)
- JavaScript DOM events and types of
- JavaScript DOM event handling
- Project 1 Information
- Lab Assignment
- Homework

More Array Methods

More Array Methods

- indexOf() - to determine whether or not an object is in an array
- lastIndexOf() - returns the last index at which an item is found
- forEach() - a cleaner version of the for loop to loop through an array
- includes() - determines if the array contains the specified item and returns true or false as output.
- every() - checks every array item against a condition & returns a falsy value
- some() - like every but the passing condition is at least one callback returns true
- map() - loops through an array, runs a function and create a new array built from the return values of each iteration
- filter() - like map but creates a new array containing only items of the original array that return a truthy value from the callback
- reduce() - melt the items in an array down to a single value by the operations performed in its callback function

More Array Methods

- indexOf() Method
  - It returns the *first* index at which the item was found, or -1 if it was not found at all. Strict equality is used to determine that an item is present in the array.
  - Example:

```
<h2>JavaScript Arrays</h2>
<p>Example of more Array Methods</p>

<script>
//Array
var fruits = ["apples", "oranges", "pears", "apples"];

//var search = fruits.indexOf("apples");
var search = fruits.indexOf("bananas");
document.write("The search returns: " + search);
</script>
```

View on Browser:

JavaScript Arrays

Example of more Array Methods

The search returns: -1

More Array Methods

- lastIndexOf() Method
  - It returns the *last* index at which the item was found, even if an identical item was found first.
  - Example:

```
<h2>JavaScript Arrays</h2>
<p>Example of more Array Methods</p>

<script>
//Array
var fruits = ["apples", "oranges", "pears", "apples"];

var search = fruits.lastIndexOf("apples");
document.write("The search returns: " + search);
</script>
```

View on Browser:

JavaScript Arrays

Example of more Array Methods

The search returns: 3

More Array Methods

- forEach() Method
  - List each item in an array. Must call a function for each array item. Function must pass one value into it: array item. Optional second value: index of item.
  - Example:

```
<h2>JavaScript Arrays</h2>
<p>Example of more Array Methods</p>

<script>
//Array
var fruits = ["apples", "oranges", "pears", "cherries"];

fruits.forEach(myFunction);

function myFunction(fruit, indexNum) {
    document.write(indexNum + " - " + fruit + "<br>");
}
</script>
```

View on Browser:

JavaScript Arrays

Example of more Array Methods

0 - apples
1 - oranges
2 - pears
3 - cherries

More Array Methods

- includes() Method
  - This method determines whether the array contains the specified item. It returns **true** or **false** as output depending on the result.
  - Example:

```
<h2>JavaScript Arrays</h2>
<p>Example of more Array Methods</p>

<script>
//Array
var fruits = ["apples", "oranges", "pears", "cherries"];

var search = fruits.includes("mangoes");
document.write("The search returns: " + search);
</script>
```

View on Browser:

JavaScript Arrays

Example of more Array Methods

The search returns: false

More Array Methods

- every() Method
  - Checks if all elements in an array pass a test (provided as a function). If it finds an array element where the function returns a *false* value, every() returns *false* (and does not check the remaining values). If no false occur, every() returns *true*.
  - Example:

```
<h2>JavaScript Arrays</h2>
<p>Example of more Array Methods</p>

<script>
//Array
var fruits = ["apples", "oranges", "pears", "cherries"];

var search = fruits.every(myFunction);
function myFunction(fruit) {
    //return fruit.length > 6;
    return fruit.length > 3;
}
document.write("The search returns: " + search);
</script>
```

View on Browser:

JavaScript Arrays

Example of more Array Methods

The search returns: true

More Array Methods

- some() Method
  - Checks if all elements in an array pass a test (provided as a function). If it finds an array element where the function returns a *true* value, some() returns *true* (and does not check the remaining values). Otherwise, it returns *false*.
  - Example:

```
<h2>JavaScript Arrays</h2>
<p>Example of more Array Methods</p>

<script>
//Array
var fruits = ["apples", "oranges", "pears", "cherries"];

var search = fruits.some(myFunction);
function myFunction(fruit) {
    //return fruit.length > 6;
    return fruit.length < 3;
}
document.write("The search returns: " + search);
</script>
```

View on Browser:

