Intro to JavaScript Week 3 Coding Assignment

**Points possible:** 70

|  |  |  |
| --- | --- | --- |
| Category | Criteria | % of Grade |
| Functionality | Does the code work? | 25 |
| Organization | Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear. | 25 |
| Creativity | Student solved the problems presented in the assignment using creativity and out of the box thinking. | 25 |
| Completeness | All requirements of the assignment are complete. | 25 |

**Instructions:** In VS Code, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week’s assignments and push this document, with your JavaScript project code, to the repository. Add the URL for this week’s repository to this document where instructed and submit this document to your instructor when complete.

**Coding Steps:**

1. Create an array called ages that contains the following values: 3, 9, 23, 64, 2, 8, 28, 93.
   1. Programmatically subtract the value of the first element in the array from the value in the last element of the array (do not use numbers to reference the last element, find it programmatically, ages[7] – ages[0] is not allowed). Print the result to the console.
   2. Add a new age to your array and repeat the step above to ensure it is dynamic (works for arrays of different lengths).
   3. Use a loop to iterate through the array and calculate the average age. Print the result to the console.
2. Create an array called names that contains the following values: ‘Sam’, ‘Tommy’, ‘Tim’, ‘Sally’, ‘Buck’, ‘Bob’.
   1. Use a loop to iterate through the array and calculate the average number of letters per name. Print the result to the console.
   2. Use a loop to iterate through the array again and concatenate all the names together, separated by spaces, and print the result to the console.
3. How do you access the last element of any array?
   1. Programmatically using array[array.length – 1];
   2. Array.pop will return the last element and remove it from the array
   3. Array.push(“Tim”) will insert the string “Tim” as the new ultimate element of the array
4. How do you access the first element of any array?
   1. Directly using array[0];
   2. Programmatically using array[array.length – array.length];
5. Create a new array called nameLengths. Write a loop to iterate over the previously created names array and add the length of each name to the nameLengths array.
6. Write a loop to iterate over the nameLengths array and calculate the sum of all the elements in the array. Print the result to the console.
7. Write a function that takes two parameters, word and n, as arguments and returns the word concatenated to itself n number of times. (i.e. if I pass in ‘Hello’ and 3, I would expect the function to return ‘HelloHelloHello’).
8. Write a function that takes two parameters, firstName and lastName, and returns a full name (the full name should be the first and the last name separated by a space).
9. Write a function that takes an array of numbers and returns true if the sum of all the numbers in the array is greater than 100.
10. Write a function that takes an array of numbers and returns the average of all the elements in the array.
11. Write a function that takes two arrays of numbers and returns true if the average of the elements in the first array is greater than the average of the elements in the second array.
12. Write a function called willBuyDrink that takes a boolean isHotOutside, and a number moneyInPocket, and returns true if it is hot outside and if moneyInPocket is greater than 10.50.
13. Create a function of your own that solves a problem. In comments, write what the function does and why you created it.

**Screenshots of Code:**

//index.js

//1

var ages = [3,9,23,64,2,8,28,93];

function fAgeDifference(params){

    return "The difference between the first and last elements is:  " + (params[params.length - params.length] - params[params.length - 1]);

}

// let ageDifference = ages[ages.length - ages.length] - ages[ages.length - 1];

// console.log(ages);

// console.log(ageDifference);

console.log("1a. " + fAgeDifference(ages));

ages.push(57);

// ageDifference = ages[ages.length - ages.length] - ages[ages.length - 1];

// console.log(ages);

// console.log(ageDifference);

console.log("1b. " + fAgeDifference(ages));

var sumAges = 0,avgAges = 0

for(let i = 0;i < ages.length;i++){

    sumAges += ages[i];

}

avgAges = sumAges / ages.length;

console.log("1c.  The average of the array elements is:  " + avgAges);

//2

var names = ["Sam", "Tommy", "Tim", "Sally", "Buck", "Bob"];

var lengthNames = 0;

for(let i = 0;i < names.length;i++){

    lengthNames += names[i].length;

}

console.log("2a.  The average element length is:  " + lengthNames/names.length);

console.log("2a.-.join.length method string:  " + names.join("").length / names.length);

var allNames = "";

for(name of names){

    allNames = allNames + name + " ";

}

console.log("2b.-concatenation.  The list of concatenated names is:  " + allNames);

console.log("2b.-.join method.  The list of concatenated names is:  " + names.join(" "));

// allNames = "";

