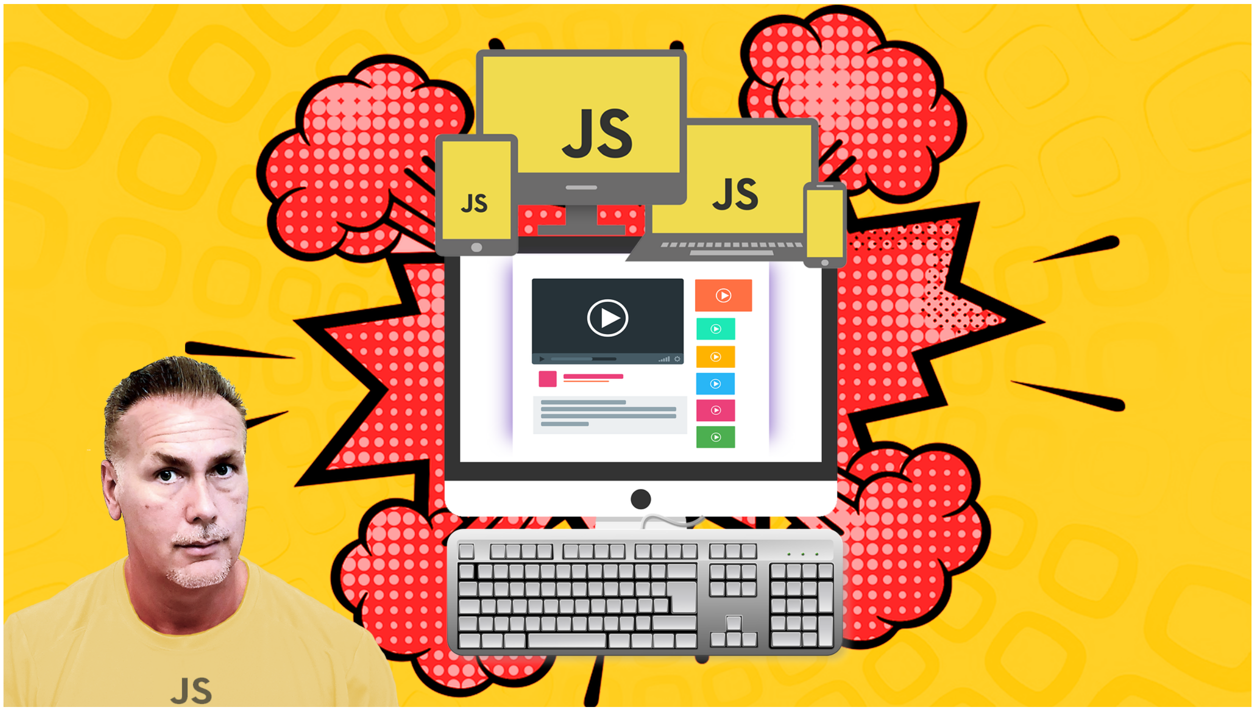
Dynamic Interactive Web Pages with JavaScript

JavaScript DOM Introduction to Interactive Dynamic WebPages



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 and then bring in the JavaScript code to make things happen.

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 Google to find a list of Code editors, almost all will support frontend code. You can download an editor at https://code.visualstudio.com/ if needed.

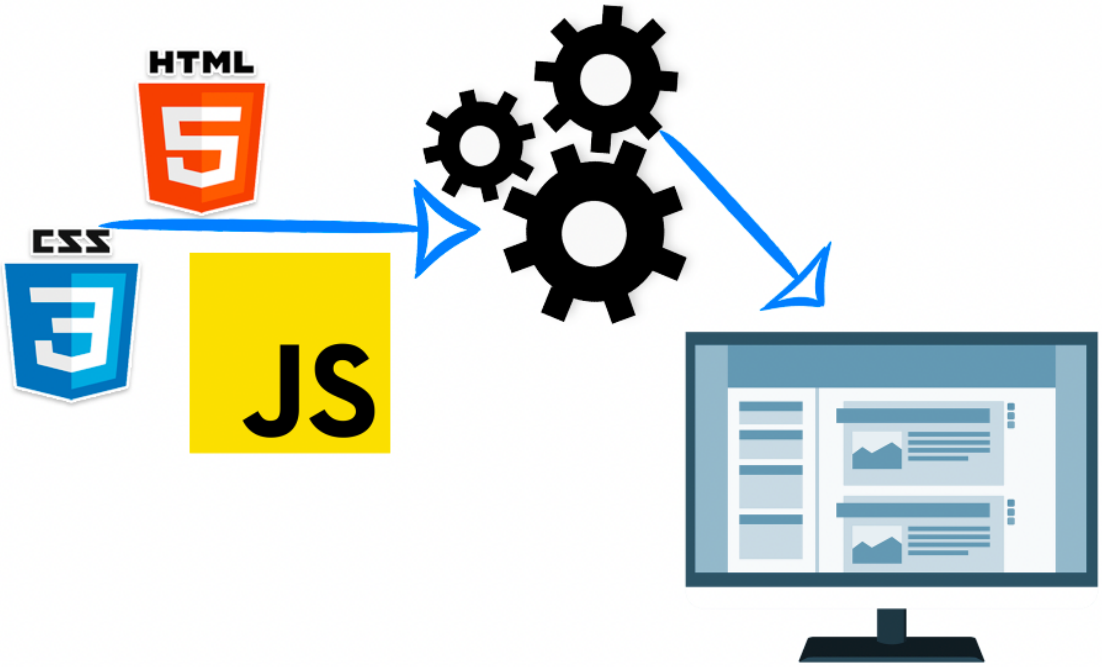
Introduction to how JavaScript works and how you can write JavaScript code

- JavaScript is a core technology on the web alongside HTML CSS
- JavaScript runs in the browser and gets rendered by the browser within the HTML page.
- You can connect to HTML elements, using the HTML DOM which is a programming interface for JavaScript to add, update and remove HTML elements.
- HTML elements can be accessed as objects that have properties, methods, and events.
- Code is a set of instructions that tells the application in a language that can be understood by it, what to do.
- JavaScript code is single threaded, which means that only one thing can happen at a time on the thread. Other requests get blocked until an operation completes and then those requests will run.
- How to write code using a code editor or IDE.
- How the elements from the HTML code are represented within the DOM tree structure, and that you can navigate from one to another.
- What Devtools are and how you can use them
- How you can use console log and console dir methods within the DevTools to view the document object and see the contents of the page elements in the DOM.
- How to add JavaScript code within your HTML file and run the JavaScript code in your browser.
- JavaScript commenting in the code and how to make your code more readable.

HTML is the markup language used to structure and provide meaning to the web content. CSS is the language used that contains the rules for styling the HTML content. JavaScript is the scripting language that allows you to create interactive and dynamic content within the web page.

You can do a lot with JavaScript such as store values, run blocks of code, apply logic with conditions and add events to your page elements. There is a lot you can do with JavaScript and the possibilities are endless.

APIs- are sets of prebuilt building blocks that allow developers to connect into and access these objects.   Think of it like a control panel which JavaScript language lets you interact with. Browser APIs allow access to the web page elements, and other data. You can access the API and do useful programming things.   The DOM (Document Object Model) API allows you to manipulate the HTML and CSS of the page.



JavaScript is a lightweight interpreted programming language. As with interpreted languages this means that the code is run from top to bottom, and returned immediately. The browser runs the code directly from the original file without having to compile the code.

Server side vs Client side, means where the code is run. The Server side code runs on the server and not within the user's computer, or browser. Client side code runs directly within the user’s computer, as when it comes to JavaScript. When the user views a web page in a browser the code is downloaded, run and displayed by the browser.

JavaScript is dynamic as it has the ability to update the display contents of the web page, and also generate content. JavaScript works within the browser on the client or user's computer. This is why we can run the code without an internet connection, as everything we need is already on the computer in the browser application. A web page that has no updating content and shows up the same all the time, is referred to as static.

Lesson JavaScript Coding Exercise :

1. Open your code editor and create a basic html file.
2. Save the html file and open it within a browser.

```
<!doctype html>
<html>
<head>
  <title>JavaScript</title>
</head>
<body>
 Hello World
</body>
</html>
```

Setup of your web development environment, getting started writing code. Use of editor visual studio code. Use of Chrome browser and Chrome DevTools to see console messages.

Getting started writing code how to code with JavaScript adding JavaScript to your HTML File

How to Add JavaScript to HTML

You can add JavaScript as an attribute inline to an HTML element although this format is not suggested. They are hard to identify and not good practice.

```
<div onclick="alert('hello')">Hello World 1</div>
```

Internal JavaScript within the HTML script tags. Should be added just before closing the body tags, if you are going to be interacting with the page elements. The code loads into the browser from the top down, and if the HTML hasn’t fully loaded and you try to access the element using JavaScript you will encounter an error.

```
<script>
  document.write('<div>Hello World 2</div>');
</script>
```

Best practice is with an external file, using the js extension. Place the script tags, and use the src attribute link to the path of where your file is located. It can either be relative to where the html file is or absolute using a file address including the full path to load it. If you use absolute then you should be able to paste the path into your web browser and see the file.

```
<script src="app.js"></script>
```

Example code to output text into a web page, using the document.write method.

```
document.write('<div>Hello World 3</div>');
document.write('<div>Hello World 4</div>');
document.write('<div>Hello World 5</div>');
```

Lesson JavaScript Coding Exercise :

1. Create an HTML file and a JS file.
2. Use the script tags to link to the JS file as the source file.
3. Using the document.write() add text to the webpage.
4. Using the document.write() add html formatted content to your webpage.

JavaScript Code Tips and Learn how to start with JavaScript

- Use Comments when possible single or multiple line
- Indent using whitespace to make the code readable
- Create code in a separate JS file and link to it from the HTML file
- Use the console for debugging and to see values

Use of console.log() to output debugging data to the browser.

