Week 3 Workshop

Web Development Fundamentals

HTML, CSS, and JavaScript

Agenda

Activity	Time
Get Prepared: Log in to Nucamp Learning Portal • Slack • Screenshare	10 minutes
Check-In	10 minutes
Week 3 Recap	60 minutes
Task 1	40 minutes
BREAK	15 minutes
Task 1 & 2	90 minutes
Check-Out	15 minutes

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.

Week 3 Recap - Overview

New Concepts This Week

- What is JavaScript?
- HTML: The script element
- HTML: The onclick attribute
- Variables
- Data types
- Functions
- Function Parameters & Arguments
- The JavaScript console
- If … Else If … Else
- Comparison OperatorsLogical Operators
- Truthy & Falsy
- Switch Statements
- += -= ++ --
- While Loops
- Arrays
- Scope
- Math.random() & Math.floor()

Next slides will review these concepts

What is JavaScript?

- Programming language originally created to run inside browsers
- Now most popular programming language for web development, used in browsers, desktops, servers, mobile apps
- ECMAScript: Official specification for JavaScript – last major version was in 2015, called ECMAScript 2015 or ES6
  - Current version is ES9, but changes since ES6 have been minor
- Many technologies use JavaScript (jQuery, Node.js, React, etc); you must first learn "vanilla JavaScript" to use them

HTML: The script element

- Use <script> element to add JavaScript to HTML page
- Add JavaScript between <script></script> tags
- ...or link to external JS file: <script src="index.js"></script>
- You can link to multiple JS files
- Generally, JavaScript inside <script> tags or in external JS file that is ***not*** inside a function will run automatically when the page is loaded

HTML: The onclick attribute

- Add onclick attribute to HTML element such as button to run JS function when element is clicked:
  - `<button type="button" onclick="runFunction()">Click Me</button>`

- There are multiple ways to trigger JavaScript from an HTML page, this is one way

Variables

- A named container for some value
- Create: Declare a variable using let or const
  - let for variables that will have their values reassigned
  - const for variables whose values will be assigned only once
- Pre-ES6 variable declaration keyword var is commonly seen in older codebases
- Values stored in variables can be of several different data types

Variables (cont)

- Use the assignment operator = to set a variable's value
- You learned about: number, string, Boolean
  - Number: const myNum = 1;
  - String: const myString = 'foo';
    - '5' is a string, not a number
    - Can combine strings into a single string with +
      - const myString = 'foo' + 'bar' is same as const myString = 'foobar';
  - Boolean: const myBoolean = true;
    - Can be true or false
- Two more data types: null and undefined
  - null is an intended non-value set by the programmer
  - undefined is the value of a variable that has been declared but not initialized
- There are other data types that will not be covered for this introductory class

Variables (cont)

- The first time you assign a value to a variable is called initialization
- You must initialize const variables at declaration:
  - const x = 1;
  - ~~const x;~~
- You *can* declare let variables then initialize later, but best practice: initialize at declaration to prevent issues with undefined variables
  - let x; <-- OK, technically permitted
  - let x = 0; <-- Better
- Only use let keyword when you first create the variable
- To assign/reassign its value later after declaration, use only the variable name
  - x = 2;

Functions

- A segment of code that can be grouped together, given a name, and called by that name from other places in the code
- Multiple ways to define a function, simplest way is with function declaration syntax:

```
function sayHello(name) {
    console.log('Hello ' + name);
}
```

- Defining/declaring a function does not run it. It must be called.
- Call a function (run the code inside it) with the function's name followed by an argument list:

```
sayHello('John');
```

Functions – Parameters and Arguments

- Function definitions must include parameter list
  - Variable names for values that will be passed in when function is called
- Function call's argument list passes in those values
- **Parameter list** can be empty:
  - `function myFn1() { … }`
- If so, function is called with empty **argument list**:
  - `myFn1();`
- Otherwise, call with arguments that correspond to parameters:
  - `function myFn2(param1, param2) { … }` ex. `(firstName, lastName)`
  - `myFn2(arg1, arg2);` ex. `('Herbert', 'Hoover')`
  - Then inside the function, `firstName` variable created with value of `'Herbert'`, `lastName` variable created with value of `'Hoover'`)
  - You do not need to declare these variables using `let` or `const`! The parameter list does it for you