JavaScript Arrays

Example of more Array Methods

The search returns: false

More Array Methods

- map() Method
  - Maps each element of an existing array by calling a function for each element and assign its results in a new array. All elements in the parent array remains as it does not mutate the original array.
  - Example:

```
<h2>JavaScript Arrays</h2>
<p>Example of more Array Methods</p>

<script>
//Array
var fruits = ["apples", "oranges", "pears", "cherries"];

var candies = fruits.map(myFunction);
function myFunction(fruit) {
    return " candy " + fruit;
};
document.write(candies);
</script>
```

View on Browser:

JavaScript Arrays

Example of more Array Methods

candy apples, candy oranges, candy pears, candy cherries

More Array Methods

- filter() Method
  - Creates a new array populated with elements that meets the filter criteria (provided as a function) of the parent array.
  - Example:

```
<h2>JavaScript Arrays</h2>
<p>Example of more Array Methods</p>

<script>
//Array
var fruits = ["apples", "oranges", "pears", "cherries"];

var myFruits = fruits.filter(myFunction);
function myFunction(fruit) {
    //return fruit.length > 6;
    return fruit.includes("es");
};
document.write(myFruits);
</script>
```

View on Browser:

JavaScript Arrays

Example of more Array Methods

apples,oranges,cherries

More Array Methods

- reduce() Method
  - reduces the array to a single final value. It takes two arguments: *reducer* & *accumulator*. The *accumulator* accumulates a value based on the action (*reducer*) performs.
  - Example:

```
<h2>JavaScript Arrays</h2>
<p>Example of more Array Methods</p>

<script>
//Array
var daysales = [305, 432, 376, 290];

var weeklySales = daysales.reduce(myFunction);
function myFunction(accumTotal,curSales) {
    return accumTotal + curSales;
};
document.write(weeklySales);
</script>
```

View on Browser:

JavaScript Arrays

Example of more Array Methods

1403

Generate Random Numbers

Generate random numbers

- In JavaScript, it is possible to generate random numbers within a specified range of numbers so that number generated can be used within the program.
- This can be easily done using the Math.random() function.
- How to use:

1. Without a specified range of numbers:
   - Math.random() => This will generate a number from 0 (inclusive) to 1 (exclusive - up to 0.999999999999999)
2. With a specified range of numbers:
   - Math.random()*max => This will generate a number from 0 to max
   - Math.random()*max+min => This will generate a number from min to max+min
   - Math.random()*max-min => This will generate a number from min to max-min

Will generate up to ?.999999999999999
Example: Max=10 => 9. 999999999999999

Generate random numbers

- Example 1 (without specified range):

```
<h2>JavaScript Random Numbers</h2>
<p>Using the Math.random() function</p>

<script>
var myNumber = Math.random();
document.write("The generated number is: " + myNumber);
</script>
```

View on Browser:

JavaScript Random Numbers

Using the Math.random() function

The generated number is: 0.9797803638811104

Generate random numbers

- Example 2 (with specified range from 0 to 10):

```
<h2>JavaScript Random Numbers</h2>
<p>Using the Math.random() function</p>

<script>
var myNumber = Math.random()*10;
document.write("The generated number is: " + myNumber);
</script>
```

Note:
By default numbers are in decimal places but can be rounded down to its nearest integer using *Math.floor()*.

View on Browser:

JavaScript Random Numbers

Using the Math.random() function

The generated number is: 9.079133525444734

Generate random numbers

- Example 3 (with specified range: 1 to 11, round to nearest integer):

```
<h2>JavaScript Random Numbers</h2>
<p>Using the Math.random() function</p>

<script>
var myNumber = Math.floor(Math.random() * 10+1);
document.write("The generated number is: " + myNumber);
</script>
```

Note:
In this case, the range tested is actually between 1 and 11 ie. 10+1=11. But with Math.floor, it will round down to 10 if generated number is 10.999999999999999

View on Browser:

JavaScript Random Numbers

Using the Math.random() function

The generated number is: 8

Generate random numbers

- Example 4 (with specified range: -1 to 9, round to nearest integer):

```
<h2>JavaScript Random Numbers</h2>
<p>Using the Math.random() function</p>

<script>
var myNumber = Math.floor(Math.random() * 10-1);
document.write("The generated number is: " + myNumber);
</script>
```