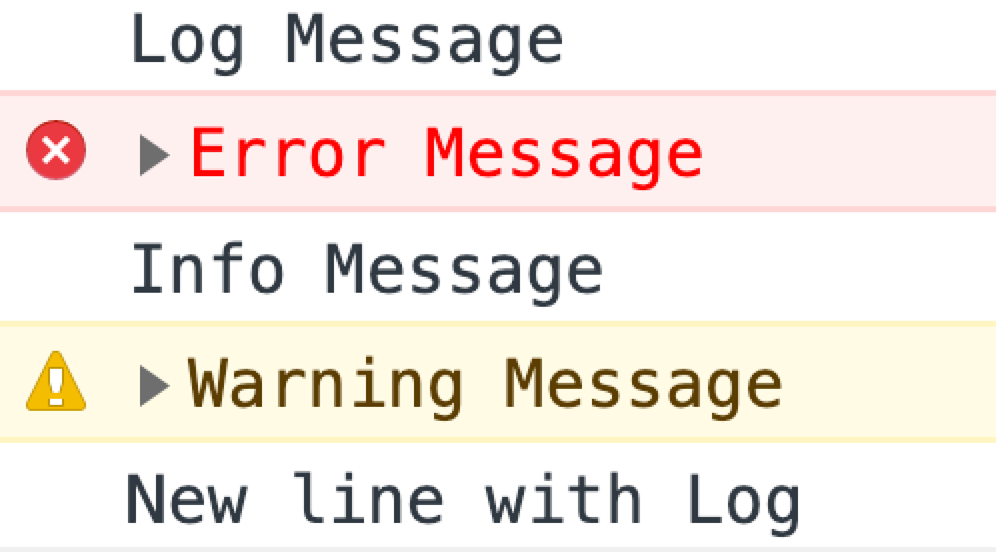
Whitespace is ignored in the code, use whitespace and indentation to make the code more human readable. Be careful not to add spacing between the syntax statements, as this can cause errors when the code is being rendered. Blocks of code can be indicated using the curly brackets, this will be used within the various code syntax in the upcoming lessons.

```
{
   document.write('<br>Output Message');
   document.write('<br>Output Message');
}
```

Use commenting of single line or multiple line in order to provide more context of the code. You can debug your application by commenting statements out with the // . Comments are ignored by the browser and will not output into the page.

```
/*
This is my code!
Laurence Svekis
*/
//alert('Hello');
```

Use of console methods allows you to debug and communicate from the JavaScript code into the browser console. This can be used to provide additional information from the JavaScript code as it runs. Console messages are only output into the console of the browser and not visible by web users.



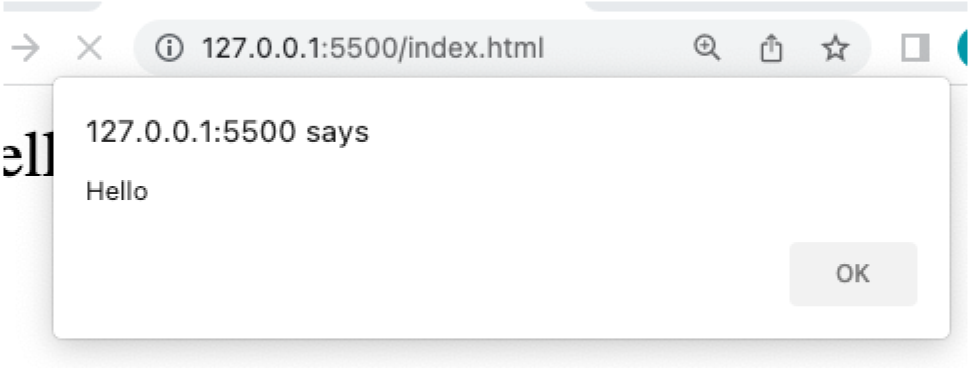
```
Log Message
✖ ▶ Error Message
Info Message
⚠ ▶ Warning Message
New line with Log
```

```
console.log('Hello');
console.error('Hello');
console.info('Hello');
console.warn('Hello');
//console.clear();
console.log('New line');
```

Window object is the parent of the document object. The window object is the top parent and contains various methods that can be used. The alert() and the prompt() are both part of the windows object. You can use these to create an interaction with the user. Both will show a standard popup message in the browser, which must be actioned before the user can interact with the rest of the browser window. Although they provide a simple way to interact with the user, these are not commonly used, since the user experience is poor and there are more interactive options now in JavaScript and the DOM. Alerts and prompts are meant for interaction with the web user.

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.

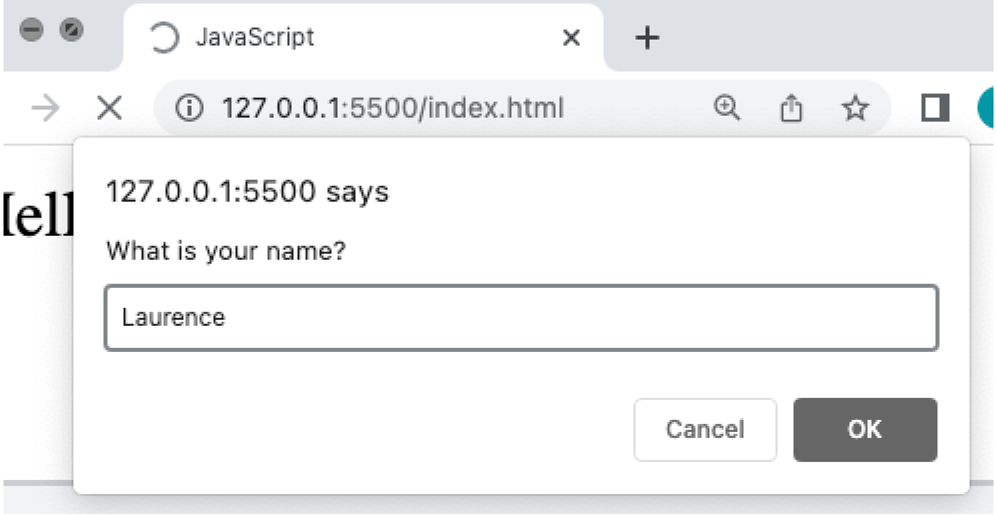
Alert window.



127.0.0.1:5500 says

Hello

Prompt window with response back



127.0.0.1:5500 says

What is your name?

Laurence

```
alert('Hello');
const res = prompt('What is your name?');
```

Lesson JavaScript Coding Exercise :

1. Comment out a line of code in your JavaScript file.
2. Comment out a block of code with a multiline comment
3. Try the various console output methods with different messages
4. Create an alert with your name as the message
5. Create a prompt that returns the name of the web user. Set a variable to capture the response value of the prompt and output the value into the document using document.write.

How to use variables in JavaScript

- Variables are one of the basic concepts of coding
- Using const or let avoid using var
- const and let are scope based
- Parent scope vs local scope
- Assign values to variables and reassign new updated values
- Use of strings and numbers as values for variables
- Dynamic type with JavaScript changing data type
- Math and output from JavaScript directly

Variables allow developers to hold values, these values can be updated as needed within the code. Typically variables are declared at the start of your code, and used throughout the code to hold application values. The values can change using “**let”** when you declare a variable. Use “**const”** when you want the value to stay the same throughout the code. This can save errors and issues with reassigning values mistakenly in the code.

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”;
```

Lesson JavaScript Coding Exercise :

1. Declare variables both numbers and strings using both let and const
2. Try the various formats to create strings using the single, double quotes and backticks
3. Output the results into the console and to the document.
4. Assign new values to the variables and output the results
5. Try the template literal strings with JavaScript wrapped in the ${} and check the output in both the console and webpage

```
let a = 'Laurence Svekis';
let b = 10;
let e = 5;
let f = 20;
let myStr1 = 'Hello\'s "world"';
let myStr2 = "\"Hello\" 'world'";
let myStr3 = `Hello'      s
"world"`;
```

```
let tempLit = `Math ${b + e + f} Hello`;
const myName = 'Laurence Svekis';
//myName = 'Svekis';

{
   const b = 1000;
   console.log(b);
   //var c = 100;
   //let d = 100;
}
//console.log(c);
//console.log(d);
console.log(b);
console.log(myName);
a = 'Hello World';
let val = b + e + f;
val = myStr1 + ' ' + myStr2 + ' ' + myStr3;
val = tempLit;
val = `${myStr1} ${myStr2} ${myStr3}`;
console.clear();
console.log(val);
document.write(val);
```

JavaScript Dynamic Type Conversion

- JavaScript DataType
- Data Types string, number, boolean - typeof type conversion
- Data Types Booleans Null Undefined
- number.toString() Number(string)
- typeof operator

- Data types include numbers, strings, booleans, objects

JavaScript Dynamic Type Conversion and how it works. JavaScript variables can be converted in different data types, they are not set to the data type that they are declared with. 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.

Lesson JavaScript Coding Exercise :

1. Declare multiple variables without assigning values to them
2. Get the data type of a variable and output it to the page
3. Add numbers and strings together
4. Convert a string to a number
5. Convert a number to a string
6. Create boolean values

```
let myStr = 'Laurence';
let myNum = 0;
let a;
let b,c,d;
let val;

myStr = 5;
myNum = 'test';

b = 5;
c = 10;
d = Number('20');
d = null;
d = true;
d = false;
d = c.toString();
a = '10a00X';
d = parseInt(a);
```

```
val = d + b + c ;
val = 100 + d;
val = typeof(d);

console.log(val);
document.write(val);
```

Variable naming Rules how to declare JavaScript Variables

- Variable identifier names must use unique names
- Variable identifier names should be meaningful and semantic if possible
- Identifiers can contain letters, digits, underscores and the dollar sign
- Variables names must begin with a letter, underscore or dollar sign
- Names are case sensitive
- Names cannot be the same as JavaScript reserved words
- No spaces in the variable name

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.

Lesson JavaScript Coding Exercise :

1. Create some string values
2. Assign meaning names to the variable identifier names
3. Use camelCase for a variable name

```
let _test5 = 'Laurence';
let $test_ = 'Svekis';
let test$ = 'hello';
let TESTtest = 'hello';
let myFirstName = 'Laurence';
//let let = 'Not a good idea';
let myLastName = $test_;
let output = myFirstName;
let fullName = `${myFirstName} ${myLastName}`;
output = fullName;
console.log(output);
document.write(output);
```

Arrays and Objects in JavaScript coding examples

Both arrays and objects in JavaScript provide a powerful way to hold content and use data within code. With one variable you can hold multiple values within both arrays and objects, in addition you can use combinations of these as they hold all data types, allowing the data contained within them to go multiple levels deep as needed.   Arrays and objects can be declared using const as the variable points to a location within the memory and not the to value that it is assigned to, which makes it possible to update the objects and arrays contents without actually having to reassign the value to the variable. Typically if you do use an array or object then you won't want to change it, as when you assign the variable a new value it would remove all the items contained within these.

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.

Lesson JavaScript Coding Exercise :

1. Create an array with various data types.
2. Create an object with various data types
3. Retrieve a value from both the array and object and output it into the console.
4. Try using both dot notation and bracket notation to retrieve values within an object.
5. Update the values of the array and object using the index value for the array and property name for the object.
6. Declare a new variable and assign the existing array to the new variable. Make updates to the contents of the new variable and see how it also updates the first array.

