Intro to JavaScript Week 4 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. Using template literals instead of concatenation, write a function that takes firstName and lastName and returns ‘fistName lastName’
2. Write the same function as above as an arrow function with a different name.
3. Look up the JavaScript functions setTimeout() and setInterval(). Notice how they each take a callback.
   1. Using setTimeout, write an inline, anonymous (has no named assigned to it) arrow function in the callback parameter position. The function should alert ‘Time is up!’. Choose whatever length of time you want for the timeout.
   2. Write an arrow function named askAreWeThereYet that alerts ‘Are we there yet?’. Using setInterval, pass askAreWeThereYet into the callback parameter position. Choose whatever length of time you want for the interval.
4. In this step you are going to write a function that takes a callback to better understand how callbacks work.
   1. Write a function named processSplicedValue that takes 3 parameters – an array, the index of the element to be spliced from the array, and a callback that will process the sliced element.
   2. Inside the function, use the first two parameters to splice an element from the array.
   3. Call the callback function and pass the spliced value into it.
   4. Outside of your function, create an array of strings, call processSplicedValue, and pass the array, an index number, and console.log into it. For example: processSplicedValue(arrayName, 2, console.log);
   5. Call the processSplicedValue function again but this time pass in the alert method instead of console.log.
   6. Call the processSplicedValue function again, but this time pass in an anonymous arrow function that alerts the spliced value.
   7. Call the processSplicedValue function one more time, but this time, pass in a custom function of your choice that you should create and name.

**Screenshots of Code:**

//index.js for FESD Javascript Week4

//1.  Use template literals to combine variables

function returnFullName(firstName, lastName){

    console.log(`1.  ${firstName} ${lastName}`);

}

returnFullName("Tim","Gibney");

//2.  Now make 1. above into an "arrow function"

const theFullNameIs = (firstName,lastName) => console.log(`2.  ${firstName} ${lastName}`);

theFullNameIs("Tim","Gibney");

//3a.  An anonymous function using the setTimeout function and alert function.

// The function waits for allotted number of milliseconds (1000 millisconds = 1 second)

// then alerts the window.  Two options follow:  An immediately invoked function execution

// and a command invoked anonymous arrow function.

// (function() {setTimeout(function() {alert("IIFE Time is up!")},2000)})();

const alertUser = duration => setTimeout(function(/\*duration\*/) {alert("3a.  Time is up!")},duration);

alertUser(5000);

//theFullNameIs("John","Gibney");//Having this function here displays that the above setTimeout functions

                                // do indeed not pause processing.  But so will the other upcoming

                                // assignment answers.

//3b.  Write an arrow function that uses the alert functions to display an alert.

const askAreWeThereYet = () => {alert("3b.  Are we there yet?")};

//Then invoke askAreWeThereYet using setInterval to repeatedly display the alert.

//Trouble is, this IIFE will run incessantly without invoking clearInterval().

var intervalID = setInterval(askAreWeThereYet,2000);

// clearInterval(intervalID);

//4a-c.  Array processing function takes someArray and splices at theIndex then

//  uses a callback to output someArray

function processSplicedValue(someArray,theIndex,callback){

    if (callback && typeof(callback) === "function") {

    someArray.splice(theIndex);

    callback(someArray);

    }else{

        console.log(`"${callback}" is not a function.`);

    }

}

//4d.

let arrayOfStrings = ["this","that","the","other"];

// processSplicedValue(arrayOfStrings,2);

let arrayOfStrings2 = ["4d.","this","that","the","other"];

processSplicedValue(arrayOfStrings2,3,console.log);

//4e.

let arrayOfStrings3 = ["4e.","this","that","the","other"];

processSplicedValue(arrayOfStrings3,3,alert);

// let arrayOfStrings4 = ["this","that","the","other"];

// processSplicedValue(arrayOfStrings4,2,2);

//4f.

let arrayOfStrings4 = ["4f.","this","that","the","other"];

processSplicedValue(arrayOfStrings4,3,function(){alert(arrayOfStrings4)});

//4g.

let arrayOfStrings5 = ["this","that","the","other"];

processSplicedValue(arrayOfStrings5,3, function addQuestionPrefix() {

        console.log("4g. " + arrayOfStrings5);

