Hands-on Activity 3.1 | Functions in JavaScript	
Course Code: CPE 026	**Program:** Computer Engineering
Course Title: Emerging Technologies 3 in CpE	**Date Performed: 9/4/2024**
Section: BSCPE41S8	**Date Submitted: 9/4/2024**
Name: Christian Ed B. Efa	**Instructor:** Engr. Roman Richard

1. Discussion

Discuss here the relevant concepts of the activity in your own words.
Module 5:

- On this module I have tackled on how the functions works and the Laboratory to proper execute the codes by the given problems.

2. Materials and Equipment

What materials did you use? Explain in detail.

- **Netacad**

Logging in the netacad and opening the course JSE (JavaScript Essentials) and using the Edube Interactive as my learning module and also my IDE .

3. Procedure

What are the procedures that you performed?
5.1.1.3 Functions

```javascript
let temperatures;
let sum;
let meanTemp;

temperatures = [12, 12, 11, 11, 10, 9, 9, 10, 12, 13, 15, 18, 21, 24, 24, 23, 25, 25, 23, 21, 20, 19, 17, 16];
sum = 0;
for (let i = 0; i < temperatures.length; i++) {
sum += temperatures[i];
}
meanTemp = sum / temperatures.length;
console.log(`mean: ${meanTemp}`); // -> mean: 16.666666666666668

temperatures = [17, 16, 14, 12, 10, 10, 10, 11, 13, 14, 15, 17, 22, 27, 29, 29, 27, 26, 24, 21, 19, 18, 17, 16];
sum = 0;
for (let i = 0; i < temperatures.length; i++) {
sum += temperatures[i];
}
meanTemp = sum / temperatures.length;
console.log(`mean: ${meanTemp}`); // -> mean: 18.083333333333332
```

- This program will calculate and display the mean daily temperature on the basis of the provided data (24 temperature measurements, in hourly intervals, starting from midnight).

5.1.1.4 Declaring functions

```
let temperatures;
let sum;
let meanTemp;

function getMeanTemp() {
    sum = 0;
    for (let i = 0; i < temperatures.length; i++) {
        sum += temperatures[i];
    }
    meanTemp = sum / temperatures.length;
}

```

- A function statement starts with the function keyword followed by the function name. Function names need to follow the same rules as variable names, and should also be meaningful. declaring a function is only a preparation. In order to execute this code, we have to call the function.

5.1.1.7 Return statement

« 5.1.1.7 The return statement »

```
function showMsg() {
    console.log("message 1");
    return;
    console.log("message 2");
}

showMsg(); // -> message 1
```

- The return statement causes the function to end exactly where this word occurs, even if there are further instructions. It allows us to return a given value from inside the function to the place where it was called.

5.1.1.9 Parameters

```
function add(first, second) {
return first + second;
}
```

```
let result = add(5, 7));
console.log(result); // -> 12
```

```
function getElement(elements, index) {
return elements[index];
}
```

```
let names = ["Alice", "Bob", "Eve", "John"];
let name = getElement(names, 2);
console.log(name); // -> Eve
```

```
function getMeanTemp(temperatures) {
let sum = 0;
for (let i = 0; i < temperatures.length; i++) {
sum += temperatures[i];
}
return sum / temperatures.length;
}

let day1 = [12, 12, 11, 11, 10, 9, 9, 10, 12, 13, 15, 18, 21, 24, 24, 23, 25, 25, 23, 21, 20, 19, 17,
console.log(`mean: ${getMeanTemp(day1)}`); // -> mean:
 16.666666666666668

let day2 = [17, 16, 14, 12, 10, 10, 10, 11, 13, 14, 15, 17, 22, 27, 29, 29, 27, 26, 24, 21, 19, 18, 17
console.log(`mean: ${getMeanTemp(day2)}`); // -> mean:
 18.083333333333332
```

- The use of parameters is optional, however most often we create functions that have defined parameters and return values.

