Week 4 Workshop

Web Development Fundamentals
HTML, CSS, and JavaScript

Agenda

Activity	Time	~Start
Get Prepared: Log in to Nucamp Learning Portal • Slack • Screenshare	10 minutes	9:00am
Check-In	10 minutes	9:10am
Week 4 Recap	60 minutes	9:20am
Tasks 1 & 2	40 minutes	10:20am
BREAK	15 minutes	11:00am
Tasks 2-4	90 minutes	11:15am
Check-Out	15 minutes	12:45pm

Week 4 Recap - Overview

- For Loops
- For ... Of Loops
- Break & Continue
- More Array Methods
- More About Strings
  - String Methods
- The HTML DOM
- Locating Nodes
- Creating & Adding Nodes
- Adding Nodes
- Removing Nodes
- Cloning Nodes
- Inline OnEvent Handlers
- Mouse Events
- setTimeout() & setInterval()
- clearTimeout() & clearInterval()
- addEventListener()
- removeEventListener()

Check-In

- How was this week for you? Any particular challenges or accomplishments?
- Did you understand the Exercises and were you able to complete them?
- How were the Challenges and Quiz this week?
- We know that this was a difficult week for many. Please ask if you have questions.

For Loops

- For loops have built-in support to set a variable to iterate for the loop condition:

```
let x = 5;
for (let i = 1; i <= 3; i++) {
    x = x * i;
    console.log(x);
}
```

Output:
5
10
30

Discuss: What would the above loop log to the console? Do you understand why?

Type it into your console to try it out.

Note: the statement `x = x * i;` could also be written more simply as: `x *= i;`

For … Of Loops

- for … of loops can be used with arrays and other iterables in JavaScript.

```
const exampleArray = [2, 4, 6, 8];
for (const n of exampleArray) {
    console.log(n * 2);
}
```

Output:

4
8
12
16

Discuss: What would the above loop log to the console? Do you understand why?

Type it into your console to try it out.

Break & Continue

- Use in any for or while loop to break out of the loop entirely (break), or skip the current iteration (continue).

- Given the code to the right, what is the response if you enter a guess of –1 when prompted?

This will jump/continue to the top of the while loop and prompt the user again

- What about a guess of true?

```
let fingers = Math.floor(Math.random() * 10) + 1;
let guess = 0;
while (guess !== fingers) {
    guess = +prompt('How many fingers am I holding up?');
    if (isNaN(guess)) {
        break;
    }
    if ((guess < 0) || (guess > 10)) {
        continue;
    }
    if (guess === fingers) {
        alert('You got it!');
    } else {
        alert('Try again!');
    }
}
```

prompt() returns a String so when true is passed this will result in the string "true". The "+" operator then tries to convert it to a Number which it then results in NaN since it doesn't contain a number wrapped in a string (e.g. "1"). It will then break out of the loop.

More Array Methods

- **concat()** - combine two arrays into one
  - Does NOT mutate the original array

```
1 const array1 = ['a', 'b', 'c'];
2 const array2 = ['d', 'e', 'f'];
3 const array3 = array1.concat(array2);
4
5 console.log(array3);
6 // expected output: Array ["a", "b", "c", "d", "e", "f"]
```

- **sort()** - sort an array of strings alphabetically
  - Mutates the original array
  - Note: When using numbers with **sort()**, results may not be what you expect.
  - [1, 13, 1000, 29, 255].sort() would result in this array:
    - [1, 1000, 13, 255, 29]
    - NOT [1, 13, 29, 255, 1000] as you might expect

- **reverse()** reverses the order of an array
  - Mutates the original array

More Array Methods (cont)

- slice() - copy a part of an array and return a new array – this does not mutate the original array
- splice() - inserts, replaces, or removes parts of an array – this mutates the original array!