Scope

- Variables declared with let and const are block scoped
  - They only exist inside the code block { … } in which they were declared, such as a function, if, switch, or while/do…while block.
- Variables declared with var are function scoped
  - Function scope is like block scope, but only for functions
- Child blocks inherit their parent blocks' variables
- Variables declared outside of any code blocks are global and can be accessed from anywhere – use sparingly

The JavaScript console

- Three primary uses:
  1. View error/warning messages
  2. Log your own messages using `console.log('...');`
  3. Test out small pieces of JavaScript and have their values immediately evaluated and echoed back to you

If … Else If … Else

- Conditional statement, allows forks in your code

```
if (condition) {
    … code to execute if condition evaluates as true …
} else if (condition2) {
    … code to execute if condition2 evalutes as true …
} else {
    … code to execute if neither condition1 nor condition2 were true …
}
```

else if and else are optional, you do not need them

You can have either or both, following an if block

You can have multiple else if blocks

You can have only one else block at the very end

Comparison Operators

- Equality Operators
  - Strict equality (aka triple equals/identity): `===`
  - Loose equality (aka double equals/equality): `==`
  - Strict inequality (aka non-identity): `!==`
  - Loose inequality (aka inequality): `!=`
    - **Discuss:** What's the difference between the strict and loose versions of the equality operators, and which are best practice to use?
- Relational Operators
  - `>` `>=` `<` `<=`
  - Greater than, greater than or equal to, less than, less than or equal to
  - Works as you would expect with numbers
  - Works in lexicographical order with strings; 'a' is lower/less than 'z'

Truthy & Falsy

- Boolean values **true** and **false** are of the Boolean data type only
- The concept of **truthy** and **falsy** mean that if a value was converted to the Boolean data type, it would be **true** or **false**.
- Example: the number 3 is truthy, the number 0 is falsy
- **Discuss:** Is the number -1 falsy?

Logical Operators

- Logical And &&: Returns first falsy value or last truthy value
  - Discuss: What is returned from evaluating (true && (3 >= 5))?
    - For practice, enter into your JavaScript console to confirm your answer
- Logical Or ||: Returns first truthy value or last falsy value
  - Discuss: What is returned from evaluating (false || (5 - 10))?
    - Try this in your JavaScript console too
- Logical Not !: Coerces its operand to Boolean then returns its opposite
  - Discuss: What is returned from evaluating !(true && false)?
    - What about from evaluating !true && false? (without parentheses)
    - Use your console to confirm your answer. Discuss – why?
  - Double Not !!
    - Discuss: What is this used as a shorthand for and why/how does it work?

Truthy & Falsy (cont)

- -1 is truthy
- There are only 6 falsy values:
  - false
  - null
  - undefined
  - empty string: "" and ''
  - 0
  - NaN (Not a Number)
- Everything else is truthy!

Switch

- Conditional statement – evaluates an expression and depending on its value, executes one of multiple case clauses and an optional default clause:

```
switch (myNum) {
    case 1: console.log('In case 1');
            break;
    case 2: console.log('In case 2');
            break;
    case 3: console.log('In case 3');
            break;
    default: console.log('In default');
}
```

```
switch (myString) {
    case 'coffee': console.log('Contains caffeine');
            break;
    case 'black tea': console.log('Contains caffeine');
            break;
    case 'lemonade': console.log('No caffeine');
            break;
    default: console.log('Drink not recognized');
}
```

  - Once the program enters a case, it will executing all following statements until it reaches the end of the switch block, or a break, *even the statements for other cases.*
  - Always use a break unless you know what you're doing and you want that behavior.
  - default clause is like the "else" in an if statement, will run if nothing else matches, best practice is to always use it

More Operators: += -= ++ --

- += and -= are binary operators

```
let x = 3;
x += 5;
//x is now 8
x -= 2;
//x is now 6
```