5.1.1.10 Shadowing

```
function add(first, second) {
return first + second;
}

let first = 10, second = 20, third = 40, fourth = 80;
console.log(add(first, second)); // -> 30
console.log(add(second, third)); // -> 60
console.log(add(third, fourth)); // -> 120
```

```
let a = 100, b = 200, c = 300;

function test(a) {
    let b = 10;
    console.log(a); // parameter a
    console.log(b); // local variable b
    console.log(c); // global variable c
}

test(1);        // -> 1
                // -> 10
                // -> 300

console.log(a); // -> 100
console.log(b); // -> 200
console.log(c); // -> 300
```

- the parameters are treated inside the function as local variables. And just like the local variables explicitly declared inside a function, they shadow the global variables of the same name (or more generally, variables from the outer scope).

5.2.1.2 Parameters validation

```
function getMeanTemp(temperatures) {
    if (!(temperatures instanceof Array)) {
        return NaN;
    }
    let sum = 0;
    for (let i = 0; i < temperatures.length; i++) {
        sum += temperatures[i];
    }
    return sum / temperatures.length;
}

console.log(getMeanTemp(10));        // -> NaN
console.log(getMeanTemp([10, 30])); // -> 20
```

- we can validate or check if the value passed to it is actually an array.

5.2.1.8 Asynchronous callbacks

« 5.2.1.8 Asynchronous callbacks »

```
let inner = function() {
console.log('inner 1');
}

let outer = function(callback) {
console.log('outer 1');
setTimeout(callback, 1000) /*ms*/;
console.log('outer 2');
}

console.log('test 1');
outer(inner);
console.log('test 2');
```

- Asynchronous callback is an operation of programs is a rather complex topic, strongly dependent on a particular programming language, and often also on the environment.

5.2.1.9 setTimeout and setInterval functions

« 5.2.1.9 setTimeout and setInterval functions »

```
let inner = function() {
console.log('inner 1');
}

let outer = function(callback) {
console.log('outer 1');
let timerId = setInterval(callback, 1000) /*ms*/;
console.log('outer 2');

setTimeout(function(){
    clearInterval(timerId);
}, 5500);
}

console.log('test 1');
outer(inner);
console.log('test 2');
```

- The setInterval function returns an identifier during the call, which can be used to remove the timer used in it (and consequently to stop the cyclical callback function call). The setTimeout function is used when you want to cause a delayed action.

5.2.1.14 LAB: Functions

« 5.2.1.14 LAB: Functions (1/2) »

```
let contacts = [{
name: "Maxwell Wright",
phone: "(0191) 719 6495",
email: "Curabitur.egestas.nunc@nonummyac.co.uk"
}, {
name: "Raja Villarreal",
phone: "0866 398 2895",
email: "posuere.vulputate@sed.com"
}, {
name: "Helen Richards",
phone: "0800 1111",
email: "libero@convallis.edu"
}];
```

```
function showContact(contacts, index) {
    if (contacts instanceof Array && typeof index === 'number' && index >= 0 && index < contacts.length) {
        console.log(`Name: ${contacts[index].name}`);
        console.log(`Phone: ${contacts[index].phone}`);
        console.log(`Email: ${contacts[index].email}`);
    } else {
        console.error("Invalid input or index out of range.");
    }
}


function showAllContacts(contacts) {
    if (contacts instanceof Array) {
        contacts.forEach((contact, index) => {
            console.log(`Contact ${index + 1}:`);
            console.log(`Name: ${contact.name}`);
            console.log(`Phone: ${contact.phone}`);
            console.log(`Email: ${contact.email}`);
            console.log('---');
        });
    } else {
        console.error("Invalid input: contacts should be an array.");
    }
}

function addNewContact(contacts, name, phone, email) {
    if (contacts instanceof Array && name && phone && email) {
        contacts.push({
            name: name,
            phone: phone,
            email: email
        });
        console.log("New contact added successfully.");
    } else {
        console.error("Invalid input: check if contact list is an array and all new contact data are provided.");
    }
}


showContact(contacts, 1);
showAllContacts(contacts);
addNewContact(contacts, "John Doe", "123-456-7890", "john@example.com");
showAllContacts(contacts);
```

- The task asked to use code such as showContacts,instanceofArray,showAllContacts,addNewContacts in order to show a single contact, showing all contacts, or adding a new contact.