If you run this code:

```
const birds = ['crow', 'hawk', 'owl', 'goose'];
let newBirds = birds.slice(1,3);
```

- The variable newBirds now holds the array: ['hawk', 'owl']
  The array in the variable birds remains the same.

slice – think COPY

splice – think CUT

If you then ran the code:

```
newBirds = birds.splice(1,3,);
```

- The variable newBirds now holds the array: [hawk, 'owl', 'goose']
  The array in the variable birds is now only: ['crow']

Slice and splice look very similar and do somewhat similar things, but they work very differently so be careful with them.

More About Strings

- Strings are *not* arrays but in some ways work similarly to arrays
- You can access each character in a string by its index, as if each character is an item in an array –
  - 'hello'[0] is 'h'
  - 'hello'[1] is 'e'
  - etc

- And you can get the length of a string with .length:
  - 'hello'.length is 5

Discuss: Given this variable:

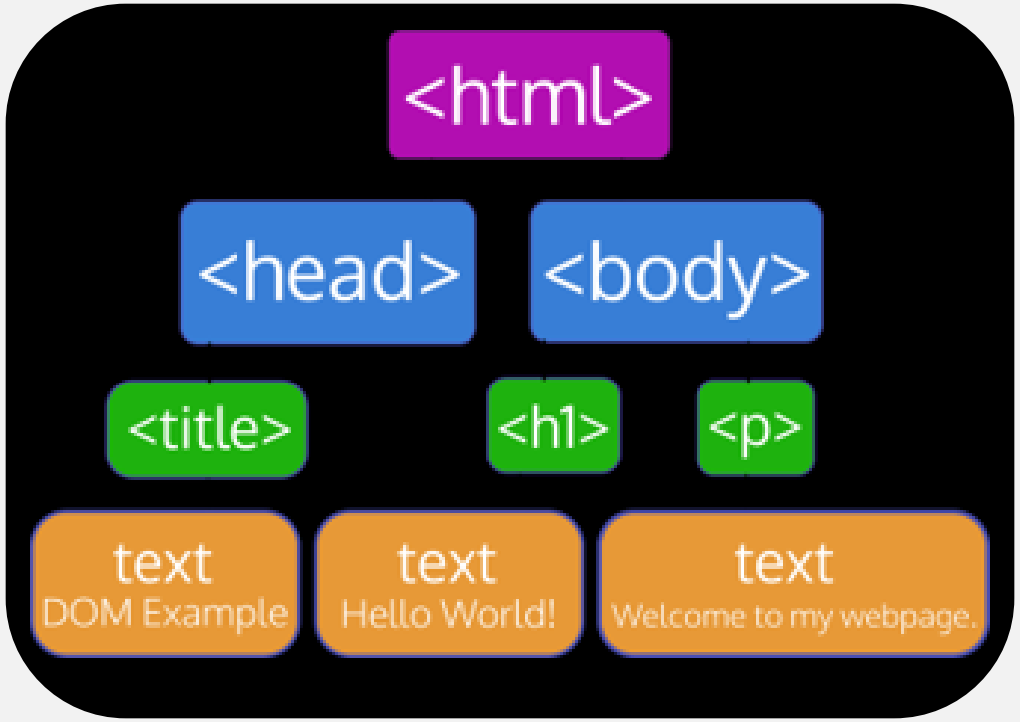
```
let myStr = 'dog';
```

Can you change **myStr**'s value from **'dog'** to **'zog'** using the code as follows?:

```
myStr[0] = 'z';
```

HTML DOM & Node Relationships

- Let's review the DOM:



- What node is the **parentNode** of <head>?

HTML

- What node is the **nextSibling** of <h1>?

P

- What node is the **previousSibling** of <p>?

H1

- What node is the **firstChild** of <body>?

H1

- What node is the **lastChild** of <html>?