        }

    );

// var users = ["Sam","Ellie","Bernie"];

// function addUser(userName){

//     setTimeout(function(){

//         users.push(userName);

//     },101);

// }

// function getUsers(){

//     setTimeout(function(){

//         console.log(users);

//     },100);

// }

// addUser("Jake");

// getUsers();

// function addUser2(userName,callback){

//     setTimeout(function(){

//         users.push(userName);

//         callback();

//     },1000);

// }

// function getUsers2(){

//     setTimeout(function(){

//         console.log(users);

//     },100);

// }

// addUser2("Jake",getUsers2);

**Screenshots of Running Application:**

//index.js for FESD Javascript Week4

//1.  Use template literals to combine variables

function returnFullName(firstName, lastName){

    console.log(`1.  ${firstName} ${lastName}`);

}

returnFullName("Tim","Gibney");

//2.  Now make 1. above into an "arrow function"

const theFullNameIs = (firstName,lastName) => console.log(`2.  ${firstName} ${lastName}`);

theFullNameIs("Tim","Gibney");

//3a.  An anonymous function using the setTimeout function and alert function.

// The function waits for allotted number of milliseconds (1000 millisconds = 1 second)

// then alerts the window.  Two options follow:  An immediately invoked function execution

// and a command invoked anonymous arrow function.

// (function() {setTimeout(function() {alert("IIFE Time is up!")},2000)})();

const alertUser = duration => setTimeout(function(/\*duration\*/) {alert("3a.  Time is up!")},duration);

alertUser(5000);

//theFullNameIs("John","Gibney");//Having this function here displays that the above setTimeout functions

                                // do indeed not pause processing.  But so will the other upcoming

                                // assignment answers.

//3b.  Write an arrow function that uses the alert functions to display an alert.

const askAreWeThereYet = () => {alert("3b.  Are we there yet?")};

//Then invoke askAreWeThereYet using setInterval to repeatedly display the alert.

//Trouble is, this IIFE will run incessantly without invoking clearInterval().

var intervalID = setInterval(askAreWeThereYet,2000);

// clearInterval(intervalID);

//4a-c.  Array processing function takes someArray and splices at theIndex then

//  uses a callback to output someArray

function processSplicedValue(someArray,theIndex,callback){

    if (callback && typeof(callback) === "function") {

    someArray.splice(theIndex);

    callback(someArray);

    }else{

        console.log(`"${callback}" is not a function.`);

    }

}

//4d.

let arrayOfStrings = ["this","that","the","other"];

// processSplicedValue(arrayOfStrings,2);

let arrayOfStrings2 = ["4d.","this","that","the","other"];

processSplicedValue(arrayOfStrings2,3,console.log);

//4e.

let arrayOfStrings3 = ["4e.","this","that","the","other"];

processSplicedValue(arrayOfStrings3,3,alert);

// let arrayOfStrings4 = ["this","that","the","other"];

// processSplicedValue(arrayOfStrings4,2,2);

//4f.

let arrayOfStrings4 = ["4f.","this","that","the","other"];

processSplicedValue(arrayOfStrings4,3,function(){alert(arrayOfStrings4)});

//4g.

let arrayOfStrings5 = ["this","that","the","other"];

processSplicedValue(arrayOfStrings5,3, function addQuestionPrefix() {

        console.log("4g. " + arrayOfStrings5);

        }

    );

// var users = ["Sam","Ellie","Bernie"];

// function addUser(userName){

//     setTimeout(function(){

//         users.push(userName);

//     },101);

// }

// function getUsers(){

//     setTimeout(function(){

//         console.log(users);

//     },100);

// }

// addUser("Jake");

// getUsers();

// function addUser2(userName,callback){

//     setTimeout(function(){

//         users.push(userName);

//         callback();

//     },1000);

// }

// function getUsers2(){

//     setTimeout(function(){

//         console.log(users);

//     },100);

// }

// addUser2("Jake",getUsers2);

**URL to GitHub Repository:**

[mctimoth/FESD-Week4 (github.com)](https://github.com/mctimoth/FESD-Week4)