Part 2

```javascript
let contacts = [{
    name: "Maxwell Wright",
    phone: "(0191) 719 6495",
    email: "Curabitur.egestas.nunc@nonummyac.co.uk"
}, {
    name: "Raja Villarreal",
    phone: "0866 398 2895",
    email: "posuere.vulputate@sed.com"
}, {
    name: "Helen Richards",
    phone: "0800 1111",
    email: "libero@convallis.edu"
}];

let showContact = function (contacts, i) {
    if (contacts instanceof Array && contacts[i]) {
        console.log(`${contacts[i].name} / ${contacts[i].phone} / ${contacts[i].email}`);
    }
}

let showAllContacts = function (contacts) {
    if (contacts instanceof Array) {
        for (let contact of contacts) {
            console.log(`${contact.name} / ${contact.phone} / ${contact.email}`);
```

app.js index.html style.css

```
Before sorting:
Maxwell Wright / (0191) 719 6495 / Curabitur.egestas.nunc@nonummyac.co.uk
Raja Villarreal / 0866 398 2895 / posuere.vulputate@sed.com
Helen Richards / 0800 1111 / libero@convallis.edu
Contacts sorted by name:
Helen Richards / 0800 1111 / libero@convallis.edu
Maxwell Wright / (0191) 719 6495 / Curabitur.egestas.nunc@nonummyac.co.uk
Raja Villarreal / 0866 398 2895 / posuere.vulputate@sed.com
Contacts sorted by phone:
Maxwell Wright / (0191) 719 6495 / Curabitur.egestas.nunc@nonummyac.co.uk
Helen Richards / 0800 1111 / libero@convallis.edu
Raja Villarreal / 0866 398 2895 / posuere.vulputate@sed.com
Contacts sorted by email:
Maxwell Wright / (0191) 719 6495 / Curabitur.egestas.nunc@nonummyac.co.uk
Helen Richards / 0800 1111 / libero@convallis.edu
Raja Villarreal / 0866 398 2895 / posuere.vulputate@sed.com
```

- showContact Function prints a specific contact's details (name, phone, email) using template literals.
- showAllContacts Function iterates through the contacts array and prints each contact's details using template literals.
- addNewContact Function adds a new contact to the array but does not print anything.
- sortContacts Function sorts the contacts based on a chosen criterion (name, phone, email) and then prints the sorted list using showAllContacts

4. Output

Screenshot of your outputs based on the procedures.

5.1.1.3 Functions

```
Console >_
mean: 16.666666666666668
mean: 18.083333333333332
```

5.1.1.4 Declaring functions

```javascript
let temperatures;
let sum;
let meanTemp;

function getMeanTemp() {
    sum = 0;
    for (let i = 0; i < temperatures.length; i++) {
        sum += temperatures[i];
    }
    meanTemp = sum / temperatures.length;
}
```

app.js

Console >_

5.1.1.7 Return statement

app.js

Console >_

message 1

5.2.1.2 Parameters validation

app.js

Console >_

```
NaN
20
```

5.2.1.8 Asynchronous callbacks

Console >_

```
test 1
outer 1
outer 2
test 2
inner 1
```

5.2.1.9 setTimeout and setInterval functions

app.js index.html style.css

Console >_

```
test 1
outer 1
outer 2
test 2
inner 1
inner 1
inner 1
inner 1
inner 1
```

5.2.1.14 LAB: Functions

Email: posuere.vulputate@sed.com

Contact 3:
Name: Helen Richards
Phone: 0800 1111
Email: libero@convallis.edu

New contact added successfully.
Contact 1:
Name: Maxwell Wright
Phone: (0191) 719 6495
Email: Curabitur.egestas.nunc@nonummyac.co.uk

Contact 2:
Name: Raja Villarreal
Phone: 0866 398 2895

Part 2

Sandbox

```javascript
let contacts = [{
    name: "Maxwell Wright",
    phone: "(0191) 719 6495",
    email: "Curabitur.egestas.nunc@nonummyac.co.uk"
}, {
    name: "Raja Villarreal",
    phone: "0866 398 2895",
    email: "posuere.vulputate@sed.com"
}, {
    name: "Helen Richards",
    phone: "0800 1111",
    email: "libero@convallis.edu"
}];

let showContact = function (contacts, i) {
    if (contacts instanceof Array && contacts[i]) {
        console.log(`${contacts[i].name} / ${contacts[i].phone} / ${contacts[i].email}`);
    }
}

let showAllContacts = function (contacts) {
    if (contacts instanceof Array) {
        for (let contact of contacts) {
            console.log(`${contact.name} / ${contact.phone} / ${contact.email}`);
```

