1. Functions and Prototyping…

Source Code :

**Index.html :**

<html>

    <body>

        <h1>MEAN Stack</h1>

        <p> This is functions and prototypes demo. </p>

        <script src="functions-and-prototypes.js"></script>

    </body>

</html>

**functions-and-prototypes.js :**

// function constructor

function Employee(name, designation, yearOfBirth){

    this.name= name;

    this.designation= designation;

    this.yearOfBirth= yearOfBirth;

}

// creating calculateAge() method to the Prototype property

Employee.prototype.calculateAge= function(){

    console.log('The current age of ' + this.name + ' is: '+(2023- this.yearOfBirth));

}

console.log(Employee.prototype);

// creating Objects

let Emp1= new Employee('Adam', 'Tester', 1997);

console.log(Emp1) ;

Emp1.calculateAge();

let Emp2= new Employee('John', 'Senior Developer', 1982);

console.log(Emp2)

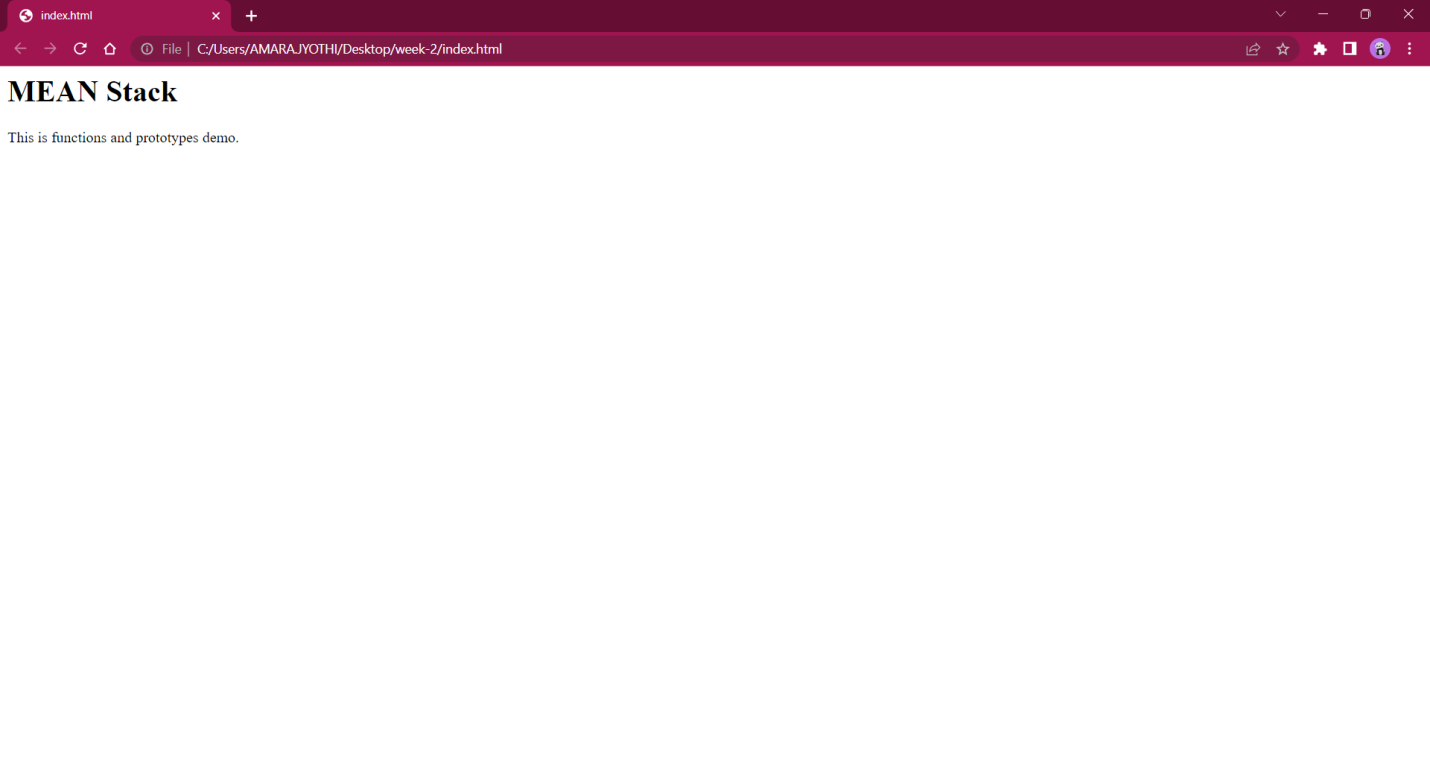
Emp2.calculateAge();

