Intro To JavaScript

Today's Objectives

- **What is Client-Side Scripting and Why Use it?**
- **What is Javascript?**
  - Compiled languages vs. interpreted languages
  - Statically typed vs. dynamically typed
- **Variables in JavaScript**
  - Declaring variables
  - Naming
- **Data Types**
  - Strict vs. loose equality
  - Number, String, Boolean, Object (includes arrays), undefined
  - Type coercion
  - Null vs undefined
- **Branching**
  - if/else if/else
  - switch
- **Loops**
  - for/while/do

Today's Objectives

- **Arrays**
  - push/pop
  - unshift/shift
  - indexOf/lastIndexOf
- **JavaScript Objects**
- **Functions in JavaScript**
  - Signature
  - Variable scope
- **Built-In Functions**
  - String methods
  - Numbers, Math, and Dates

What is Client-Side Scripting?

- Executes code on the user's browser, allowing us to interact with the HTML rendered and the CSS sent by the server.

- Interacts with HTML on the page (the DOM - Document Object Model).

- JavaScript is the scripting language all browsers understand.

Why Do We Use Client-Side Scripting?

- Creates less stress on the server and more interactive engaging experiences for users.
  - Allows client (browser) to perform validation immediately.
  - Fewer calls to server.
- Allows page interaction/manipulation.
  - Can respond to user events.
  - Can make calls to web services/APIs to dynamically update page.
  - Can update page without page refresh via DOM manipulation.
- Separation of Concerns.
  - HTML: Presentation content
  - CSS: Presentation styling
  - JavaScript: Behavior and logic

What is JavaScript?

- Programming language with similarities to Java.

- How is JavaScript different than Java
  - Java requires a runtime while JavaScript requires a browser.
  - Java is compiled while JavaScript is interpreted.
  - Java is statically typed while JavaScript is dynamically typed.

Adding JavaScript to an HTML Document

- **`<script>`** tag:
  - **`<script> // some JavaScript </script>`**
  - **`<script src="exercises.js"></script>`**

Declaring JavaScript Variables

- JavaScript doesn't require data type in declaration.
- Declare variables that will change using **`let`**.

```
let myText = 'Hello world!';

// can be changed

myText = 'Howdy world!';

// can also be declared without value,

let myOtherText;

// then assigned later

myOtherText = 'Hello other world!';
```

Declaring JavaScript Variables

- Declare variables that will not change using **`const`**.

```
const MY_CONST_TEXT = 'Hello world!';

// CANNOT change: below will throw an error

MY_CONST_TEXT = 'Howdy world!';

// CANNOT be declared without value:
// below will throw an error

let myOtherText;
```

Declaring JavaScript Variables

- Avoid using **`var`** - considered **harmful**!
  - Used in older versions of JavaScript
  - Allows multiple declarations without warning
  - Function scope (vs. block scope)
  - Use **`let`** or **`const`** instead

JavaScript Variable Naming

- Variable names are comprised of letters **`A-Z, a-z,`** characters **`_, $, and digits 0-9.`**
- Variable names must start with a **`letter, _, or $.`**
- Variable names are case-sensitive.
- Variable names may be not be a reserved keyword.
- Follow best practice conventions:
  - Use camelCase for multi-word variable names.
  - Use uppercase for constants and separate words with an underscore, **`_`**.
  - Boolean variable should begin with **`is`**

JavaScript Data Types

- Number
  - integer
  - floating-point
  - **`NaN`**
- String
  - Zero or more characters enclosed in double(`"`) or single (`'`) quotation marks (**`"foo"`** or **`'foo'`**).
  - Build larger strings from smaller ones in code with string concatenation using the concatenation operator, +, just as you do in Java.
- Boolean

JavaScript Data Types

`null` vs. `undefined`:

- `null` is a value of type `Object`
- `undefined` is a value of type `undefined`
- `null` must be assigned. It means **nothing**.
- `undefined` occurs from the "`let var_name;`" statement
  - It also may be assigned

JavaScript Data Types

JavaScript is loosely typed

- Variables aren't associated with any particular data type when declared and are free to hold any type of value.
- Variables can be assigned and re-assigned values of any datatype.
- JavaScript does type coercion as necessary.