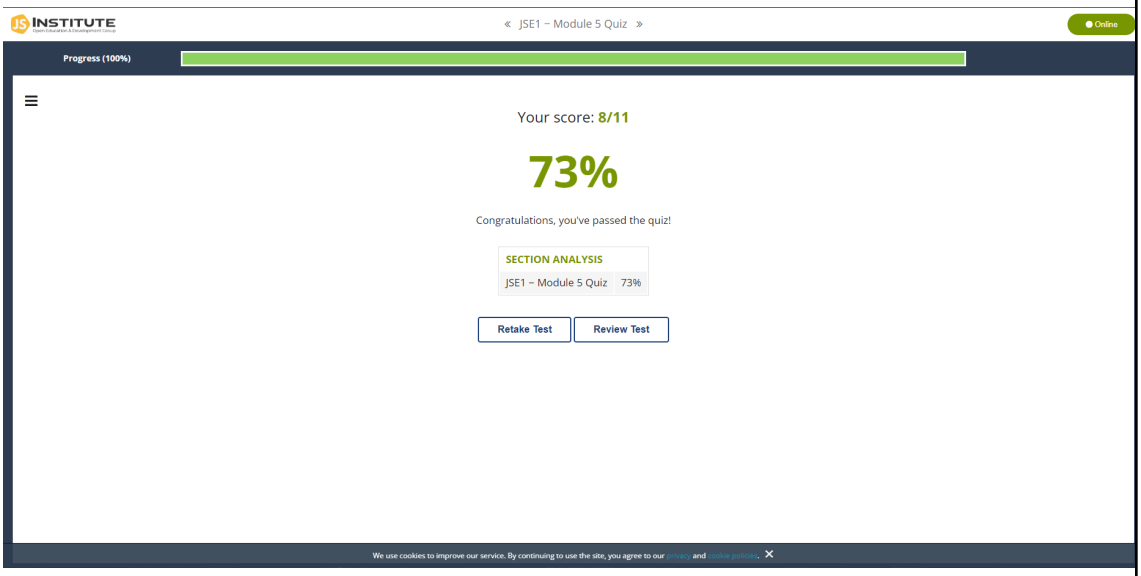
app.js index.html style.css

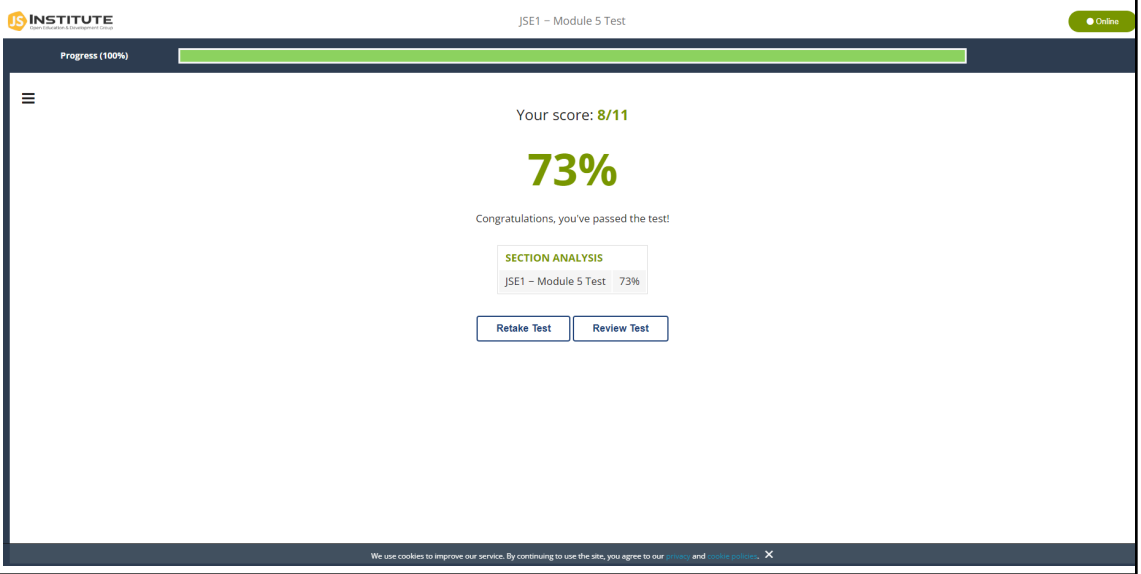
Fullscreen

Before sorting:
Maxwell Wright / (0191) 719 6495 / Curabitur.egestas.nunc@nonummyac.co.uk
Raja Villarreal / 0866 398 2895 / posuere.vulputate@sed.com
Helen Richards / 0800 1111 / libero@convallis.edu
Contacts sorted by name:
Helen Richards / 0800 1111 / libero@convallis.edu
Maxwell Wright / (0191) 719 6495 / Curabitur.egestas.nunc@nonummyac.co.uk
Raja Villarreal / 0866 398 2895 / posuere.vulputate@sed.com
Contacts sorted by phone:
Maxwell Wright / (0191) 719 6495 / Curabitur.egestas.nunc@nonummyac.co.uk
Helen Richards / 0800 1111 / libero@convallis.edu
Raja Villarreal / 0866 398 2895 / posuere.vulputate@sed.com
Contacts sorted by email:
Maxwell Wright / (0191) 719 6495 / Curabitur.egestas.nunc@nonummyac.co.uk
Helen Richards / 0800 1111 / libero@convallis.edu
Raja Villarreal / 0866 398 2895 / posuere.vulputate@sed.com

5. Supplementary Activity

Include here screenshots of the module completion test.





6. Assessment Rubric

Lab Activity Rubric

Criteria	Ratings						Pts
SO 7 PI 1 **Student Outcome 7.1** Acquire and apply new knowledge from outside sources. threshold: 4.8 pts	6 pts Excellent | Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently and applies knowledge learned into practice	5 pts Good | Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently	4 pts Satisfactory | Look beyond classroom requirements, showing interest in pursuing knowledge independently	3 pts Unsatisfactory | Begins to look beyond classroom requirements, showing interest in pursuing knowledge independently	2 pts Poor | Relies on classroom instruction only	1 pts Very Poor | No initiative or interest in acquiring new knowledge	6 pts
