Introduction to JavaScript

JavaScript is everywhere - all your favorite websites and also the ones you don't like use JavaScript. Makes your web content come to life - JavaScript allows for interaction with content and makes things happen. JavaScript is the dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on websites. Used in all browsers it's the most popular coding language ever. Websites and web apps everywhere are using JavaScript. It runs directly in your browser and you can create html files with a text editor directly on your computer to run in your browser. Select the html file and open it with any browser.

Code is a set of instructions for what you want to happen. Example : When a new person comes to your website, ask their name. Showing a welcome message with their name would be an example of something you might ask JavaScript to do. The instructions for the code would be to provide the user an input area, get the input value of their name, then use that value to return a response. Select a page element and update the contents of the element with the welcome message including the value of the input for the user's name.

The developer console shows you information about the currently loaded Web page, and also includes a console that you can use to execute JavaScript expressions within the current webpage. Open your browser, select the devtools and try it out, we will be using it in the lessons of this course. With most modern browsers you can write and execute javascript

directly from your browser. Within the Chrome browser you can open DevTools when you want to work with the DOM or CSS, right-click an element on the page and select Inspect to jump into the Elements panel. Or press Command+Option+C (Mac) or Control+Shift+C (Windows, Linux, Chrome OS). When you want to see logged messages or run JavaScript, press Command+Option+J (Mac) or Control+Shift+J (Windows, Linux, Chrome OS) to jump straight into the Console panel.

You will need an editor to write your code. You can use Visual Studio code if you don't already have an editor. First lesson will help you set up and start coding on your computer.
Getting started with JavaScript is easy: all you need is a modern Web browser. Create an index html file and open it in a browser Add JavaScript to the html either linking to a script file or JavaScript directly between the script tags in the HTML.

By the end of the first lesson you should have your editor and browser setup and ready to code. You can download an editor at https://code.visualstudio.com/ if needed. Start by creating an index.html and app.js file. Then write some JavaScript code and try it out. Open and run the HTML file in your browser. You can use the functions from the lesson such as alert() or document.write('Hello'); Alert stops the code execution, if on top it does not output the content until the button is clicked. Place JavaScript at the bottom so the rest of the code can render.

Create an HTML file and link to a JS File

```
<!DOCTYPE html>
<html>
   <head>
       <title>JavaScript Course</title>
   </head>
   <body>
       <div class="output">Output 1</div>
       <div class="output">Output 2</div>
       <script src="app1.js"></script>
   </body>
</html>
```

```
document.write('Hello World 2');
alert('Hello');
```

Getting Started with JavaScript

Double quotes and single quotes or backticks can be used to contain string content. A semicolon is not necessary after a statement if it is written on its own line. But if more than one statement on a line is desired, then they must be separated by semicolons. It is considered best practice, however, to always write a semicolon after a statement, even when it is not strictly needed. Whitespace is ignored in JavaScript code, you can use it for writing code that is more readable and understandable.

Creating variables -

var - Declares a variable, optionally initializing it to a value.
let - Declares a block-scoped, local variable, optionally initializing it to a value. Blocks of code are indicated by {}
const - Declares a block-scoped, read-only named constant. Cannot be changed.

Variables must be declared before you use them. Comma separate to declare multiple variables. Camelcase is more readable as in the example below.

let a,b,c,d;
let userfirstname = "Laurence";
let userFirstName = "Laurence";

Rules for naming variables must start with a letter, underscore (_), or dollar sign ($). Subsequent can be letters of digits. Upper or lower case. No spaces. No limit to the length of the variable name. Variable names are case sensitive. Cannot use reserved words.

```
let first1Name = "Laurence \"Hi\" ";
let _$lastName = 'Svekis \'Hello\' ';
let fullName = `Laurence's "Svekis" `;
const id = '100';
const ID = '500';
firstName = "New Name";
fullName = `Laurence Svekis`;
//id = '1000';

///console.log(firstName);
//console.log(lastName);
console.log(fullName);
```

Laurence Svekis - [JavaScript Creating Dynamic and Interactive Web Pages for Beginners]

```
console.log(id);

//document.write('Hello World 2');
console.log('Hello');  // Test message
//alert('Hello');
/*
Multiple
line
comment
*/
```

JavaScript Data Types and JavaScript Objects

JavaScript uses different data types, JavaScript dynamically assigns the data type that it presumes is desired, you can also change data types of the variable if needed. Arrays are objects that have a preset order of items, using the index of the item to select and identify the item within the array list. Arrays also have built in methods which make them a powerful way to use the data and manipulate and select items contained in the array. Objects also can contain multiple items in the same variable, they are identified by a property name which is used in order to select the item from the object. Each property name can only be used once and is unique within the object. Property names can be set with quotes or as single words within the objects. They get assigned values as a pair with the property name, using the colon to assign the value and comma separate multiple named pair values.

```
let num = 100;
let num1 = '100';
let num2;
let num3 = null;

const arr = ['hello',1,2,'four',true,[1,2,3]];
const arr1 = arr;
const myObj = {
   first : 'Laurence',
   last : 'Svekis',
```

```
   "full name" : 'Laurence Svekis',
   id : 100,
   id0 : 1,
   id1 : 2
};
myObj["full name"]= "LS";
let val = "id0";
console.log(myObj[val]);
val = "id1";
console.log(myObj[val]);
console.log(myObj);
console.log(arr1);
arr[1] = 'one';
console.log(arr);
console.log(typeof(arr));
console.log(arr1);
num = '200' + num ;
num1 = num1 + 100;

let boo = true;

//console.log(typeof(boo));

/*
console.log(typeof(num));
console.log(typeof(num1));
console.log(typeof(num2));
console.log(typeof(num3));
*/
```