```
const arr1 = ['one',100,false,null,[1,2,3]];
const obj1 = {
   first:'Laurence',
   'full Name':'Laurence Svekis',
   id : 100,
   status : true,
   arr : [1,2,3,'Hello World']
};
const arr2 = arr1;
const obj2 = obj1;
obj2.id = 5000;
arr2[0] = 'Hello World';
arr1[3] = `Laurence Svekis`;
obj1.first = `Linda Jones`;

let val = arr1[4][0];
val = obj1['full Name'];
val = obj1.first;

val = obj2.arr[3];

console.log(val);
document.write(val);
```

JavaScript Functions how functions can be used in code

- functions can be used to run blocks of code
- two types of functions Function Expression and Function declaration
- Functions can take
- Function naming are the same rules as variables

- You can pass in values into the function within the parentheses, parameter names can be separated by commas to add multiple values
- When the function is invoked the parameters are known as function arguments which are the values received by the function in the parameters
- Functions can return values, the return statement will end the function execution and return the value.
- Functions have scope within the curly brackets. Local values within the function scope can only be accessed within the function
- Benefit of functions is that you can write the code once and reuse it as many times as needed. You can get different results with different argument values and return values.

Functions allow developers to run a block of code,when the function is run it's defined invoking the function when it's executing the code. 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;
};
```

Lesson JavaScript Coding Exercise :

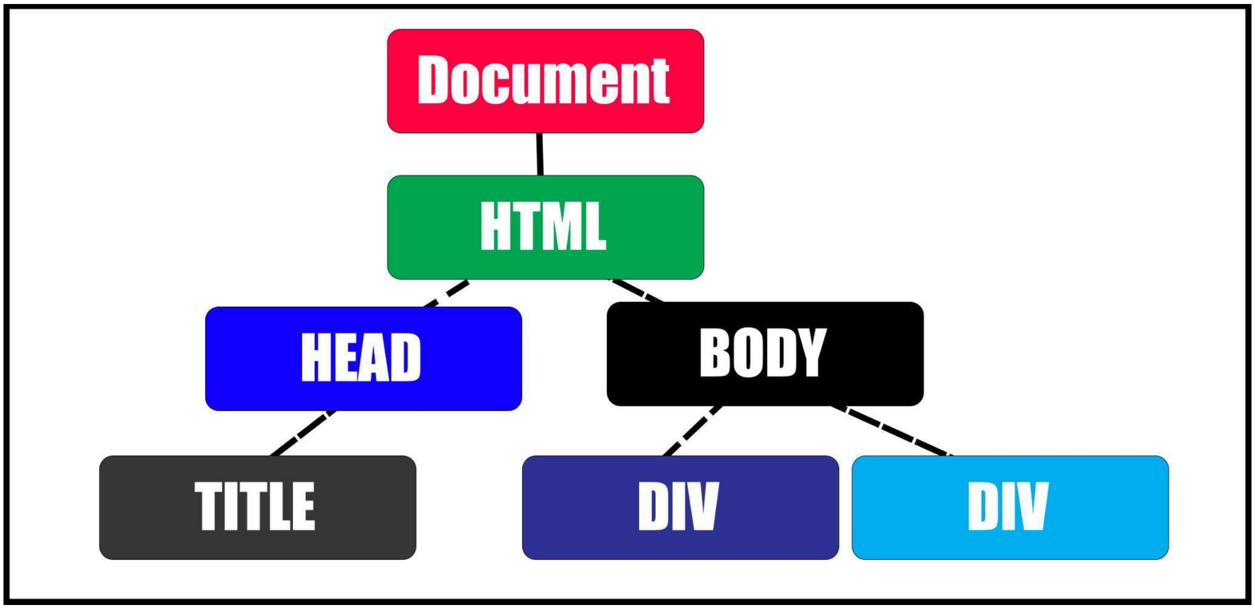
1. Create a global value that will be incremented every time the function is executed. Add this increment of the value to each function. counter++;
2. Create a function expression that within its code can output a value into the console
3. Invoke the function. Update the function with a parameter, where the function uses that argument value and outputs that value into the console
4. Create a function declaration.   Add 3 parameters, that will take the value and add them all together. Use return to return the resulting total of the values that were passed into the function.

5. Assign the returned value of the function to a variable that is output into the web page.

```
const fun1 = function (val) {
   counter=counter+val;
   const output = `<div>Hello ${counter}</div>`;
   document.write(output);
}
let counter = 0;
let word = '';
fun1(21);
fun1(2);
fun1(3);
fun2('h');
fun2('e');
fun2('l');
fun2('l');
fun2('o');
//fun1 = 100;
console.log(fun1);

const fun5 = fun1;
fun5(1000);

function fun2(val){
   word = word + val;
   counter++;
   let output = `<div>Hello ${counter}</div>`;
   output += `<div>Hello ${word}</div>`;
   document.write(output);
}
let valTotal = 0;
valTotal += fun3(10,44,55);
valTotal += fun3(4,66);
valTotal += fun3(5);
console.log(valTotal);

function fun3(num1=0,num2=0,num3=0){
   const total = num1 + num2 + num3;
   console.log(valTotal);
   return total;
}

const temp = 5000;
fun4(100);

function fun4(val){
   const temp = 9;
   const myVal = val + temp;
   //console.clear();
   console.log(myVal);
}
```

JavaScript Document Object Model for interactive web pages

The Document Object Model (DOM) is an object that contains a data representation of the page elements. The DOM is structured in a tree like format, as objects that comprise the web page and content of the HTML web document. Document Object Model (DOM) is a programming interface for HTML documents, that is the logical structure of a page and how the page content can be accessed and manipulated. Bring your web pages to life with JavaScript and connect to the web page elements. Accessing the DOM you can create fully interactive content that responds to the user. DOM and JavaScript lets you create Dynamic web page content that can change without page refresh and present new elements and updated content to the user. Improve your web users experience with JavaScript and the DOM.



Lesson JavaScript Coding Exercise :

1. Create your HTML file add some page elements
2. Assign to a variable the document.body.children array which will represent all the child elements within the body.
3. To select the first one use the index value of 0, as just like arrays these are zero based. Output the object to the console with console.dir()
4. Select some properties from the element, output them into the console.
5. Update the textContent of the element to a value of your name.
6. Go to your favorite website and open the devTools, type the document.body.textContent = "Laurence Svekis"

```
<!doctype html>

<html>

<head>

  <title>JavaScript</title>

</head>
```

```
<body>
 <div id="one" class="red">Hello World 1</div>
 <div>Hello World 2</div>
 <script src="dom1.js"></script>
</body>
</html>
```

```
const ele = document.body.children[0];
console.dir(ele);
let val = 'Laurence Svekis';
document.body.children[0].textContent = val;
ele.textContent = 'UPDATED';

console.log(ele.className);
console.log(ele.innerHTML);

ele.innerHTML += '<h1>Hello</h1>';
```

DOM methods with JavaScript select HTML page elements

There are methods in the Document object that allow us to better select elements we want to manipulate with code. querySelector() and querySelectorAll() allow JavaScript to select page elements from the content within the document object.

JavaScript querySelectorAll Get Page Elements Select ALL. Use of querySelector and querySelectorAll to select matching page elements. Different selectors including tag, id, and class. Select page element, iterate contents of node list output and update page elements.

Lesson JavaScript Coding Exercise :

1. Update your HTML to include 2 div's and an input element.

2. Using querySelector() and querySelectorAll() select all the page elements and assign variables to the page element objects.
3. Update the text content of the first element
4. Assign a value to the input element myInput.value
5. Update some of the style properties of the second div to make it look more like a button.
6. Add an event listener to the button, that assigns a function to the element click event onclick
7. Within the click function get the value of the input, update the textContent of the first div to be the value of the input field. Assign a black string to the input field value.

```
<div id="one" class="red">Hello World 1</div>

<input type="text" >

<div >Hello World 2</div>
```

```
const ele1 = document.querySelector('div');

const ele2 = document.querySelector('#one');

const ele3 = document.querySelector('.red');

ele1.textContent = 'Laurence Svekis';


const eles = document.querySelectorAll('div');

console.log(ele1);

console.log(ele2);

console.log(ele3);

console.dir(eles);


eles[0].textContent = 'Hello World';

console.log(ele1.textContent);



const myInput = document.querySelector('input');

console.log(myInput.value);

myInput.value = 'Laurence';
```

```
eles[1].textContent = 'Click Me';
eles[1].style.border = '1px solid black';
eles[1].style.width = '200px';
eles[1].style.textAlign = 'center';
eles[1].onclick = clicker;

function clicker(){
   let temp = myInput.value;
   ele2.textContent = temp;
   myInput.value = '';
}
```

Logic Conditions with JavaScript if Statement Switch

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.