Note:
In this case, the range tested is actually between -1 and 9 ie. 10-1=9. But with Math.floor, it will round down to 8 if generated number is 8.999999999999999

View on Browser:

JavaScript Random Numbers

Using the Math.random() function

The generated number is: -1

Generate random alphabets

- Example:

```
<h2>JavaScript Random Numbers</h2>
<p>Using the Math.random() function</p>

<script>
var alphabet = "abcdefghijklmnopqrstuvwxyz";
var randomAlphabet = alphabet[Math.floor(Math.random() * alphabet.length)]
// example if var randomAlphabet = alphabet[12]  => this output the letter m
document.write("The generated letter is: " + randomAlphabet);
</script>
```

View on Browser:

JavaScript Random Numbers

Using the Math.random() function

The generated letter is: c

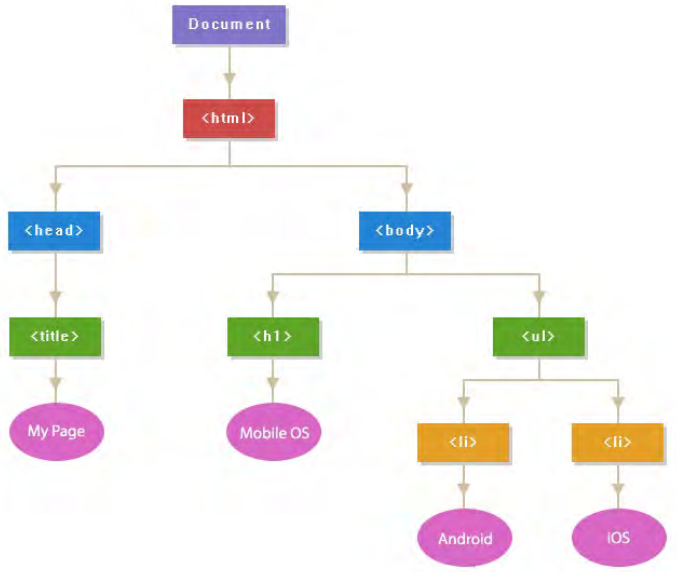
Document Object Model

Document Object Model (DOM)

- When a web page is loaded, the browser creates a Document Object Model of the page. The HTML DOM model is constructed as a tree of objects:

Example

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>My Page</title>
5  </head>
6  <body>
7      <h1>Mobile OS</h1>
8      <ul>
9          <li>Android</li>
10         <li>iOS</li>
11     </ul>
12 </body>
13 </html>
```



Document Object Model (DOM)

- With the object model, JavaScript can exercise its power to create a dynamic HTML. It can:
  - change all the HTML elements in the page
  - change all the HTML attributes in the page
  - change all the CSS styles in the page
  - remove existing HTML elements and attributes
  - add new HTML elements and attributes
  - react to all existing HTML events in the page
  - can create new HTML events in the page

Document Object Model (DOM)

- DOM is a W3C (World Wide Web Consortium) standard. It defines a standard for accessing documents' structure, contents and styles.
- What is the HTML DOM?
  - HTML DOM is a standard object model and programming interface for HTML. It defines:
    - The HTML elements as *objects*
    - The *properties* of all HTML elements
    - The *methods* to access all HTML elements
    - The *events* for all HTML elements
  - In other words: HTML DOM is a standard for how to *get*, *change*, *add*, or *delete* HTML elements.

Document Object Model (DOM)

- As mentioned earlier, HTML DOM can be accessed with JavaScript. In the DOM, all HTML elements are defined as *objects*.
- The programming interface is the *properties* and *methods* of each object.
  - A *property* is a value that you can get or set (like changing the content of an HTML element).
  - A *method* is an action you can do (like add or deleting an HTML element).

Examples of *Properties*:
- innerHTML
- attributes
- style
- childNodes

Examples of *Methods*:
- getElementById()
- getElementsByTagName()
- appendChild()
- removeChild()

Document Object Model (DOM)

- How to use Properties: (x = HTML Element)
  - x.innerHTML = ? - change the inner text value of x
  - x.attributes - change the attribute value of x
  - x.style - change the style of x
  - x.childNodes - change the child nodes of x
- Example:

```
<h2>HTML DOM Properties</h2>
<p>This demo illustrates how to insert content into an empty
element on the page using the innerHTML property.</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

View on Browser:

HTML DOM Properties

This demo illustrates how to insert content into an empty element on the page using the innerHTML property.