BODY

More Node Relationships

Discuss: What is the difference between firstChild and firstElementChild, and lastChild and lastElementChild?

firstChild & lastChild

Can return ANY Node Type
(TEXT, ELEMENT, COMMENT, etc)

FirstElementChild & lastElementChild

Returns only an Element Node
(HTMLElement / tag)

Node type		Description	Children
1	Element	Represents an element	Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
2	Attr	Represents an attribute	Text, EntityReference
3	Text	Represents textual content in an element or attribute	None
4	CDATASection	Represents a CDATA section in a document (text that will NOT be parsed by a parser)	None
5	EntityReference	Represents an entity reference	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
6	Entity	Represents an entity	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
7	ProcessingInstruction	Represents a processing instruction	None
8	Comment	Represents a comment	None
9	Document	Represents the entire document (the root-node of the DOM tree)	Element, ProcessingInstruction, Comment, DocumentType
10	DocumentType	Provides an interface to the entities defined for the document	None
11	DocumentFragment	Represents a "lightweight" Document object, which can hold a portion of a document	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
12	Notation	Represents a notation declared in the DTD	None

Locating Nodes

Given this HTML:

```
<html>
  <body>
    <h1 id="myH1">Find Me</h1>
  </body>
</html>
```

Discuss: Which of the following ways will correctly find this node?

a) document.getElementsByTagName('h1');
b) document.getElementById('myH1')
c) document.querySelector('myH1')

"B" is the correct answer

"A" would not work as it will return an array. The below would have worked
document.getElementsByTagName('h1')[0];

"C" would not work because the querySelector expects the argument to be in the form of a CSS Selector
The below would have worked
document.querySelector('#myH1')

Creating and Adding Nodes

- To create a new element node in JavaScript, use **document.createElement('elementname')**
  - e.g. document.createElement('div');

- To create a new text node, use **document.createTextNode('your text')**
  - e.g. document.createTextNode('Hello World');

- Once you have created a new node, it isn't part of the DOM yet, you must add it to the DOM using a node method such as *node*.appendChild() or *node*.insertBefore()

Removing Nodes

- Two ways to remove nodes:
  - *node*.removeChild(*child node to remove*)
  - *node*.remove()

- Example – let's say you have this HTML:

```
<p id="myP"><img src="cat.jpg" /></p>
```

- And this JavaScript:

```
const myPara = document.querySelector('#myP');
```

- If you did this:

```
myPara.remove();
```

// This would remove the entire <p> element and its child <img>.

- If you did this:

```
myPara.removeChild(myPara.firstChild);
```

// This would remove just the <img> element.

Removing Nodes

- Use a while loop to remove all the child nodes of a parent node without removing the parent node:

```
while (node.firstChild) {
  node.removeChild(node.firstChild);
}
```

Discuss: Are there any volunteers who would like to try to walk through and explain the above code in their own words?

- Repeat the loop as long as **node.firstChild** comes back with a **truthy** value (meaning there is at least one child node).
- Inside the loop, remove the first child node only, then start the loop again.
- Once **node.firstChild** comes back with a **falsy** value (e.g. **null**) that means there are no more child nodes left, so exit the loop

Cloning Nodes

- **Discuss**: What is the difference between using *node*.cloneNode() and *node*.cloneNode(true)?

node.clone()	Clones only the node itself
node.clone(true)	Clones the node as well as the node branch (descendant nodes)

Inline OnEvent Handlers

- There are various events in JavaScript such as **click**, **mouseover**, etc
- You can set functions to respond to these events, these are called event handler functions, or just event handlers
- To set an inline event handler in HTML, you use the name of the event such as 'click', add 'on' to it – so 'onclick', 'onmouseover', etc
- Then you use it as an attribute in an HTML element:

```
<div onclick="runThisFunction();">Click Me</div>
```

- Give the function to run as the attribute value, use quotes around it and include the argument list, e.g. "runThisFunction()"

Mouse Events

- You learned these inline on-event handlers for mouse events:
  - onclick
  - onmousedown
  - onmouseup
  - onmouseover
  - onmouseout

Discuss: What's the difference between **onmousedown** and **onclick**?

`onmousedown` will trigger as soon as either the left or right (or middle) is pressed.