SO 7 PI 2 **Student Outcome 7.2** Learn independently threshold: 4.8 pts	6 pts Excellent | Completes an assigned task independently and practices continuous improvement	5 pts Good | Completes an assigned task without supervision or guidance	4 pts Satisfactory | Requires minimal guidance to complete an assigned task	3 pts Unsatisfactory | Requires detailed or step-by-step instructions to complete a task	2 pts Poor | Shows little interest to complete a task independently	1 pts Very Poor | No interest to complete a task independently	6 pts
SO 7 PI 3 **Student Outcome 7.3** Critical thinking in the broadest context of technological change threshold: 4.8 pts	6 pts Excellent | Synthesizes and integrates information from a variety of sources; formulates a clear and precise perspective; draws appropriate conclusions	5 pts Good | Evaluate information from a variety of sources; formulates a clear and precise perspective.	4 pts Satisfactory | Analyze information from a variety of sources; formulates a clear and precise perspective.	3 pts Unsatisfactory | Apply the gathered information to formulate the problem	2 pts Poor | Gather and summarized the information from a variety of sources but failed to formulate the problem	1 pts Very Poor | Gather information from a variety of sources	6 pts
SO 7 PI 4 **Student Outcome 7.4** Creativity and adaptability to new and emerging technologies threshold: 4.8 pts	6 pts Excellent | Ideas are combined in original and creative ways in line with the new and emerging technology trends to solve a problem or address an issue.	5 pts Good | Ideas are creative and adapt the new knowledge to solve a problem or address an issue	4 pts Satisfactory | Ideas are creative in solving a problem, or address an issue	3 pts Unsatisfactory | Shows some creative ways to solve the problem	2 pts Poor | Shows initiative and attempt to develop creative ideas to solve the problem	1 pts Very Poor | Ideas are copied or restated from the sources consulted	6 pts
							Total Points: 24