Hello World!

Document Object Model (DOM)

- How to use Methods:
  - x.getElementById(*id*) - get the element with a specified id
  - x.getElementsByTagName(*name*) - get all elements with a specified tag name
  - x.appendChild(*node*) - insert a child node to x
  - x.removeChild(*node*) - remove a child node from x
- Example:

```
<h2>HTML DOM Properties</h2>
<p>This demo illustrates how to select an element on the
page using the getElementById() method.</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

View on Browser:

HTML DOM Properties

This demo illustrates how to select an element on the page using the getElementById() method.

Hello World!

Events and DOM Events

Events

- What are events?
  - Events are things that will occur. Going to a birthday party is an event. Watching a YouTube video is an event. Events are therefore actions.
  - With events, there are also expected reactions. Going to a birthday party reaction => bringing a birthday gift. Watching a YouTube video reaction => laughing hard.
  - In JavaScript, those reactions are also refer to as event handling - in other words how one react to that action.
  - On an application like a website, common user interactions with the site are considered as events. In fact more appropriately JavaScript events. After all, JavaScript makes those interactions possible – recall that JavaScript is a behavioral language.

HTML Events

- Examples:
  - When a user *clicks* the mouse
  - When a web page has *loaded*
  - When an image has been *loaded*
  - When the *mouse moves over* an element
  - When an input field is *changed*
  - When an HTML form is *submitted*
  - When a user *strokes a key*

HTML DOM Events

- Types of events:
  - onmouseover
  - onmouseout
  - onmouseup
  - onmousedown
  - onclick
  - onload
  - onfocus
  - onchange
  - onsubmit

Handling Events

Reacting to Events

- When an event occurs, there's a reaction - JavaScript therefore can be executed when an event occurs, like when a user clicks on an HTML element.
- To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute, like this: onclick=*JavaScript*
- A simple example: user click on existing text and is replaced with a new text.

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="this.innerHTML='Hello!'">Click This Text!</h1>

</body>
</html>
```

View on Browser (before click):

Click This Text!

View on Browser (after click):

Hello!

Reacting to Events & Handling Events

- The previous example illustrates how changes affect itself ie. the same element when clicked.
- But it can also affect other elements. Events are *handled* if changes are to affect another element. They are called *event handlers*.
- Event handlers are basically handled by JavaScript functions.
- The basic syntax:

```
<tagName onclick="functionName()">Some Text</tagName>

<script>
function functionName() {
  some statement
}
</script>
```

In this case, the event *onclick* calls the handler function *functionName*.

Reacting to Events & Handling Events

- Examples of Event Handling:

```
<h2 onclick="changeText()">Click This Text!</h2>
<p id="demo"></p>

<script>
function changeText() {
  document.getElementById("demo").innerHTML = "Hello!";
}
</script>
```

View on Browser:

Click This Text!

Hello!

```
<p>Click on the button to display the current date.</p>
<button onclick="displayDate()">Get Date &amp;amp; Time</button>

<p id="demo"></p>

<script>
function displayDate() {
  document.getElementById("demo").innerHTML = Date();
}
</script>
```

View on Browser:

Click on the button to display the current date.

Get Date & Time

Wed Nov 25 2020 15:55:23 GMT-0600 (CST)

Reacting to Events & Handling Events

- Examples of Event Handling:
  - You can assign event to the DOM ie. without putting the event directly on the DOM element. Let's use the get date example:

```
<p>Click on the button to display the current date.</p>
<button id="myBtn">Get Date &amp;amp; Time</button>

<p id="demo"></p>

<script>
document.getElementById("myBtn").onclick = displayDate;

function displayDate() {
  document.getElementById("demo").innerHTML = Date();
}
</script>
```

View on Browser:

Click on the button to display the current date.

Get Date & Time

Wed Nov 25 2020 15:55:23 GMT-0600 (CST)

Reacting to Events & Handling Events

- In this next example, we will look at non user events – in this case a browser triggered event. Events such as *onload* is mainly used for browser trigger – in other words, an event occur when the webpage loads on the browser. The handling will be similar to those we saw in previous examples.

```
<body onload="insertText()">

<p>This example shows texts inserted in an empty
element when the page loads on the browser.</p>

<p id="demo"></p>

<script>
function insertText() {
  document.getElementById("demo").innerHTML = "Hello!";
}
</script>
```

View on Browser:

This example shows texts inserted in an empty element when the page loads on the browser.