`onclick` will only trigger when the **left mouse** button is **pressed** and **released** on the same object.

setTimeout() & setInterval()

- Set timer events in JavaScript to delay the execution of a function for a given length of time
- setTimeout() calls a function once after the time has run out
- setInterval() calls a function repeatedly, resetting the clock once the time runs out
- You call each the same way: the first argument is a function name, the second is a time in milliseconds:
  - setTimeout(doSomething, 10000); // will call doSomething() in 10 seconds
  - setInterval(doSomething, 10000); // will call doSomething(() every 10 seconds

clearTimeout() & clearInterval

- setTimeout() and setInterval() both return a unique timer ID as their return value when you call them
- Save these return values to a variable so that you can use them to clear the timer if necessary. Otherwise you have no way to stop the timer if you need to. E.g.:
- `let timerID = setInterval(doSomething, 10000);`
- `clearInterval(timerID);` // stop the interval timer

addEventListener

- addEventListener() is the recommended way in vanilla JS to add event handler functions to events

```
node.addEventListener('eventname', functionname);
```

Example: You want to have a button respond to clicks by calling a function named doSomething. You have the button node in a variable called btnNode already. You can do this:

```
btnNode.addEventListener('click', doSomething);
```

Remember: addEventListener can be used with the same event, on the same node, more than once.

onevent Handlers like onclick can only be used once per element

Note: No 'on' prefix! It's 'click' and **not** 'onclick'

- The event name goes inside quotes, single or double
- The function name does **NOT** go inside quotes, and you don't include the argument list – it's just doSomething, **not** 'doSomething' or doSomething()

removeEventListener()

- Use just like addEventListener() to remove an event handler function from an event
- So if you used this to add an event handler function to an event:
- btnNode.addEventListener('click', doSomething);
- You can use this to remove it:
- btnNode.removeEventListener('click', doSomething);

Questions during this week tasks?

- If we have extra time before the Workshop then feel free to bring up any unresolved questions, and to discuss any Challenge Questions or Code Challenges.

- Otherwise, please start the Workshop Assignment and save the discussion for after the assignment is finished, or online.

Workshop Assignment

- It's time to start the workshop assignment!

- Break out into groups of 2-3.
  - Sit near your workshop partner(s) in in person
  - For online Workshops your instructor may break you out into different virtual rooms

- Work closely with each other.
  - Don't forget that the 20-minute rule becomes the 10-minute rule during workshops!
  - 10-minute rule does *not* apply to talking to your partner(s). Work together throughout. This will be useful practice for working with teams in real life.

- Follow the workshop instructions very closely.
  - Talk to your instructor if any of the instructions are unclear to you.

Assignment Submission & Check-Out

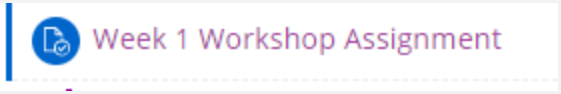
- Submit the **matching-game.html** page at the bottom of the assignment page in the learning portal.

- Example instruction on the next slide

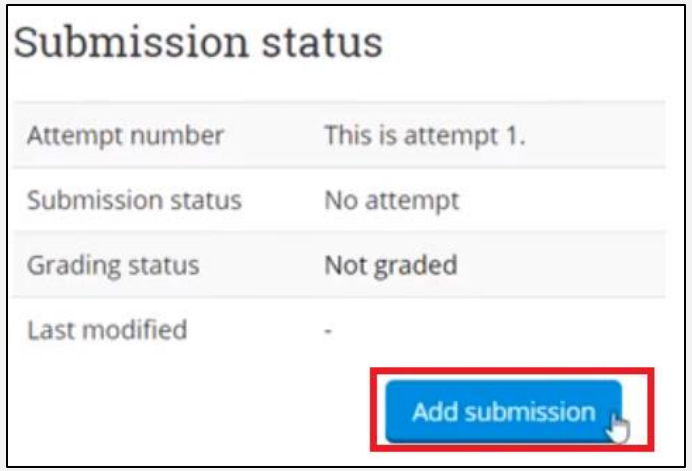
Congratulations!!!

Submitting Your Assignment

- Go to https://learn.nucamp.co
  - Click "**Workshop Assignment: Students' Work**"



  - Upload your work by clicking "**Add Submission**", select the file, and then click "save"



- Note that your work is in Draft status
  - Click "**Submit assignment**" to submit it