// for(name of names){

//     allNames = allNames + names.pop() + " ";

// }

// console.log("2b.  The list of concatenated names is:  " + allNames);

//5.  Add element lengths of the names array to nameLengths array

var nameLentghs = [];

let nameLengths = names.map(function(element){

    return element.length;

})

console.log("5.  The nameLengths array is:  " + nameLengths);

//6. Sum array element's lengths

let summedArray = nameLengths.reduce(function(summedItems,item){

    return summedItems + item;

});

console.log("6.  The summed array numbers add up to:  " + summedArray)

//7.  Prompt for a word and a number and concatenate word that number of times

function copiedWord(word  = prompt("Please enter any word."),n = prompt("Please enter any number.")){

    let concatenatedWord = "";

    let enteredParameters = [];

    for(let i = 1;i <= n;i++){

        concatenatedWord = concatenatedWord + word;

    }

    enteredParameters.push(word);

    enteredParameters.push(n);

    enteredParameters.push(concatenatedWord);

    return enteredParameters;

}

let returnedParameters = copiedWord(); //using back ticks "`" for templating

console.log(`7.  Here's the word ${returnedParameters[0]} concatenated ${returnedParameters[1]} times:  ${returnedParameters[2]}`);

//8.  Create function prompting for firstName and lastName and return the full name

function fullName(firstName = prompt("Please enter the first name:"),lastName = prompt("Please enter the last name:")){

    return firstName + " " + lastName;

}

console.log("8.  The full name is:  " + fullName());

//9.  Create function to take array of numbers and return true if summed array is > 100

function greaterThan100(passedArray){

    var summedArray = 0;

    for(var i = 0;i < passedArray.length;i++){

        summedArray += passedArray[i];

    }

    if(summedArray > 100){

        return console.log("9.  True");

    }

    else{

        return console.log("9.  False");

    };

};

console.log(greaterThan100([10,20,30,40]));

console.log(greaterThan100([10,20,30,40,50,60]));

console.log(greaterThan100.length);

//10.  Create function to take and array of numbers and return their averag

function returnAverage(passedArray){

    var summedArray = 0;

    for(let i = 0;i < passedArray.length;i++){

        summedArray = summedArray + passedArray[i];

    }

return summedArray / passedArray.length;

};

console.log("10. The average of the numbers in the array is:  " + returnAverage([10,20,30,40,50]));

//11.  Create function taking two arrays and compating their averages, return true if

//      average of first array is greater than average of second

function compareArrays(passedArray1,passedArray2){

    if(returnAverage(passedArray1) > returnAverage(passedArray2)){

        return true;

    }else{

        return false;

    }

}

console.log(`11.  The first array's average; ${returnAverage([10,20,30,40,50])} \

, is greater than the second array's average; ${returnAverage([11,22,33,44,55])}:  `

+ compareArrays([10,20,30,40,50],[11,22,33,44,55]))

//12.  Create function to take two parameters, boolean isHotOUtside, and number moneyInPocket

//      returns true if it is hot out side and money in pocket is greater than 10.50

function willBuyDrink(isHotOutside,moneyInPocket){

    if(isHotOutside === true && moneyInPocket > 10.50){

        return true;

    }else{

        return false;

    }

}

console.log("12.  I will buy a drink, true or false?  " + willBuyDrink(true,10.51));

//13.  Create a function of my own design to solve a problem

//A function to tell you the week day given the date in "MM/DD/YYYY" format

//Hopefully demonstrating closure too.  As soon as the function is complete

//

function getTheWeekDay(date) {

    let theDate = new Date(date);

    //var theFormattedDate = `13.  ${theDate.getMonth()+1}/${theDate.getDate()}/${theDate.getFullYear()} is a ${returnsTheWeekDay()}`;

    console.log(`13.  ${theDate.getMonth()+1}/${theDate.getDate()}/${theDate.getFullYear()} is a ${returnsTheWeekDay()}.`);

    function returnsTheWeekDay(){ //Here's the enclosed function

        if (Object.prototype.toString.call(theDate) === "[object Date]"){

            switch (theDate.getUTCDay()){

                case 0:

                    return 'Sunday';

                case 1:

                    return "Monday";

                case 2:

                    return "Tuesday";

                case 3:

                    return "Wednesday";

                case 4:

                    return "Thursday";

                case 5:

                    return "Friday";

                case 6:

                    return "Saturday";

            }

        }

    }

}

getTheWeekDay("12/12/2020");

//returnsTheWeekDay("10/10/2020"); //is not defined in the global scope therefor closure occurred

//quiz

//7

let groceryItems = ["lettuce","tomatoe","plums"]

let allItems = "";

for(item of groceryItems){

    console.log(item);

}

//8

function baseExponent(base,exponent){

    return Math.pow(base,exponent);

}

console.log(baseExponent(2,3));

//6

console.log(names.length);

//9 Write a function that takes an array of customer objects and returns a new array containing only the customers who have purchased more than 5 items.