JavaScript Functions

Functions provide a powerful way within code to run blocks of code, also they contain their own scope so variables set within the function can live only within that function. Create functions in code to do repeat code tasks, and handle specific coding objectives. You can pass values into a function then use those values within the function code, and return a result back from the function code. Functions can use arguments within the parameters, although they are not mandatory. Function also can use return although not mandatory to return a response from the function. You can pass values into function to be used within the code.
There are two types of functions, function declaration which uses the keyword function to assign the function code, or a function expression which is similar to assigning a variable a value, but in this case it's the function code.

```
// Function declaration
function test(num) {
      return num;
}
// Function expression
var test = function (num) {
      return num;
};
```

```
let val = "Laurence";
let counter = 0;
const fun1 = function(){
   val = val + "Svekis";
   counter++;
   console.log(val);
   console.log(counter);
}
fun2();
fun2();
fun2();
fun2();
function fun2(){
   val = val + "Svekis";
   counter++;
```

```
   console.log(val);
   console.log(counter);
}



function fun3(){
   let val = "Svekis";
   let val1 = 100;
   console.log(val);
}
console.log(val);
fun3();
console.log(val);
//console.log(val1);


function fun4(a,b,c){
   console.log(a,b,c);
   let val = a + b * c;
   console.log(val);
}

fun4(3,6,8);
fun4(233,1236,28);

function fun5(a,b,c){
   counter++;
   let val = a * b * c + counter;
   return val;
}
```

```
let output = fun5(4,7,8);
console.log(output);
output = fun5(4,7,8);
console.log(output);
```

Interactive Web Pages JavaScript

JavaScript can connect to web page elements using the document object. The document object is built by the browser as a tree type structure to represent the entire web page contents of elements and their properties. Using JavaScript to access the DOM or Document Object Model from the browser allows us to within the code select page elements, and even update the page elements. The document object is a massive object that has its own set of methods and properties values that can all be selected and manipulated with JavaScript Code.
Using querySelector or querySelectorAll make a selection of your page elements, assign a variable to the element object in order to be able to easily use that object within your code.
You can update the textContent of the page element using the property value of textContent.
You can add event listeners on your elements with the onclick event object, assigning a function that gets invoked once the event is triggered. Try it out select a page element, add an event listener and then make some updates to the element using the various element property values.

```
<!DOCTYPE html>
<html>
   <head>
       <title>JavaScript Course</title>
   </head>
   <body>
       <div class="output">Output 1</div>
       <div class="output">Output 2</div>
       <div class="output">Output 3</div>
       <script src="app4.js"></script>
   </body>
</html>
```

```
let counter = 0;
const output1 = document.querySelector('.output');
```

```
console.log(output1.textContent);
output1.textContent = "Hello World";
console.dir(output1);
const outputs = document.querySelectorAll('.output');
outputs[1].onclick = fun1;
outputs[0].onclick = function(){
   counter++;
   output1.textContent = 'Clicked '+counter;
   console.log(counter);
}
//document.write('Hello');
outputs[2].addEventListener('click',fun1);
function fun1(){
   counter++;
   outputs[1].textContent = 'Output '+counter;
   console.log('hi '+counter);
}
```

Logic Conditions with JavaScript

Conditions can be used within code to apply logic, and run different actions based on the result of the condition. Depending on the value either True or False the code can run different blocks of code.
The if statement will check a condition, if true it runs the specified code. If false it does not run the code. Else provides an alternative if the condition is not true then else statement allows us to run a block of code on false if none of the previous conditions are true. You can also use the else if statement to check another condition with a separate block of code to be run if the previous condition came back as false.
Using the switch statement allows us to specify several alternatives checking to see if any of the conditions match and allow the corresponding code to be executed.
JavaScript code can use Comparison Operators to check if a statement is true or false. To check multiple conditions you can apply Logical Operators to check logic between conditions. The example below will show how to use the Logical operators and the results that can be expected with the various combinations of true or false.

```
> (true && true)
<· true
> (true && false)
<· false
> (true || false)
<· true
> (false || false)
<· false
```

```
const output = document.querySelector('.output');
if(true){
   console.log('was true');
}
if(false){
   console.log('was false');
}
let a = 8;
let b = "5";
let c = a + b;
let d = 10;
let val;
if(a >= 5){ val = `${a} was greater than 5`; }
if(b === 5){ val = `${b} is equal to ${5}`; }

if((a > d) && (a > 0)){
   val = `${a} > ${d}`;
}else{
   val = `Not True ${a} > ${d}`;
}

val = (a > d) ? "A is Higher" : "A is Lower";
```

```
if(5 > 10) { val = 'true';}
   else if(10 == 5) { val = 'nope';}
   else{ val = 'False';}

switch(a){
   case 5:
       val = '5';
       break;
   case 15:
       val = '15';
       break;
   default:
       val = 'Non were true in the switch';
}

console.log(val);
output.textContent = val;
```

