CA 107 – WEB DEVELOPMENT Lab

Topic : Lab of web development.

(Prepared by : ***MANDEEP KUMAR***)
[Guided by : ***Prof. Anirban Choudhary***]

- **NAME : MANDEEP KUMAR**
- **Scholor No. : 212120139**
- **Subject code: 107**
- **Stream : MCA**
- **Semester : Ist**

1. ACKNOWLEDGEMENT

I am MANDEEP KUMAR students of M.C.A 1st Sem, Session:- 2021-2024, From MANIT, Bhopal sincerely thanks to offer our heartlly gratitude to all those who helped us in preparing this Assignment.

Apart from the efforts of myself, the success of any Assignment depends largely on the encouragement and guidelines of Prof. Anirban Choudhury Sir. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this Assignment. I can't say thank you enough for his tremendous support and help. I feel motivated and encouraged every time, Without his encouragement and guidance this assignment would not have materialized. The guidance and support received from all the members who contributed and who are contributing to this assignment,I am grateful for their constant support and help.

Thank You Sir

2. Contents

3. Lab 1

➡ HTML

a) ABOUT ME

Gits hub id:-›
https://mandeepsingh9.github.io/feedbackJavaScript/

4. Lab 2

➡ CSS

a) ABOUT Me

Gits hub id:-›
https://mandeepsingh9.github.io/feedbackJavaScript/

5. Lab 3

➡ Javascript

a) Feedback Form

Gits hub id:-›
https://mandeepsingh9.github.io/feedbackJavaScript/

5. Lab 4

1. Write a function named minimum, that takes two argument and returns their minimum. Illustrate all the possible way of writing function in Java Script.

```
// compare firstNum and secondNum and return the smaller
of the two,
function min( firstNum, secondNum )
{
    if ( firstNum < secondNum )
     return firstNum;
     else
     return secondNum;

}

console.log(min(0, 10));
// → 0
console.log(min(0, -10));
// → -0
```

2.

A) Write a range function, that take two arguments, start and end, and returns an array containing all the numbers from the start up to, and including end Also, write a sum function, that takes an array of numbers, and returns the sum of these numbers.

```
Trace: console.log(sum(range(1,10))); //55.
// range function
function range( start, end, increment ) {
// create the result array
var result = [];
// test to see if we have an increment, otherwise set it to 1
if ( increment == undefined )
increment = 1;
// calculate the number of times to loop (this is because you
might be going
// up or down with your increment)
numLoops = Math.abs( (end - start)/ increment ) + 1 ;
// loop that many times
for ( var i = 0; i < numLoops; i ++ ) {
// add (push) the value of start to the array
result.push( start );
// increment the value of start
// loop that many times
for ( var i = 0; i < numLoops; i ++ ) {
// add the number at that index to the sum
arrayTotal += numArray[i];
}

// return the sum
return arrayTotal;
```

```
}
console.log(range(1, 10));
// → [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
console.log(range(5, 2, -1));
// → [5, 4, 3, 2]
console.log(sum(range(1, 10)));
// → 55
```

B) Modify the range function to take an optional argument that indicate that step value, when building the array. If no step is given, the elements go by incrementing by one, corresponding to old behavior. Trace: function call: range(1,10,2); // [1,3,5,7,9].

```
function range(start, end, increment){
var array = [];
var current = start;
var counter;
if (increment == undefined){
counter = 1;
}
else {
counter = increment;
}
if (increment > 0){
while(current <= end){
array.push(current);
current += counter;
}
}
else if (increment < 0){
while(current >= end){
array.push(current);

current += counter;
}
```

```
}
return array;
}
```

3. Write two functions, reverseArray and reverseArrayInplace, where reverseArray, takes an array, as argument and produce a new array, that has the same elements in reverse order, and reverseArrayInplace, modifies the array given as an argument by reversing its element.

```
function reverseArray( arrayToInvert ) {
// create a variable for the new array
var invertedArray = [];
// count how many things are in the original array
numLoops = arrayToInvert.length;
// loop that many times backwards and put each element into the
new array
for ( i = numLoops -1; i >= 0; i-- ) {
invertedArray.push( arrayToInvert[i]);
}
// return the inverted array
return invertedArray;
}
function reverseArrayInPlace( arrayToInvert ) {
// calculate the length of the array
invertArrayLength = arrayToInvert.length;
// calculate half the length of the array
numLoops = Math.floor( invertArrayLength / 2 );
// loop that many times and swap the first and last elements
for ( i = 0; i < numLoops; i++ ) {
// hold on to the current number
var holdNum = arrayToInvert[i];
// get the position of the swap number
var swapPos = invertArrayLength - 1 - i;
var arrayValue = [1, 2, 3, 4, 5];
reverseArrayInPlace(arrayValue);
```

```
console.log(arrayValue);
// → [5, 4, 3, 2, 1]
```

4.) Write a class Vec, that represents a vector in two-dimensional space. It takes x and y parameters (Number), which it should save to properties of the same name. Create two method, plus and minus, that take another Vec as parameter and returns a new Vec that has the sum and difference of the two vectors {this and the parameter} x and y values respectively. Add the getter property length to the prototype that computes, the length of the vector, that is, the distance of the point (x,y) from the origin.

```
class Vec {
constructor (x, y) {
this.x = x;
this.y = y;
}
plus(v) {
return new Vec(this.x + v.x, this.y + v.y);
}
minus(v) {
return new Vec(this.x - v.x, this.y - v.y);
}
get length() {
return Math.sqrt(Math.pow(this.x, 2) + Math.pow(this.y, 2));
}
}
console.log(new Vec(1, 2).plus(new Vec(2, 3)));
// → Vec{x: 3, y: 5}
console.log(new Vec(1, 2).minus(new Vec(2, 3)));
// → Vec{x: -1, y: -1}
console.log(new Vec(3, 4).length);
// → 5
```

5.

Write a class called Group, which has three methods: add, delete, and has. Its constructor creates an empty Group, and adds a value to the Group, only if it isn't already a member. Similarly, delete removes its argument from the Group, if it has a member, and has returns a Boolean value indicating whether its argument is a member of the Group.

```
class Group {
constructor() {
this.group = [];
return this;
}
add(value) {
if (!this.has(value)) {
this.group.push(value);
return this;
}
}
delete(value) {
if (this.has(value)) {
this.group = this.group.filter(x => x !== value)
return this;
}
}
has(value) {
return this.group.includes(value)
}
static from(iterableObject) {
var newGroup = new Group(iterableObject);
for (let value of iterableObject) {
```

```
newGroup.add(value);
}
return newGroup;
}
}
let group = Group.from([10, 20]);
console.log(group.has(10));
// → true
console.log(group.has(30));
// → false
group.add(10);
group.delete(10);
console.log(group.has(10));
// → false
```