//Below is an example of one customer object. The properties of all customer objects will have the same names as this one.

let customer = [

    {

    name: "Tom",

    itemsPurchased: [

        "Apples",

        "Grapes",

        "Milk",

        "Toothbrush",

        "Water",

        "Chocolate"

    ]

    },

    {

    name: "John",

    itemsPurchased: [

        "Apples",

        "Grapes",

        "Milk"

    ]

    },

    {

    name: "Mary",

    itemsPurchased: [

        "Apples",

        "Grapes",

        "Milk",

        "Eggs",

        "Sandwich",

        "Bread"

    ]

    },

    {

    name: "Bertha",

    itemsPurchased: [

        "Apples",

        "Grapes",

        "Milk",

        "Eggs",

        "Sandwich",

        "Bread"

    ]

    },

    {

    name: "Pratt",

    itemsPurchased: [

        "Apples",

        "Grapes",

        "Milk",

        "Eggs",

        "Sandwich",

        "Bread"

    ]

    },

]

//Here customer.length is used to iterate across the array of customer(s) rather than customer.itemsPurchased.length

//

function buildBigBuyers(){

let bigBuyers = [];

    for (let i = 0;i < customer.length;i++){

        if (customer[i].itemsPurchased.length > 5){

            bigBuyers.push(customer[i]);

            console.log(bigBuyers.name);

        }

    }

}

buildBigBuyers();

//"item" in this case, is the same as "customer[i]" in buildBigBuyers above

function buildBigBuyers2(){

let bigBuyers = [];

    customer.forEach(function(item){

        if (item.itemsPurchased.length > 5){

            bigBuyers.push(item);

                console.log(bigBuyers);

        }

    })

}

buildBigBuyers2();

//Lexically (statically) scoped closure

let x2 = 10

function f(){

    console.log(`Function f()'s x2 is:  ${x2}`);

    return x2;

}

function g(){

    let x2 = 20;

    console.log(`Function g()'s x2 is:  ${x2}`);

    return f();

}

//Set breakpoints at all three console.logs and note the variable x2

//as it is from the beginning and as it comes and goes as g() is invoked

console.log(`This is x2: ${g()}`);

// var bigBuyers = customer.filter()

//10 Write a function that takes two arguments, an array of strings and a string. The function should return the index in the array that the string is located at.

// If the string is not found in the array return -1.

function findStringInArray(arrayOfStrings = [],stringToFind = ""){

    return arrayOfStrings.indexOf(stringToFind)

}

console.log(findStringInArray(names, "Tim"));

//Closures - Brought to you by lexical scoping - Used for variable and function privacy and security.

// Each and every function has closure in that any

// functions or variables declared inside a function are not visible to outer functions.  The

// reverse is not true; outer functions and variables are available to inner functions.  To

// retrieve inner functions or variables, a function's return statement must be used to pass

// the variable or function up the lexical stack to outer functions or variables.

function makeFunc() {

    var name = "Some Name";

    function displayName() {

        console.log(name);

    }

    return displayName;

}

var myFunc = makeFunc();

myFunc();

//Scope chain demo

const bestAvenger = 'Iron man';

var bestProgrammingLanguage = '';

function a () {

  const bestActor = "Neymar";

  console.log(bestAvenger); // output:Iron man

  function c() {

    const bestProgrammingLanguage = 'Html';

    console.log(bestActor); // output:Neymar

    b();

  }

  c();

}

function b() {

  console.log(bestProgrammingLanguage); // not defined error

}

a();

//Scope chain demo

// const myNumber = '3';

// (function (callback) {

//   console.log("Inside callback function " + myNumber);

//   const myText = 'hello';

//   callback();

// })(function () {

//     console.log("Inside anonymous function " + myNumber);

//     console.log("Inside anonymous function " + myText);

//   })

// function printProps(o){

//     for(var p in o)

//         console.log(p + ": " + o[p] + "\n");

// }

// printProps(console);