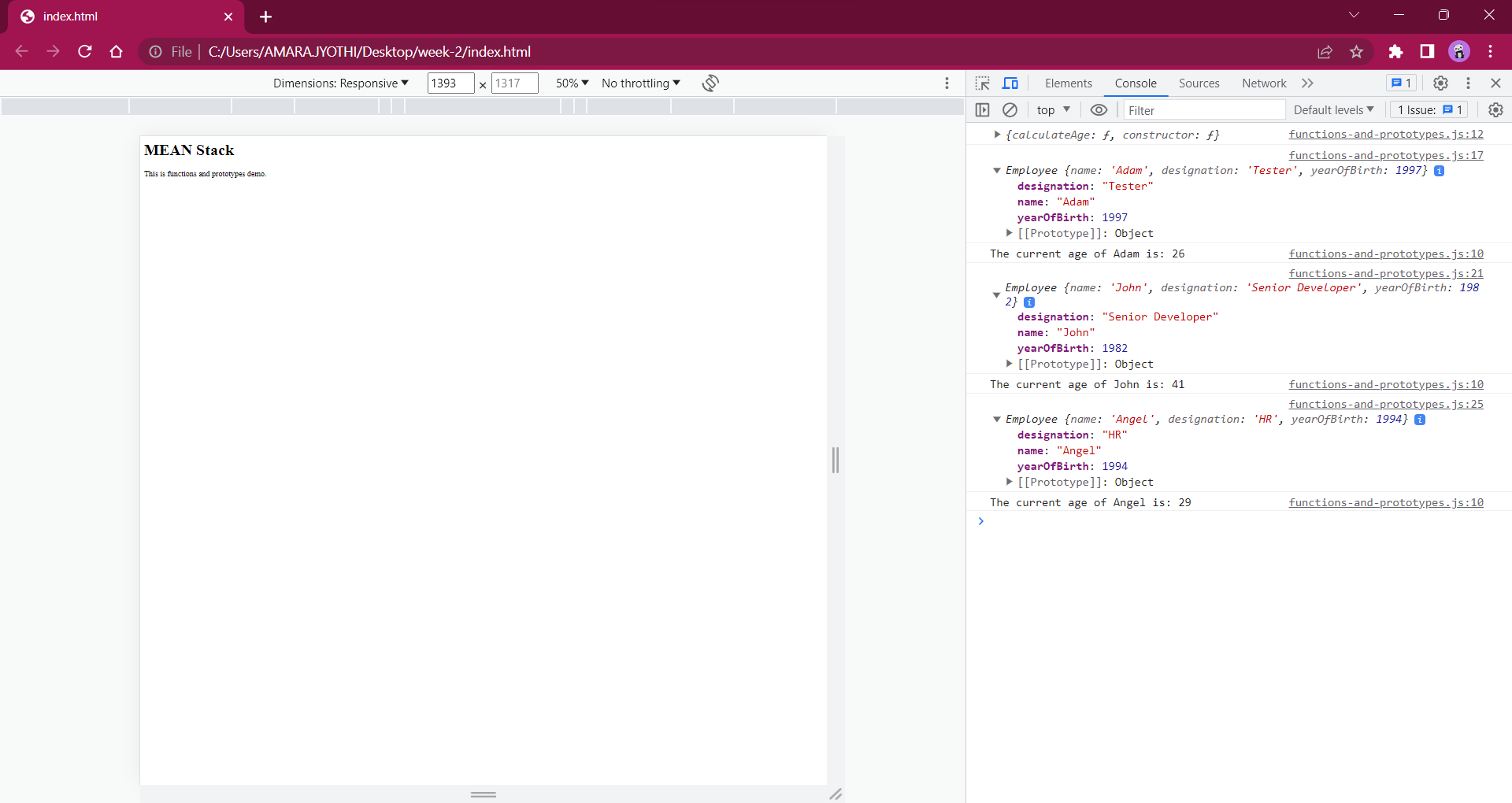
let Emp3= new Employee('Angel', 'HR', 1994);

console.log(Emp3)

Emp3.calculateAge();

Output :





2. Working with Functions…

Source Code :

**Index.html :**

<html>

    <body>

        <h1>MEAN Stack</h1>

        <p> This is JAVASCRIPT-Functions demo. </p>

        <script src="function.js"></script>

    </body>

</html>

**function.js :**

var x = (4 \* 3) - 5;

    var y = 3 \* 4;

    console.log(x);

    console.log(y);

    function myFunction(num1, num2) {

        var p = num1 + num2;

        var q = num1 \* num2;

        return(p + q);