Lesson JavaScript Coding Exercise :

1. Select the page elements you want to use assign variables to them
2. Create a global value for counter of 0
3. Once the button is clicked increment the counter by 1

4. Check to see if the counter value is less than 2, if that's true then update the style color to red
5. Add an else if to the condition, checking if the value of counter is less than 4, if it is then update the style color to green for the output element.
6. Add an else if to the condition, checking if the value of counter is less than 6, if it is then update the style color to purple for the output element.
7. Using else if none of the conditions are true update the style of the element to be blue
8. Create a switch to check for values of the counter. Create cases for the values you want to track, update the output area text with different string values depending on the case. Try the switch with and then without the break keyword.
9. Add in a default within the switch statement

```
<!doctype html>
<html>
<head>
  <title>JavaScript</title>
</head>
<body>
 <div>Hello World 1</div>
 <input type="text" >
 <button type="button">Click</button>
 <script src="dom3.js"></script>
</body>
</html>
```

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');

myInput.value = 'Laurence';

let counter = 0;
```

```
btn.onclick = ()=>{
    counter++;
    let boo = (counter < 3);
    if(counter < 2){
        output.style.color = 'red';
    }else if(counter < 4){
        output.style.color = 'green';
    }else if(counter < 6){
        output.style.color = 'purple';
    }else{
        output.style.color = 'blue';
    }
    output.textContent = `Counter : ${counter} ${boo}`;
    console.log(counter);
    updater();
}

let val = '';
function updater(){
    switch (counter) {
        case 0:
            val = `Case #1 ${counter}`;
            break;
        case 1:
            val = `Case #2 ${counter}`;
            break;
        case 3:
            val = `Case #3 ${counter}`;
            break;
        default:
            val = `DEFAULT ${counter}`;
    }
    output.innerText = val;
}
```

Operators in JavaScript How to apply operators

Lesson JavaScript Coding Exercise :

1. Get the value of the input field. Convert the value to a number using the Number() method.
2. In the condition check if the result of the input is an actual number of NaN using the isNaN() method
3. Using the modulus check if the value of the input is odd or even. If it has a remainder then it will be odd otherwise it is even.
4. Perform math calculations on the input value and output the results back to the user in the output field.

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
let a=val=b = 100;
//let val = 10 = a;

let c = a*b/100;
let d = 504 % 50;
a = a +1;
a++;
a--;
```

```
console.log(a);
console.log(d);
let e =10;
e *= a;
console.log(e);
console.clear();
console.log('5' == 5);
console.log('5' === 5);
console.log('5' != 6);
console.log('5' !== 5);
console.log( 10 <= 10);
console.log(('5' == 5) && true);
console.log(('5' == 5) && false);
console.log(('5' != 5) || true);
console.log(('5' == 5) || false);

btn.onclick = ()=>{
   let val = Number(myInput.value);
   console.log(typeof(val));
   console.log(isNaN(val));
   let html = `<div>Results ${val}</div>`;
   if(!isNaN(val)){
       html += `<div>Was a number</div>`;
   }else{
       html += `<div>NOT a number</div>`;
   }
   if(val % 2){
       html += `<div>Odd number</div>`;
```

```
    }else{
        html += `<div>Even number</div>`;
    }
    html += `<div>${val * 50} = ${val} X 50</div>`;
    html += `<div>${val / 10} = ${val} / 10</div>`;
    html += `<div>${val * val} = ${val} X ${val}</div>`;
    output.innerHTML = html;
}
```

Ternary Operator JavaScript Short one statement conditions

Lesson JavaScript Coding Exercise :

1. Get the value entered into the input field, check if it's a number. If it is not a number then in the condition update the output text with a message for the user. Change the style background and color of the output element.
2. If it is a number using the ternary operator create a output message to the user depending on the value that was entered into the input field
3. Check if the number entered is larger than 19 or equal to 19, and if it is then show a message that the person is allowed in. Otherwise deny entry for the person.

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
btn.onclick = clickedMe;
let val = (true) ? 'true' : 'false';
val = (true && false) ? 'true' : 'false';
val = (10 > 5) ? 'true' : 'false';
val = (isNaN('test')) ? 'true' : 'false';

output.textContent = 'How old are you?';
btn.textContent = 'Entry Checker';
```

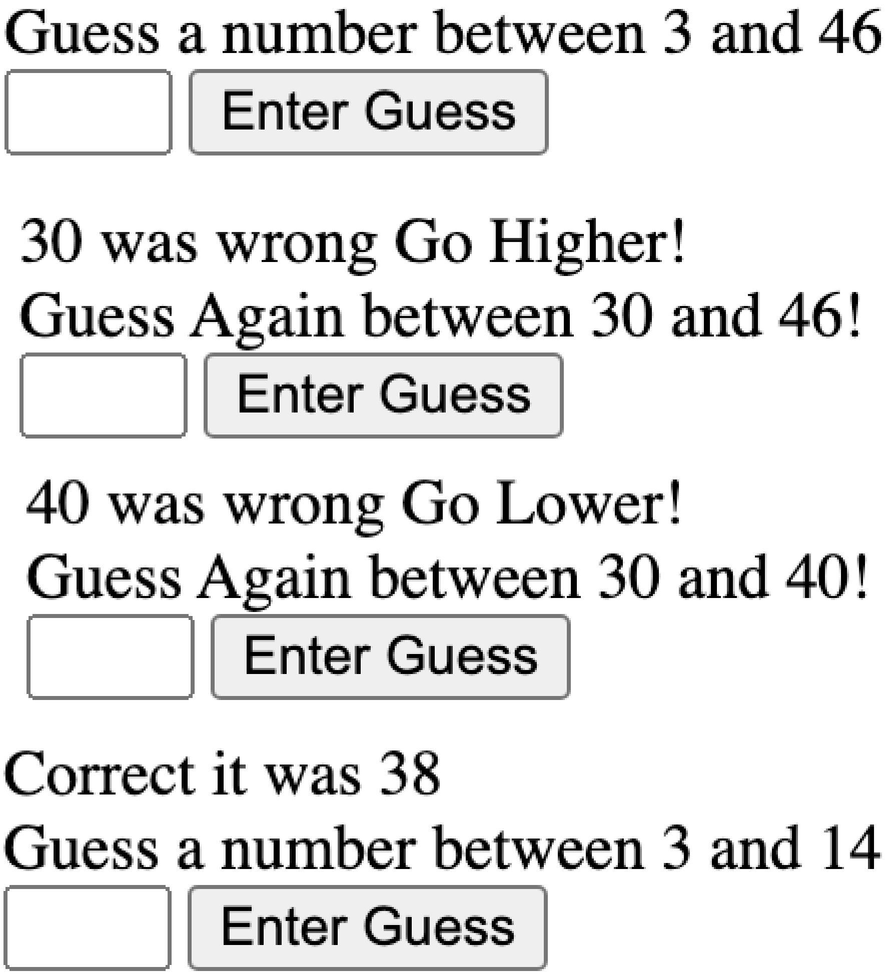
```
console.log(val);

function clickedMe(){
   const myAge = myInput.value;
   if(!isNaN(myAge)){
       console.log('ready');
       output.style.backgroundColor = 'white';
       output.style.color = 'black';
       output.textContent = 'Please enter  your Age?';
       const message = myAge >= 19 ? `${myAge} is allowed to enter.` : `${myAge} is not old enough!`;
       output.innerHTML += `<div>${message}</div>`;
       myInput.value = '';
   }else{
       myInput.value = '';
       output.style.backgroundColor = 'red';
       output.style.color = 'white';
       output.textContent = 'Please enter a number for your Age!';
   }
}
```

Math Random Values JavaScript get Random Numbers

JavaScript Math object contains various methods that can be used for math functionality, in addition it also contains the random method that creates random values in JavaScript. The Math.random() method returns a floating-point number in the range 0 (inclusive of 0) to less

than 1 (not including 1). The random value can then be multiplied and rounded to the nearest whole number to include the randomized range of values desired by the developer.
Random values are ideal for games and to create unique custom experiences for web users.

Random Number Interactive Guessing Game with JavaScript and the DOM coding exercise
In this exercise create a random number guessing game that will provide a random range of numbers that the user has to guess the correct number from. The application will provide the user feedback whether the guess was too high or too low, allowing for the user to narrow the range of the hidden number.   Once the solution is found matching the input value to the hidden number, the game starts again generating the random number and a new range to guess. Feedback is provided to the user in the HTML element, so that the user playing the game knows the results and what to do next. This is a perfect example of a simple game that can be created with JavaScript and interacting with the DOM page elements.



Guess a number between 3 and 46

30 was wrong Go Higher!
Guess Again between 30 and 46!

40 was wrong Go Lower!
Guess Again between 30 and 40!

Correct it was 38
Guess a number between 3 and 14

Lesson JavaScript Coding Exercise :

1. Select the HTML page elements as variables within your JavaScript code.
2. Create declare variables globally for a lowValue, highValue, and hiddenNumber
3. Create a function to start the game called starter().

4. Create a function to generate random numbers using min and max values are parameters within the function. Wrap the Math.random() within the Math.floor() to round down to the nearest whole number. Math.floor(Math.random()). Multiply the random value by the max minus the minimum, to include the max value add plus 1 to the multiplied number. Within the result add 1 to increment from 0 to the starting value of the minimum number.
5. Within the starter() function, generate a low value, a high value and a hidden number value with random numbers.
6. Update the HTML output div to provide instructions for the user to guess a number between the random range.
7. For the input element, change the type to number, and set attributes to min and max from the random values.
8. Add an event listener for onclick to the button page element, once clicked the button should invoke a function called clickedMe()
9. Within the clickerMe() function get the input value.
10. Check if the hidden number matches the value of the input value. If it does then output is correct and run the starter function again.
11. If the guess is incorrect, check whether the guess is lower or higher than the hidden number value. If it's lower, update the lowValue with the input value, and provide feedback to the user to guess higher. If the guess is high, provide feedback to the user, and update the high value with the input value.
12. The game should continue until the correct number is guessed, once the correct number is found then the game will reset and run again with new random numbers.

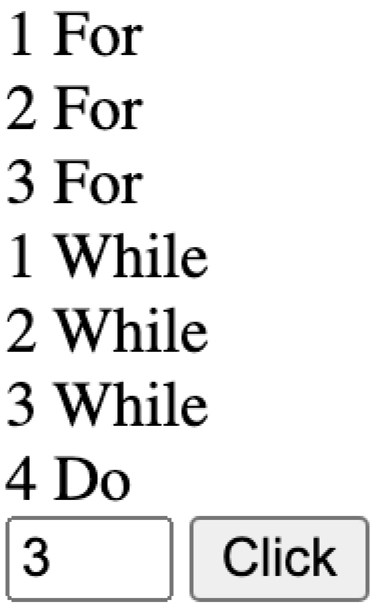
```
<!doctype html>
<html>
<head>
  <title>JavaScript</title>
</head>
<body>
 <div>Hello World 1</div>
 <input type="text" >
 <button type="button">Click</button>
 <script src="app.js"></script>
</body>
</html>
```

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
let lowValue = 1;
let highValue = 10;
let hiddenNumber = 0;
output.innerHTML = '';
starter();
function starter(){
   myInput.value = '';
   lowValue = getRan(0,5);
   highValue = getRan(lowValue +1,50);
   hiddenNumber = getRan(lowValue,highValue);
   output.innerHTML += `<div>Guess a number between ${lowValue}
and ${highValue}</div>`;
   btn.onclick = clickedMe;
   myInput.setAttribute('type','number');
   myInput.setAttribute('min',lowValue);
   myInput.setAttribute('max',highValue);
   btn.textContent = 'Enter Guess';
}
function clickedMe(){

   const valInput = myInput.value;
   if(valInput == hiddenNumber){
       console.log('correct');
       output.innerHTML = `<div>Correct it was ${
hiddenNumber}</div>`;
```

```
       starter();
   }else{
       //let message = (valInput < hiddenNumber) ? 'Go Higher!' : 'Go Lower';
       let message ;
       if(valInput < hiddenNumber){
           message = `${valInput} was wrong Go Higher!`;
           lowValue = valInput;
       }else{
           message = `${valInput} was wrong Go Lower!`;
           highValue = valInput;
       }
       output.innerHTML = `<div>${message}</div>`;
       console.log(hiddenNumber);
       myInput.value = '';
       output.innerHTML += `<div>Guess Again between ${lowValue} and ${highValue}!</div>`;
   }
  //let temp = Math.random()*10+1;

  //console.log(temp);
  //temp = Math.floor(temp);
  //output.textContent += `${temp}, `;
}

function getRan(min,max){
   return Math.floor(Math.random() * (max-min+1) + min);
}
```