Hello!

Reacting to Events & Handling Events

- Back to user events – in this case a form element event. Events such as *onchange* is mainly used for form input trigger – for example, an event occur when the user tab away after entering text in a text field. The handling will be similar to those we saw in previous examples.

```
<p>This example shows an event triggered when user tab
away after entering text in a text field.</p>

Enter your name: <input type="text" id="fname"
onchange="insertText()">

<p id="demo"></p>

<script>
function insertText() {
  document.getElementById("demo").innerHTML = "Hello!";
}
</script>
```

View on Browser:

This example shows an event triggered when user tab away after entering text in a text field.

Enter your name: Rich

Hello!

Reacting to Events & Handling Events

- These next properties are not about events and events handling but are commonly used in conjunction with event handling.
- We have seen in previous examples how information can be inserted or changed in a HTML element called by an event.
- What if we want to obtain or capture information from form elements, such as an input text field entered by users? These are common practices in a real world application.
- The following are two form text properties we can use to do this:
  - value - capture/obtain information entered by user
  - length - find out number of characters entered in the input element

Reacting to Events & Handling Events

- Example of the value property by referencing the id of the text box:

```
<p>This example illustrates how to capture text entered in a
text box and then display on the page.</p>

<form id="myForm" action="">
  First name: <input type="text" id="fname" name="fname"><br>
  Last name: <input type="text" id="lname" name="lname"><br>
</form>

<button onclick="myFunction()">Click to Display</button>

<p id="demo"></p>

<script>
function myFunction() {
  var x = document.getElementById("fname").value;
  document.getElementById("demo").innerHTML = "Hello " + x;
}
</script>
```

View on Browser:

This example illustrates how to capture text entered in a text box and then display on the page.

First name: Jane

Last name: Doe

Click to Display

Hello Jane

Reacting to Events & Handling Events

- Example of using the length property to display information entered in each text box:

```
<p>This example illustrates capturing texts entered in text
boxes and then display them collectively on the page.</p>

<form id="myForm" action="">
  First name: <input type="text" id="fname" name="fname"><br>
  Last name: <input type="text" id="lname" name="lname"><br>
  Age: <input type="text" id="age" name="age"><br>
  Email: <input type="email" id="email" name="email"><br>
</form>
<button onclick="myFunction()">Click to Display</button>
<p id="demo"></p>

<script>
function myFunction() {
  var ref = document.getElementById("myForm");
  var txt = "";
  for (var i=0; i<ref.length; i++) {
    txt = txt + ref.elements[i].value + "<br>";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script>
```

View on Browser:

This example illustrates capturing texts entered in text boxes and then display them collectively on the page.

First name: Jane
Last name: Doe
Age: 33
Email: jd@email.com
Click to Display

Jane
Doe
33
jd@email.com

Reacting to Events & Handling Events

- You can also use HTML DOM method to change the style of contents in a html element. To do this, use the style object on the method. Here's the syntax:

  document.getElementById(*idname*).style.*property* = *new style*

- To use the DOM style object, you must specify which style property you want to style. Remember: font size, color or background color? The property names are a little different from its CSS counterpart. Here are some examples:

  backgroundColor, fontSize, listStyleType, marginLeft, textAlign

- For full list of them, visit:

  https://www.w3schools.com/jsref/dom_obj_style.asp

Reacting to Events & Handling Events

- This example uses an both non event and an event with a handler.

```
<h1 id="heading1">Hello World!</h1>

<p>Demo on changing the style of a HTML element.</p>

<button onClick="changeStyle()">Change</button>

<script>
document.getElementById("heading1").style.color = "#ccef48";
document.getElementById("heading1").style.fontFamily = "Impact";
document.getElementById("heading1").style.fontSize = "2em";

function changeStyle() {
    document.getElementById("heading1").style.letterSpacing = "0.2em";
}
</script>
```

View on Browser:

Resources

https://www.w3schools.com/js/js_arrays.asp
https://www.dyn-web.com/javascript/arrays/add.php
https://www.w3schools.com/js/js_random.asp

Questions?