- ++ and -- are unary operators that only add or subtract 1
  - can be used prefix or postfix and have different behaviors depending
  - recommended to use **+= 1** and **-= 1** instead of these in most cases

While Loops

- Repeat a block of code until a condition evaluates as false

```
let i = 0;
while (i < 5) {
    i += 1;
    console.log('i is', i);
}
```

```
i is 1
i is 2
i is 3
i is 4
i is 5
```

Do … While Loops

- Variant of while loops where the code block always executes at least once, even if the while condition is false

```
let i = 0;
while(i) { // i is falsy, so loop will not be entered
    console.log('Got in the loop');
}
```

```
undefined
```

```
let i = 0;
do {
    console.log('Got in the loop');
} while(i); // do ... while, so loop will be entered once
```

```
Got in the loop
```

Arrays

- Numerically indexed list of values: [item1, item2, item3, …]
- Zero-indexed – index starts at 0, not 1
- const fruits = ['apple', 'banana', 'cherry']
  - 'apple' is at index 0 and can be accessed with fruits[0]
  - 'banana' is at index 1 and can be accessed with fruits[1]
  - 'cherry' is at index 2 and can be accessed with fruits[2]
- *arrayname*.length will give you the count of items in the array
- For example: fruits.length is 3
- You can modify the value: fruits[1] = 'boysenberry'; will result in the array being changed to: ['apple', 'boysenberry', 'cherry']

Array Methods

- Some are mutator methods – they change the array
- Others are not – only access the array
- Some have parameters, others don't
- Most will return some value, different for each method
- Very useful – there are many, it will take time to learn them all

Array Methods – push(), pop(), unshift(), shift()

- push() adds an item to end of array, returns new array length
  - Use with argument of item(s) to add
- pop() removes an item from end of array, returns removed item
  - No arguments
- unshift() adds 1 or more item to start of array, returns new array length
  - Use with argument of item(s) to add
- shift() removes an item from start of array, returns removed item
  - No arguments
- All four of these are mutator methods
- **Discuss:** Which two of these four affect the index of all other items in the array and why?

Array Methods – join()

- join() – returns a string with the array items
  - Takes an argument of a string that will be used as the separator between array items in the returned string
  - If no argument is given, comma is used
  - e.g. fruits.join() will return a string of 'apple,banana,cherry
  - fruits.join('+') will return a string of 'apple+banana+cherry
  - Does not mutate the original array – the array fruits will still be the same after you use join() on it

Array Methods: includes(), indexOf()

- Both array methods will check to see if a value exists in an array
- includes(*value to check for*) will return true if so, false if not
- indexOf(*value to check for*) will return the numeric index of the item if it exists in the array, and –1 if not
- Example: for an array of: fruits = ['apple', 'banana', 'cherry'];
  - fruits.includes('banana') would return true
  - fruits.indexOf('banana') would return 1
- Discuss: Why does indexOf return -1 and not 0 for a not found item?

Math.random()

- Math.random() generates a random number between 0 and 1 such as:
  - 0.03439834432
  - 0.99999999999
  - 0
- Potential values include 0 but not 1
- If you want a value between 0 and a max number (not inclusive of the max number), multiply by the max number:
  - Math.random() * 10 would generate a random number between 0 and 9.99999999999...

Math.floor()

- Math.floor() takes a number as an argument and returns an integer
  - Math.floor(9.9999) would return 9
  - Math.floor(9.1111) would return 9
  - Math.floor(3.14) would return 3
- Use it along with Math.random() to generate a random integer:
  - Math.floor(Math.random() * 10) would generate a random integer between 0 and 9, including 0 and 9
- Add 1 to the result to get a value that's between 1 and the max number, inclusive of the max number:
  - Math.floor(Math.random() * 10) + 1 would generate a random integer between 1 and 10, including 1 and 10

If there is time...

- ...left from what the Agenda slide allocated for the Recap section, then this is the time to bring up any unresolved questions, and to discuss any Challenge Questions or Code Challenges.
- Otherwise, please continue to 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).
  - Your instructor may assign partners, or have you choose.
- 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 color-guessing-game.html page at the bottom of the assignment page in the learning portal.