JavaScript For, While Do While Loops Run blocks of code

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 set up 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.



```
1 For
2 For
3 For
1 While
2 While
3 While
4 Do
```

Lesson JavaScript Coding Exercise :

1. Select the page elements as JavaScript variables
2. Update the input field attributes to be a number input type, with a min and max value.
3. Add a click event listener to the button element
4. When the button is clicked, get the input value as a variable called num. Create a for loop to iterate a block of code the number of times from the input num value.
5. Create a while loop that will output html the number of times from the input value.
6. Create a do while loop which will create the html output the number of times from the input.
7. Update the html with the new content.

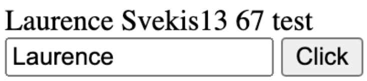
```
<!doctype html>

<html>

<head>
```

```
  <title>JavaScript</title>
</head>
<body>
 <div>JavaScript</div>
 <input type="text" >
 <button type="button">Click</button>
 <script src="dom7.js"></script>
</body>
</html>
```

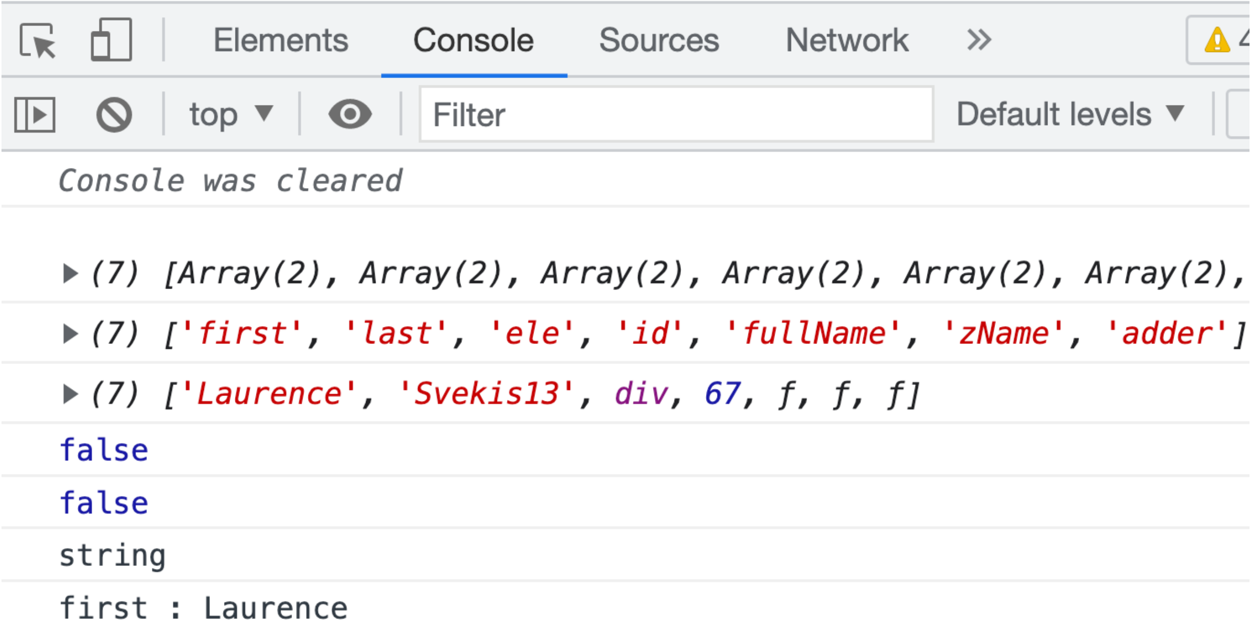
```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
myInput.setAttribute('type','number');
myInput.setAttribute('max',20);
myInput.setAttribute('min',0);
myInput.value = 5;
btn.onclick = btnClicked;

function btnClicked(){
   console.clear();
   let num = myInput.value;
   let html = '';
   for(let i=0;i<num;i++){
       console.log(i);
       html += `<div>${i+1} For</div>`;
   }
   let i = 10;
   while(i<num){
       i++;
       html += `<div>${i} While</div>`;
   }

   do{
       i++;
       html += `<div>${i} Do</div>`;
   }
   while(i<num);
   output.innerHTML = html;
   console.log('ready');
}
```

JavaScript Objects how to use Objects in code

One of JavaScript's data types is an object. These can be used to store various key value pair collections and even more complex entities. The document is a giant object that contains a lot of entities. Learning more about objects will help better define what can be done with the DOM entities, how they behave and why as well as how they can be used.



Laurence Svekis13 67 test



```
Console was cleared
▶ (7) [Array(2), Array(2), Array(2), Array(2), Array(2), Array(2),
▶ (7) ['first', 'last', 'ele', 'id', 'fullName', 'zName', 'adder']
▶ (7) ['Laurence', 'Svekis13', div, 67, f, f, f]
false
false
string
first : Laurence
```

Lesson JavaScript Coding Exercise :

1. Select the page elements from the HTML file as JavaScript objects.
2. Create a global object that has a value for first, last, id and create several methods in the object.
3. Create a method in the object that uses the first and last name values from the object and combines them into a string, including the id and any value the is sent to the function as an argument.
4. Create a method within the object that can update the objects first and last name. Generate a random value for the object id. Return the combined current object property values
5. When the button is clicked update the value of the object first property
6. Output on the page and in the console the new string invoking the methods for the fullName from the object
7. Get the object entries as nested array values and create a new array using the Object.entries()
8. Get and create an array with the object keys Object.keys()
9. Get and create an array with the object values Object.values()
10. Using the in or condition check if a key is contained within the object
11. Loop through the properties names from an object and output them in the console
12. Check to see the datatype of the property value, if it's a string or number output it into the console otherwise output a warning into the console.

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
btn.onclick = btnClicked;
myInput.value = 'Laurence';

const myObj = {
   first : 'Laurence',
   last : 'Svekis',
   ele : output,
   id : 0,
   fullName : function(val){
       return `${this.first} ${this.last} ${this.id} ${val}`
   },
   zName(val){
       return `${this.first} ${this.last} ${this.id} ${val}`
   },
   adder(first,last){
       this.first = first;
       this.last = last;
       this.id = Math.floor(Math.random()*100)
       return `${this.first} ${this.last} ${this.id}`;
   }
};

function btnClicked(){
   const newFirst = 'Laurence';
   const ranNum = Math.floor(Math.random()*100);
   const newLast = `Svekis${ranNum}`;
```

```
    console.log(myObj.adder(newFirst,newLast));
    output.innerHTML = `<div>${myObj.fullName('test')}</div>`;
    console.log(myObj.zName('test'));
    console.clear();
    const arr1 = Object.entries(myObj);
    console.log(arr1);
    const arr2 = Object.keys(myObj);
    console.log(arr2);
    const arr3 = Object.values(myObj);
    console.log(arr3);
    const key = myInput.value;
    console.log(myObj[key] !== undefined);
    console.log(key in myObj);
    for(const prop in myObj){
        const val = myObj[prop];
        console.log(typeof(val));
        if(typeof(val) == 'string' || typeof(val) == 'number'){
            console.log(`${prop} : ${myObj[prop]}`);
        }else{
            console.warn(`Other type ${typeof(val)}`)
        }
    }
    console.log(typeof(arr1));
}
```

Object Construction with JavaScript coding

You can use functions to construct objects, using the arguments within the function.

Lesson JavaScript Coding Exercise :

1. Get the value from the input field, using it to create a new object for a person. This should set the first, last and using both of those create a full name for the object key values.
2. Add to the function default values in case the arguments do not have values.

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
btn.onclick = btnClicked;
myInput.value = 'Laurence';

function FullName(first='Laurence',last='Svekis'){
   //first = first || 'Laurence';
   this.firstName = first,
   this.lastName = last,
   this.full = `${first} ${last}`;
}



function btnClicked(){
   const person = new FullName(myInput.value,'Smith');
   console.log(person.full);
}
```

Common JavaScript Array Methods to Update Array values

Arrays are also objects as a data type but they contain specific properties that allow the developer to interact with the contents of the array and its data.

arr.push(val); // add to array return the array length

arr.pop(); //remove last
arr.shift(); //remove first item
arr.unshift(val); //add to the front of array array length returned
arr.splice(1); // return array with all items after the index of 1
splice(start, deleteCount, val); //changes the contents of an array
slice(start, end); // returns a copy of a portion of an array into a new array
arr.slice(); //duplicate array as new array
arr.slice(5); // return array items from index 5
arr.slice(1,4); // return portion of array using slice
arr.toString(); // returns a string representation of the array items
arr.join(' - '); // returns a string representation of the array items using the argument value as a separator for the items in the array.