JavaScript Loops For and While

Loops allow us to execute blocks of code a number of times, they also allow us to iterate through a list of items such as items in an array or other iterable list.
Loops will always require several parameters, such as a starting value, a condition to stop the loop and a way to move through the items to the next item in the loop. We can setup a simple for loop by setting a variable to use with a starting value, then applying a condition to continue the loop if the condition is true, and incrementing the value of the variable so that eventually the condition is no longer true.

```
let counter = 0;
const arr = ['a','e',13,14,5];
const myObj = {
   a : 'one',
```

```
   b : 'Three',
   message : 'Hello World'
};
for(let i=0;i<10;i++){
   counter++;
   //console.log(i,counter);
}
for(let i=0;i<arr.length;i++){
   console.log(arr[i]);
}

arr.forEach((val,ind,arrV)=>{
   console.log(val,ind);
})
console.clear();
for(val in arr){
   console.log(arr[val]);
}
for(val in myObj){
   console.log(myObj[val]);
}
for(val of arr){
   console.log(val);
}
console.clear();
let i = 1;
while (i<10){
   console.log(i);
   i++;
}
i = 100;
```

```
do {
   console.log(i);
   i++;
}
while(i<10);
console.clear();

const outputs = document.querySelectorAll('.output');
console.log(outputs);
outputs.forEach((ele,ind)=>{
   console.log(ele.textContent);
   ele.textContent = `New ${ind+1}`;
   ele.val = 0;
   ele.onclick = function(){
       ele.val++;
       ele.textContent = `Click ${ele.val}`;
   }
})
```

Built In methods JavaScript

JavaScript has many built in methods, which are prebuilt functions that allow us to perform a certain predefined action. Math is an object in JavaScript that allows you to perform mathematical tasks on numbers, we can also generate random values with Math.
String methods help you to work with strings and do manipulations of the string values. Strings all have a length property with a returned value of the length of the string. Every character in the string has its own index value which allows us to use JavaScript code to select those characters.

```
let val;
let a = ' Hello World ';
```

```
a = a.trim()
val = a.slice(0,5);
val = a.substring(6,12);
val = a.substr(-5);
val = a.replace('World','Coders');
val = a.toUpperCase();
val = a.search('llo');
val = a.indexOf('ll');
//val = a.trim();

const outputs = document.querySelectorAll('.output');
outputs.forEach(ele=>{
   let myStr = ele.textContent;
   myStr = myStr.replace('Output','new words').toUpperCase();
   ele.textContent = myStr;
   ele.onclick = function(){
       ele.textContent = ran(1,total);
   }
})
let total = 100;
//val = ran(1,total);

console.log(val);

function ran(min,max){
   return Math.floor(Math.random() * (max - min)) + min;
}
```

JavaScript List Maker

Course project is designed to challenge you in applying what you have learned in the previous lessons. Use JavaScript to transform the index.html code, with three divs. Change the page elements to have one as an updatable field, the second into a submit button, and the third into a dynamic list that adds new items when the submit button is pressed. Take the value of the intake field, when the button is clicked and add that value as a new list item to the list in the last element. Clear the first element content and allow users to once again update the element with new content to be again added to the bottom of the list. Create event listeners so that when list items are clicked they update the font color either to red or toggle to black.

```html
<!DOCTYPE html>
<html>
   <head>
       <title>JavaScript Course</title>
   </head>
   <body>
       <div class="output">Output 1</div>
       <div class="output">Output 2</div>
       <div class="output">Output 3</div>
       <script src="app8.js"></script>
   </body>
</html>
```

```javascript
const outputs = document.querySelectorAll('.output');
outputs[0].textContent = '';
outputs[0].style.border = '1px solid black';
outputs[0].setAttribute('contenteditable', 'true');

outputs[1].style.background = 'red';
outputs[1].textContent = 'SUBMIT';
outputs[1].style.textAlign = 'center';
outputs[1].style.padding = '5px';
outputs[1].style.color = 'white';
```

```
outputs[1].addEventListener('click', (e) => {
    //add to list
    let myStr = outputs[0].textContent;
    if (myStr != '') {
        outputs[0].textContent = '';
        const li = document.createElement('li');
        li.textContent = myStr;
        li.onclick = function () {
            if (li.style.color == 'red') {
                li.style.color = 'black';
            } else {
                li.style.color = 'red';
            }
        }
        ul.append(li);
    }
})

outputs[2].innerHTML = '';
const ul = document.createElement('ul');
outputs[2].append(ul);
```