JavaScript Data Types

Strict and loose equality

- === vs. ==
- === means types and values are equal (strict equality)
- == means values are equal (loose equality)
- Types are coerced
- !== and != are the "not equal" equivalents
- **Falsy** values:
  - When coerced to Boolean, value is false
  - false, 0, "", null, undefined, NaN
- All other values are **Truthy**
- More craziness: https://codeburst.io/javascript-double-equals-vs-tripleequals-61d4ce5a121a

Logical Branching

- if
- else if
- else
- switch

Looping

- for
- while
- do

```
for (let i = 0; i < 5; i++) {
    console.log("Hello world!");
}

let i = 0;
while (i < 5) {
    console.log("Hello world!");
    i++;
}

let i=0;
do {
    console.log("Hello world!");
    i++;
} while (i < 5);
```

String Interpolation in JavaScript

- String interpolation in JavaScript use the ` mark (know as a tick) to enclose the literal template.

- Values enclosed in ${} are populated with the variable name with the {}.

```
let birthDate = '03/15/1970';

console.log(`Birthdate is ${birthDate }`);
```

JavaScript Scope

- Can declare a variable at any point in a block, but you must declare it before you use it.
- Once declared, the variable is in scope.
- Variables are in scope until the end of the block when they are discarded and go out of scope.
- Nested blocks:
  - Each nested block can declare and use its own set of local variables.
  - Statements within the inner block can use both variables from the inner and outer scope
  - JavaScript allows a variable in an inner block to have the same name as a variable in an outer block (this is called variable shadowing) but this should be avoided.

JavaScript Arrays

- Defining arrays:
  - `let scores = [];`
  - `let scores = [10, 20, 30];`
- Accessing arrays
  - `scores[2];`
  - **index is 0 based.**
- Array size can be modified in JavaScript!
- Can check size of array with **`length`** property.

JavaScript Array Functions

- **`push`** adds element to end of array
- **`pop`** removes element from end of array
  - returns element removed
- **`unshift`** adds element before first element of array
- **`shift`** removes element at first element of array
  - returns element removed
- **`includes`** indicates whether an array contains a given value
- **`indexOf`** returns the index of first occurence of value in array, or -1 if not found
- **`lastIndexOf`** returns the index of last occurence of value in array, or -1 if not found

JavaScript Object Literals

- { } denotes an object
- Key : value pairs, separated by commas

```
const person = {
    firstName: 'Lisa',
    lastName: 'Simpson',
    age: 42,
    relatives: [
        'Marge Simpson',
        'Homer Simpson',
        'Bart Simpson'
    ]
};
```

- Access element: person.firstName

JavaScript Functions (JavaScript Version on Methods)

- no access modifier
- function keyword
- function name
  - usually camel-case
- no return type
- parameter names
  - no type defined
- return statement

```
function sumVals(val1, val2)
{
    return val1 + val2;
}
```

Built-In Functions

String methods

- https://www.w3schools.com/js/js_string_methods.asp
- https://developer.mozilla.org/enUS/docs/Web/JavaScript/Guide/Text_formatting

Numbers, Math and Dates

- https://developer.mozilla.org/enUS/docs/Web/JavaScript/Guide/Numbers_and_dates
- https://www.w3schools.com/js/js_number_methods.asp
- https://www.w3schools.com/js/js_math.asp
- https://www.w3schools.com/js/js_dates.asp

Using the Dev Tools JavaScript Debugger

Exercise Notes

- exercises.js
  - `createObject()` is intended to be a function that returns an object with the given fields and using your info for the values in those fields
- challenge-exercises.js
  - `titleCase`:
    - Words not in minor words list should be Pascal case
    - Words that ARE in minor words list show be all lowercase