Lesson JavaScript Coding Exercise :

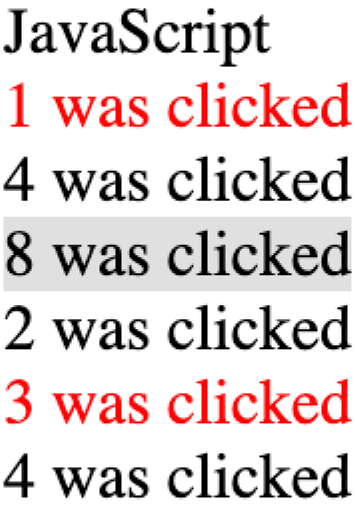
1. Setup HTML elements as JavaScript objects, add an onclick event to the button
2. Create 2 arrays, using concat join them to create a 3rd array, add one of the original arrays twice into the new array.
3. Try the different array methods push() pop() shift() and unshift() to update the data in the array.
4. Using splice() update the original array, create a new array from the array items that were spliced.
5. Using slice() create a new array from portions of the existing array
6. Convert an array into a string, also convert an array into a string using a dash as a separator.
7. Output the results into the console and on the web page.

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
btn.onclick = btnClicked;
myInput.value = 'Laurence';
const arr1 = ['One'];
const tempArr = ['onex','twox','threex'];
const arr = arr1.concat(tempArr,tempArr);



function btnClicked(){
   const val =  myInput.value;
   let temp = arr.push(val,'LAST'); // add to array return the array length
   console.log(temp);
   temp = arr.pop(); //remove last
   console.log(temp);
   temp = arr.shift(); //remove first item
   console.log(temp);
   temp = arr.unshift('First'); //add to the front of array array length returned
   console.log(temp);
   console.log(arr.length);
   arr[arr.length - 1] = 'LAST';
   //delete arr[0];
   //console.log(arr);
   arr.push('one1','two2','three3');
   //temp = arr.splice(1);  return array with all items after the index of 1
   //temp = arr.splice(1,3);
   temp = arr.splice(1,3,'Add1','Add2','Add3');
   console.log(temp);
   temp = arr.splice(3,0,'New1','New2','New3');
   console.log(temp);
   temp = arr.slice(); //duplicate array as new array
   temp[0] = 'TEST';
   temp = arr.slice(5); // return array items from index 5
   temp = arr.slice(1,4); // return portion of array using slice

   console.clear();
```

```
    console.log(temp);
    console.log(arr);
    const myStr = arr.toString();
    const myStr1 = arr.join(' - ');

    output.innerHTML = `<div>${myStr}</div>`;
    console.log(arr);
    console.log(myStr);
    console.log(myStr1);
}
```

Looping through Array contents forEach Methods and for loops for array data

Iterate through all the items in an array, selecting the items and their values. You can use the array length property in a for loop, or the array method of forEach(item,index,array) and return back the item values of the array.



```
JavaScript
1 was clicked
4 was clicked
8 was clicked
2 was clicked
3 was clicked
4 was clicked
```

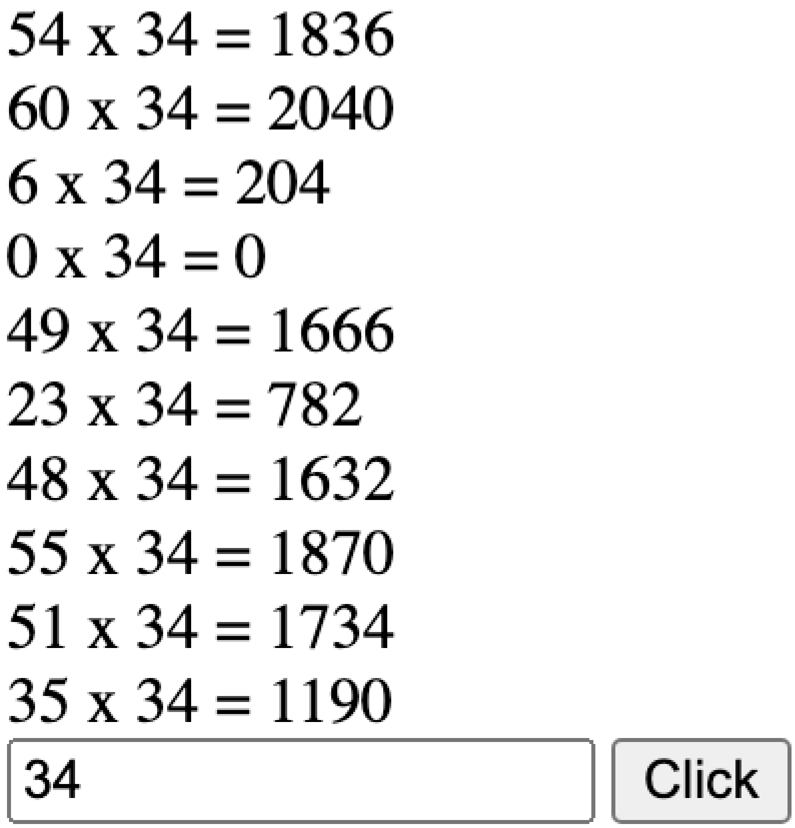
Lesson JavaScript Coding Exercise :

1. Select all the HTML elements into the JavaScript code.
2. Using a for loop update the output element with new divs
3. Using document.querySelectorAll() select all the matching results for divs within a div

4. Using forEach loop through all the new divs, add a new property value of val, with a value of 0 to each new div.
5. Add a click event to each new div, that will output the element, the index value and the event.target element into the console. Update the val value of the element increment by 1. Check if the style color is red, if not set to red otherwise set to black. Output the number of times each div was clicked into the element text.
6. On the main button click, add a loop through all the arr items, using the for loop and output the array item value into the console.
7. Using forEach get all the array items and output them as a div into the page, listing the index plus one and the item value in the new element.

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
btn.onclick = btnClicked;
myInput.value = 'Laurence';
const arr = [];
for(let i=0;i<10;i++){
   output.innerHTML += `<div>${i} Test Div</div>`;
}
const eles = document.querySelectorAll('div div');
console.log(eles);
eles.forEach((ele)=>{
   console.log(ele);
   ele.val = 0;
   ele.onclick = (e)=>{
       console.clear();
       console.log(ele);
       console.log(e.target);
       ele.val++;
       ele.innerText = `${ele.val} was clicked`;
       if(ele.style.color == 'red'){
           ele.style.color = 'black';
       }else{
           ele.style.color = 'red';
       }

   }
})



function btnClicked(){
   arr.push(myInput.value);
   console.log(arr.length);
   for(let i=0;i<arr.length;i++){
       console.log(`${i} ${arr[i]}`);
   }
   let html = '';
   arr.forEach((item,index,arr1)=>{
       html += `<div>${index+1}. ${item}</div>`;
       console.log(item);
       console.log(index);
       console.log(arr1);
       //console.clear();
       //arr1[0] = 'TEST';
   })
   output.innerHTML = html;
}
```

Array Methods for Items contained in the array JavaScript coding



```
54 x 34 = 1836
60 x 34 = 2040
6 x 34 = 204
0 x 34 = 0
49 x 34 = 1666
23 x 34 = 782
48 x 34 = 1632
55 x 34 = 1870
51 x 34 = 1734
35 x 34 = 1190
```

34

Lesson JavaScript Coding Exercise :

1. Create an array with strings
2. Create an array populated with random numeric values 0-100
3. Using the array map() method create a new array with the value of the string array, and its index value as the new items
4. Using the map() method create a new array with all the numeric values from the numeric array multiplied by 2
5. Using filter() return only the results that have a value of larger than 50 from the numeric array values
6. Using reduce() add all the value together as a total return value from the numeric array values
7. Using every() check to see if all the values are less than 150
8. Using some() check to see if any of the values are less than 50
9. Output the new arrays into the console to inspect them
10. Using map intake the value of the input field, then multiply that by the number value of the items in the numeric array.

11. Using forEach() loop through all the contents of the new array from the previous step, output the results into the html of the page.