// function myFunction(){

//     for(let i = 0; i < 100;i++){

//         console.log(i);

//     }

// }

// myFunction();

// function createFullName(firstName,lastName){

//     console.log("This is the full name entered:  " + firstName + " " + lastName);

//     return firstName + " " + lastName;

// }

// var fullName = createFullName("Tim","Gibney");

// console.log("Welcome, " + fullName);

var x = [ 'p0', 'p1', 'p2' ];

call\_me(x);

function call\_me(params) {

  for (i=0; i<params.length; i++) {

    console.log(params[i])

  }

}

var x = 1;

for(i=0;i<3;i++){

    console.log(x+=5\*i);

}

hello = () => console.log("Hello");

hello();

/\*

Variable and function hoisting - Javascript allows variables and functions to be called within code when the actual

variable or function declaration is listed after the initial variable or function call; that is to say, when a .js

file is parsed, javascript moves (hoists) the variable or function to the top of its scope in preparation for use.  WITHIN

ITS SCOPE!  This makes for some caveats:  Variables and functions created within functions are NOT hoisted to their outer

function's scope.  (Scoping requires that all objects remain available only within their scope.  To pass scoped variables or

functions to outer/upper scope, a return statment must be used or a global variable must be set to the scoped variable or

function.)  When variables are hoisted, they are hoisted and initialed as "undefined", even if, when initially declared, the

variable is assigned an object.

\*/

// function getCircumference(radius) {

//     console.log(circumference)

//     circumference = PI\*radius\*2;

//     const PI = 22/7;

//   }

//   getCircumference(2); // ReferenceError: circumference is not defined

// var getCircumference2(radius) = () => PI\*radius\*2;

// circumference = PI\*radius\*2;

// const PI = 22/7;

// getCircumference2(2);

**Screenshots of Running Application:**

1a. The difference between the first and last elements is: -90 [index.js:12:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

1b. The difference between the first and last elements is: -54 [index.js:19:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

1c. The average of the array elements is: 31.88888888888889 [index.js:26:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

2a. The average element length is: 3.8333333333333335 [index.js:34:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

2a.-.join.length method string: 3.8333333333333335 [index.js:35:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

2b.-concatenation. The list of concatenated names is: Sam Tommy Tim Sally Buck Bob [index.js:41:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

2b.-.join method. The list of concatenated names is: Sam Tommy Tim Sally Buck Bob [index.js:43:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

5. The nameLengths array is: 3,5,3,5,4,3 [index.js:57:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

6. The summed array numbers add up to: 23 [index.js:64:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Layout was forced before the page was fully loaded. If stylesheets are not yet loaded this may cause a flash of unstyled content. [index.js:69:28](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

7. Here's the word Theatre concatenated 3 times: TheatreTheatreTheatre [index.js:82:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

8. The full name is: Timothy Gibney [index.js:90:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

9. False [index.js:104:24](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

undefined [index.js:108:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

9. True [index.js:101:24](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

undefined [index.js:109:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

1 [index.js:110:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

10. The average of the numbers in the array is: 30 [index.js:120:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

11. The first array's average; 30 , is greater than the second array's average; 33: false [index.js:133:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

12. I will buy a drink, true or false? true [index.js:148:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

13. 12/12/2020 is a Saturday. [index.js:159:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

lettuce [index.js:192:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

tomatoe [index.js:192:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

plums [index.js:192:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

8 [index.js:200:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

6 [index.js:205:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

undefined 4 [index.js:272:21](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Array [ {…} ]

[index.js:285:25](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Array [ {…}, {…} ]

[index.js:285:25](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Array(3) [ {…}, {…}, {…} ]

[index.js:285:25](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Array(4) [ {…}, {…}, {…}, {…} ]

[index.js:285:25](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Function g()'s x2 is: 20 [index.js:301:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Function f()'s x2 is: 10 [index.js:295:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

This is x2: 10 [index.js:307:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

2 [index.js:317:9](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Some Name [index.js:330:17](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Iron man [index.js:344:11](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Neymar [index.js:347:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

<empty string> [index.js:353:11](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

p0 [index.js:394:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

p1 [index.js:394:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

p2 [index.js:394:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

1 [index.js:400:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

6 [index.js:400:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

16 [index.js:400:13](file:///c:\Users\Tim\Documents\NashuaCC\FESD-Week3\index.js)

Hello

**URL to GitHub Repository:**

[**https://github.com/mctimoth/FESD-Week3**](https://github.com/mctimoth/FESD-Week3)