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
btn.onclick = btnClicked;
myInput.value = 10;
myInput.setAttribute('type','number');
const arr = ['one','two','three','four','five'];
const temp1 = [];
for(let i=0;i<10;i++){
    const ranValue = Math.floor(Math.random()*100);
    temp1.push(ranValue);
}

function btnClicked(){
    const arr1 = arr.map((item,ind,array)=>{
        console.log(item);
        return `${item} ${ind} `;
    })
    const arr2 = temp1.map(val=> val*2);
    const arr3 = temp1.filter((val)=>{
        return val > 50;
    });
    const arr4 = temp1.reduce((previous,current,index)=>{
        console.log(previous);
        console.log(current);
        return previous + current;
    }); // total of all the items in the array

    const arr5 = temp1.every((val)=>{
        console.log(val);
        return val < 150;
    }) //boolean on the condition all have to be true
    const arr6 = temp1.some((val)=>{
        console.log(val);
        return val < 50;
    }) //boolean on the condition all have to be true
    console.log(arr1);
    console.log(arr2);
    console.log(arr3);
    console.log(arr4);
    console.log(arr5);
    console.log(arr6);
    const arr7 = temp1.map((val)=>{
        console.log(val);
        return val * myInput.value;
    })
    let html = '';
    arr7.forEach((ele,ind) => {
        html += `<div>${temp1[ind]} x ${myInput.value} = ${ele}</div>`;
    })
    output.innerHTML = html;
}
```

Array Methods for Sorting Array items using JavaScript Random Array

Get the random index value of the item from the array, and return the random item from the array.
How you can sort the arrays both in reverse and alphabetically.

Lesson JavaScript Coding Exercise :

1. Create an array with values, add to the array random numbers
2. When the button is clicked create a new array from the existing array filtering the data types with either numbers or strings
3. Using the array length, and math random() select a random index value for an array item.
4. When clicked, toggle either the sort() or reverse() methods to the array sorting in place.
5. Output the array as a string into the html of the page
6. Using sort() add a math random return of either positive or negative, which will randomize the array in place.
7. Wrap the randomizing of the array in a loop, testing 10 iterations to see the random array orders, output them to the html of the page
8. Using forEach() loop through all the array items and add them to the HTML

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
btn.onclick = btnClicked;
myInput.value = 10;
myInput.setAttribute('type','number');
const arr = ['one','two','three','four','five'];
let boo = true;
for(let i=0;i<10;i++){
   const ranValue = Math.floor(Math.random()*100);
   arr.push(ranValue);
}
console.log(arr);
function btnClicked(){
   const arr1 = arr.filter((ele)=>{
       return typeof(ele) == 'string';
   })
   const ind = Math.floor(Math.random()*arr.length);
   output.innerHTML = `Random Array Item with index of ${ind} value ${arr[ind]}`;
   let html = '';
   if(boo){
       arr.sort();
       boo = false;
   }else{
       arr.reverse();
   }
   html += `<div>${arr.toString()}</div>`;
   for(let i=0;i<10;i++){
       arr.sort(()=>{
           return Math.random() - 0.5;
       });
       html += `<div>${arr.toString()}</div>`;
   }
   arr.forEach((ele,ind)=>{
       html += `<div>${ind} - ${ele}</div>`;
   })
   output.innerHTML += html;
   console.log(arr1);
}
```

JavaScript Array Method Examples to find array items

JavaScript has several methods that can check for values contained within an array.

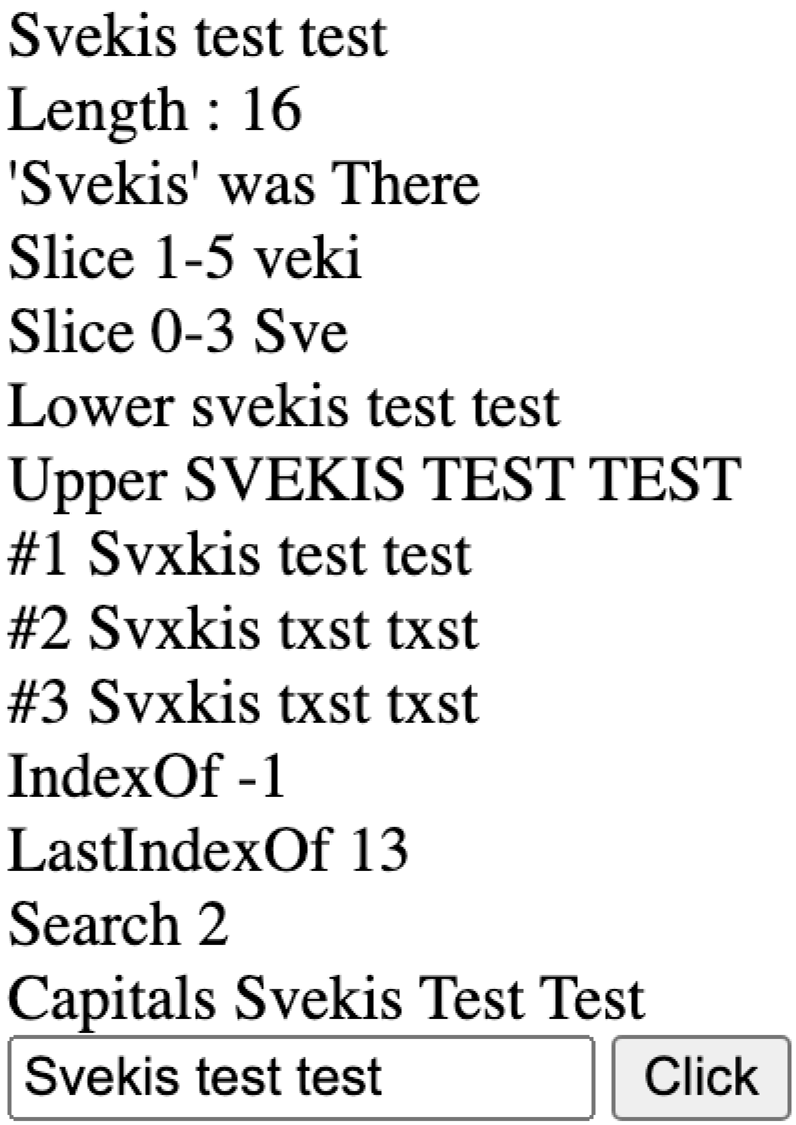
Lesson JavaScript Coding Exercise :

1. Create an array with some string and number values
2. Using the arr.includes() check if a value from the input field is in the array
3. Create a variable called html that can be built to return a string of the responses on the array methods. Using a ternary operator, create a response message if the term from the input is in the array or not.
4. Add to the html the response for includes
5. Create a check for the arr.indexOf() a value in the array, will return the index value or -1 if it's not found. Output the result to the HTML
6. Find the index value of the last item that matches, arr.lastIndexOf(), output the results into the HTML
7. Using find() return the first match for a number in the array, output the results to the HTML
8. Using find() return the first match for the condition checking for a == to the input value and the array item value. If none is found it will return undefined. Output the response result into the HTML
9. Using find() return the first matching result from the array of a value larger than 10. Notice it should also include and return the string.

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
btn.onclick = btnClicked;
const arr = ['Svekis','Laurence','100',50,10];

function btnClicked(){
   const val = myInput.value;
   const res = arr.includes(val);
   const message = (res) ? `was found` : `Not Found`;
   let html = `<h2>Search Results for ${val}</h2>`;
   html += `<div>${val} - ${message}</div>`;
```

```
    html += `<div>Includes ${res}</div>`;
    const indexPos = arr.indexOf(val);
    html += `<div>IndexOf ${indexPos}</div>`;
    const indexPosLast = arr.lastIndexOf(val);
    html += `<div>IndexLast ${indexPosLast}</div>`;
    const finderVal = arr.find((v)=>{
        return typeof(v) == 'number';
    });
    html += `<div>Find 1 ${finderVal}</div>`;
    const finderValP = arr.find((v)=>{
        return v == val;
    });
    html += `<div>Find 2 ${finderValP}</div>`;
    const finderValNum = arr.find((v)=>{
        return v > 10;
    });
    html += `<div>Find 3 ${finderValNum}</div>`;
    output.innerHTML = html;
    console.log(res);
    console.log(indexPos);
    console.log(indexPosLast);
    console.log(finderVal);
    console.log(finderValP);
    console.log(finderValNum);
}
```

JavaScript String Methods common functionality



```
Svekis test test
Length : 16
'Svekis' was There
Slice 1-5 veki
Slice 0-3 Sve
Lower svekis test test
Upper SVEKIS TEST TEST
#1 Svxkis test test
#2 Svxkis txst txst
#3 Svxkis txst txst
IndexOf -1
LastIndexOf 13
Search 2
Capitals Svekis Test Test
Svekis test test
```

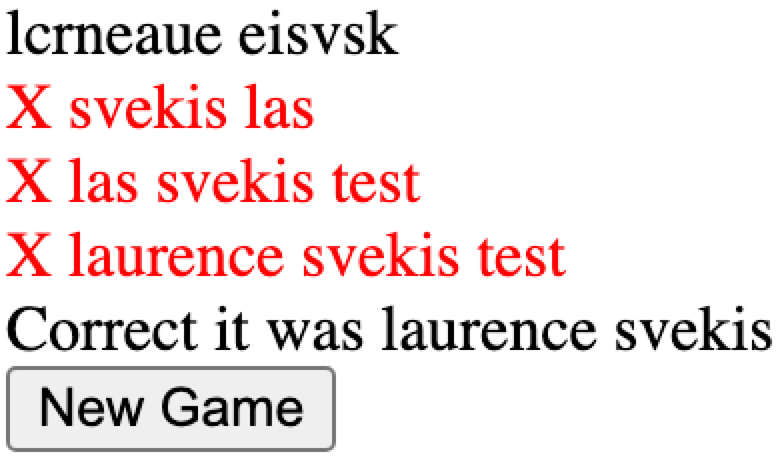
Lesson JavaScript Coding Exercise :

1. Use trim() to remove whitespace from the input field value
2. When The button is clicked output string method results into the html for the input value
3. Output the length of the string
4. Check if it includes a value return the results as a message to the user
5. Slice out parts of the string using the index values
6. Output the string as toLowerCase() and toUpperCase()
7. Using replace - replace all the e’s in the string with x’s replace('e','x'), replace(/e/g,'x'), replaceAll('e','x')
8. Get the indexOf a string value, also get the lastIndexOf() the same string value
9. Using search get the index of the string search('e')
10. Create a function that can capitalize words in the string.

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
btn.onclick = btnClicked;
myInput.value = 'Laurence Svekis';

function btnClicked(){
   const val = myInput.value.trim();
   let html = `<div>${val}</div>`;
   html += `<div>Length : ${val.length}</div>`;
   const key = 'Svekis';
   const inChecker = val.includes(key);
   const mes1 = inChecker ? ` was There` : ` was NOT`;
   html += `<div>'${key}' ${mes1}</div>`;
   html += `<div>Slice 1-5 ${val.slice(1,5)}</div>`;
   html += `<div>Slice 0-3 ${val.slice(0,3)}</div>`;
   html += `<div>Lower ${val.toLowerCase()}</div>`;
   html += `<div>Upper ${val.toUpperCase()}</div>`;
   const updateStr1 = val.replace('e','x');
   html += `<div>#1 ${updateStr1}</div>`;
   const updateStr2 = val.replace(/e/g,'x');
   html += `<div>#2 ${updateStr2}</div>`;
   const updateStr3 = val.replaceAll('e','x');
   html += `<div>#3 ${updateStr3}</div>`;
   const index1 = val.indexOf('re');
   html += `<div>IndexOf ${index1}</div>`;
   const index2 = val.lastIndexOf('e');
```

```
    html += `<div>LastIndexOf ${index2}</div>`;
    const sea1 = val.search('e');
    html += `<div>Search ${sea1}</div>`;
    html += `<div>Capitals ${makeCap(val)}</div>`;
    console.log(inChecker);
    output.innerHTML = html;
}

function makeCap(words){
    const arr = words.split(' ');
    const temp = [];
    arr.forEach((word)=>{
        const f = word.charAt(0).toUpperCase();
        const remainLetters = word.slice(1);
        console.log(f+remainLetters);
        temp.push(f+remainLetters);
    })
    return temp.join(' ');
}
```

Interactive Word Scramble Game with JavaScript shuffle letters in strings



lcrneaue eisvsk

X svekis las

X las svekis test

X laurence svekis test

Correct it was laurence svekis

Lesson JavaScript Coding Exercise :

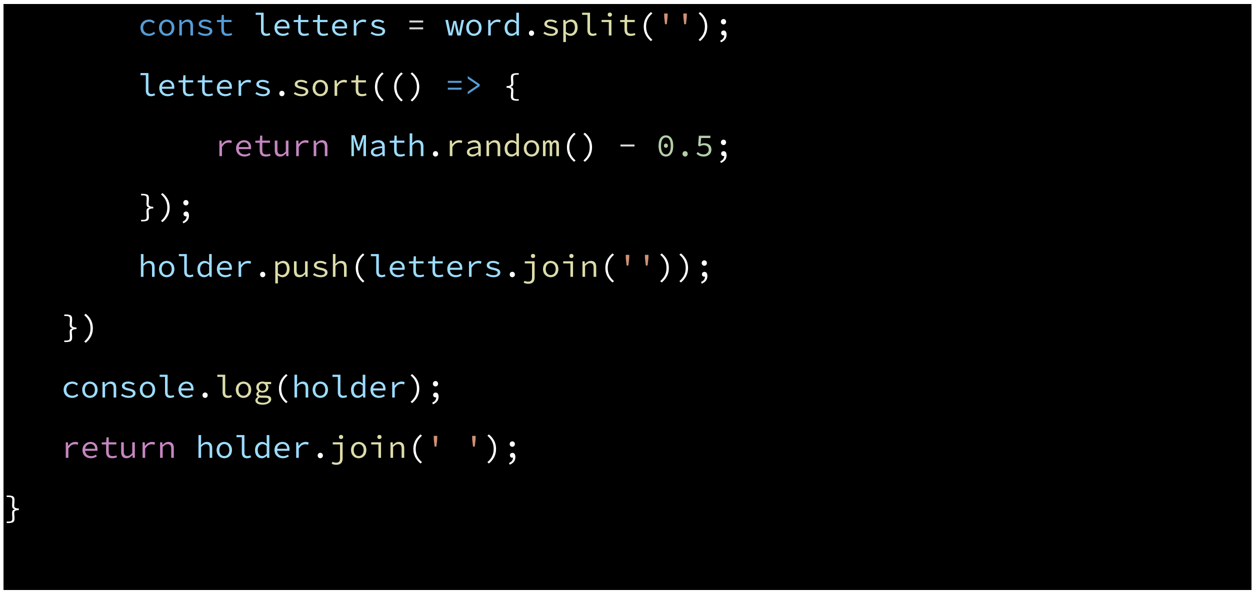
1. Create an array of words and phrases that you want to use for the game
2. Create a global game object that can hold the word that has been scrambled
3. create a function to start the game, called starter. Invoke the starter function.
4. Update the button text to say check answer
5. Update the input border and display properties to be solid black, and displayed as block
6. Clear the current value of the input field
7. Using Math random select a random item from the array of words
8. Create a function to scramble the words in the phrase called maker()
9. Within the maker() function convert the string into an array of each word. Create a holder array to use when you create the scrambled version of the words
10. Loop through all the words in the string, change the words to toLowerCase()
11. Split the letters of the word into a separate array. Using the sort() randomize the letters in the array. Using join() convert the array back into a string of randomly sorted letters from the word. Add it to the holding array
12. Return back the holder array, join the words back into a string.
13. Assign the scrambled words to the game's hidden value. Output the scrambled version of the phrase into the HTML page.
14. When the button is clicked check for the button text, if its new game then launch the starter() function to relaunch the game.
15. Convert the input to lowerCase. Compare the value with the hidden value of the selected string from the array. If there is a match the player solves it. Update the button text to the new game.
16. If it's incorrect, provide feedback to the player, asking them to try a new letter combination. It will continue to play until the correct solution is found.

```
<!doctype html>

<html>

<head>
```

```
  <title>JavaScript</title>
</head>
<body>
 <div>JavaScript</div>
 <input type="text" >
 <button type="button">Click</button>
 <script src="dom16.js"></script>
</body>
</html>
```

```
const output = document.querySelector('div');
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
btn.onclick = btnClicked;
const arr = ['JavaScript', 'Laurence Svekis', 'Html', 'code'];
const game = {
   hidden: ''
};
starter();

function btnClicked() {
   if (btn.textContent == 'New Game') {
       starter();
   } else {
       const checkVal = myInput.value.toLowerCase();
       if (checkVal == game.hidden) {
           console.log('correct');
           output.innerHTML += `<div>Correct it was
${checkVal}</div>`;
```

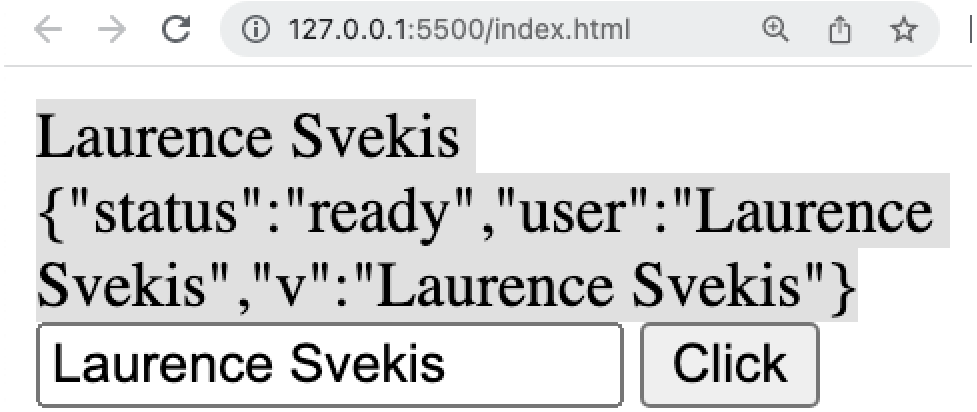
```
            btn.textContent = 'New Game';
            myInput.style.border = 'black solid 1px';
            myInput.style.display = 'none';
        } else {
            console.log('wrong');
            myInput.style.border = 'red solid 1px';
            output.innerHTML += `<div style="color:red">X
${checkVal}</div>`;
        }
    }
}

function starter() {
    btn.textContent = 'Check Answer';
    myInput.style.border = 'black solid 1px';
    myInput.style.display = 'block';
    myInput.value = '';
    const ind = Math.floor(Math.random() * arr.length);
    game.hidden = arr[ind].toLowerCase();
    const rep = maker(game.hidden);
    output.innerHTML = `<div>${rep}</div>`;
}

function maker(words) {
    const tempArr = words.split(' ');
    const holder = [];
    tempArr.forEach(word => {
        word = word.toLowerCase();
```



```
        const letters = word.split('');
        letters.sort(() => {
            return Math.random() - 0.5;
        });
        holder.push(letters.join(''));
    })
    console.log(holder);
    return holder.join(' ');
}
```

LocalStorage with JavaScript JSON parse and Stringify of Objects

Localstorage provides a way to store values as strings into the browser, that can be retrieved using JavaScript the next time the user visits the webpage.



Laurence Svekis
{"status":"ready","user":"Laurence Svekis","v":"Laurence Svekis"}

Laurence Svekis

Lesson JavaScript Coding Exercise :

1. Try converting an array into a string, then parse it back to a usable array from the string. Do the same for the object. JSON.stringify(arr) JSON.parse(strArr)
2. When the HTML loads invoke a function that checks the local storage for a value using a key. If a value is returned then parse the string into an object. Output the contents of the object into the page.
3. Once the button is clicked, create an object that gets converted to a string, into localstorage.

```
const myInput = document.querySelector('input');
const btn = document.querySelector('button');
const output = document.querySelector('div');

btn.onclick = btnClicked;

const arr = [1,2,3,4,5,6];
const strArr = JSON.stringify(arr);
console.log(strArr);
const obj = {
   first : 'Laurnce',
   last : 'Svekis'
};
const strObj = JSON.stringify(obj);
console.log(strObj);

const arr1 = JSON.parse(strArr);
console.log(arr1);

const obj1 = JSON.parse(strObj);
console.log(obj1);
console.clear();
const holder = {status:'ready'};
init();

function init(){
   const val = localStorage.getItem('val');
```

```
    console.log(val);
    const temp = JSON.parse(val);
    console.log(temp);
    if(temp != null){
        console.log('value');
        output.textContent = temp.user;
        output.innerHTML += `<div>${JSON.stringify(temp)}</div>`;
        myInput.value = temp.v;
    }else{
        output.textContent = 'Not found';
    }
}

function btnClicked() {
    const valInput = myInput.value;
    holder.user = valInput;
    holder.v = valInput;
    const temp = JSON.stringify(holder);
    localStorage.setItem('val',temp);
    output.textContent = valInput;
    output.innerHTML += `<div>${JSON.stringify(holder)}</div>`;
}
```