    }

    console.log( myFunction(3, 4));

    var result = myFunction(3,3);

    console.log(result);

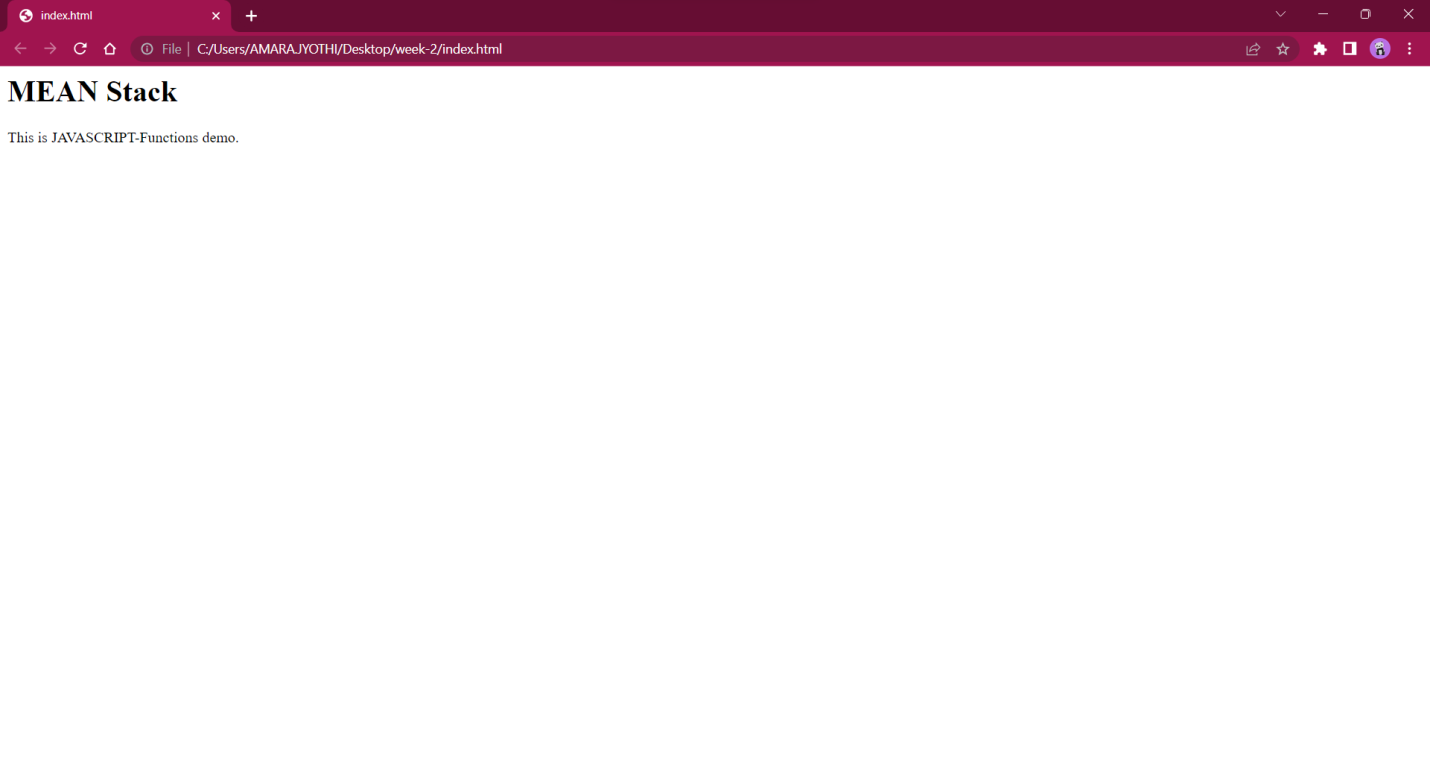
    function convertToCelcius(f){

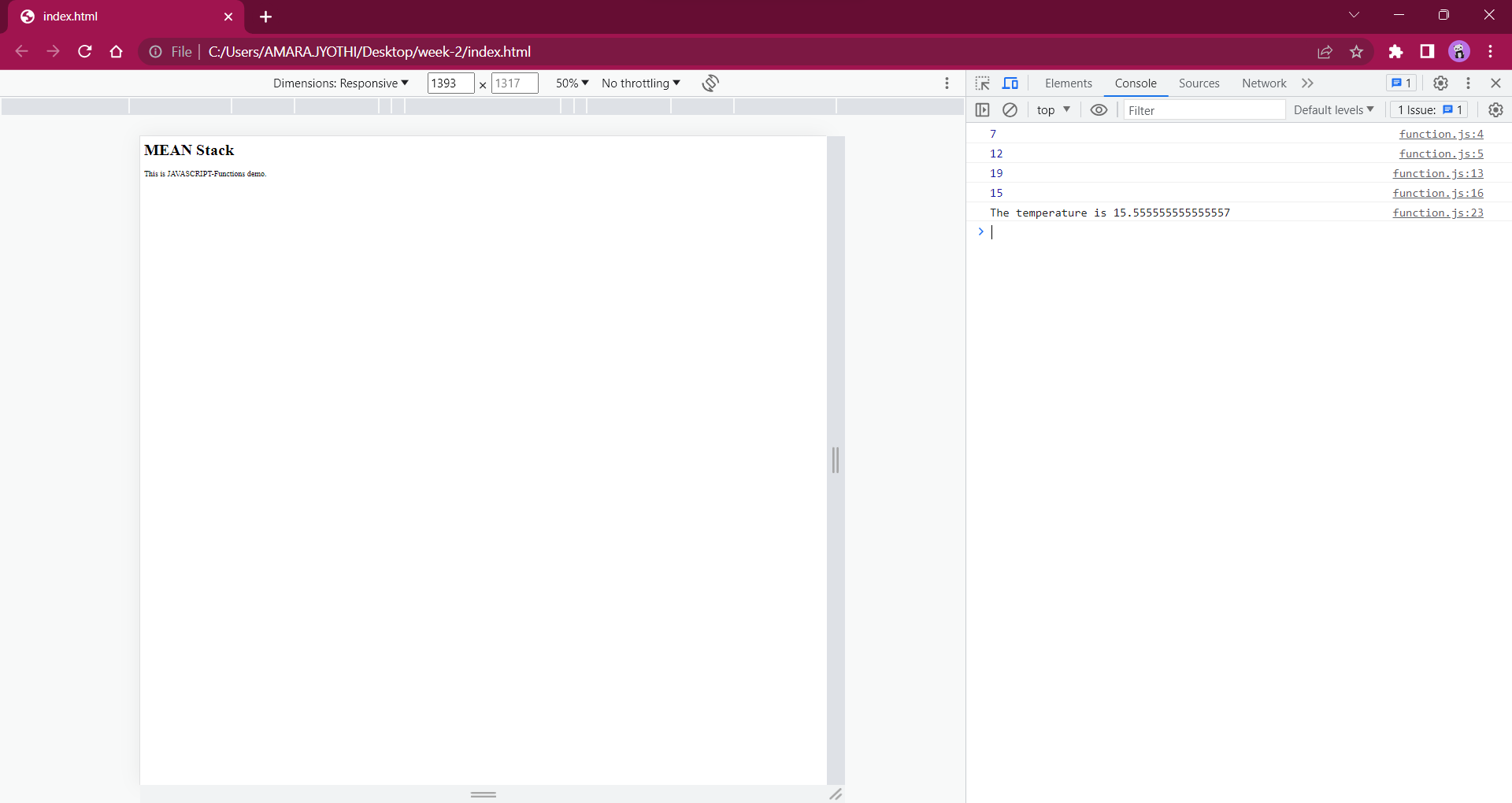
            return (5/9) \* (f-32);

    }

    console.log("The temperature is "+ convertToCelcius(60));

Output :





3. IIFEs, Callbacks, and Closures…

Source Code :

**Index.html :**

<html>

    <body>

        <h1>MEAN Stack</h1>

        <p> This is IIFEs,Callbacks and Closures demo. </p>

        <script src="IIFEs\_Callbacks\_Closures.js"></script>

    </body>

</html>

**IIFEs\_Callbacks\_Closures.js :**

//IIFE and Closure

const empId = (function() {

    let count = 0;

    return function() {

      ++count;

      return `emp${count}`;

    };

  })();

  console.log("New Emplyee IDs are listed as follows");

  console.log("Ram: "+empId());

  console.log("Dany: "+empId());

  console.log("Angel: "+empId());

  //Callbacks

  console.log("\n"); //to start the output from the next line

  function fullName(firstName, lastName, callback){

    console.log("My name is " + firstName + " " + lastName);

    callback(lastName);

  }

  var greeting = function(ln){

    console.log('Welcome ' + ln);

  };

  fullName("Ram", "Harsh", greeting);

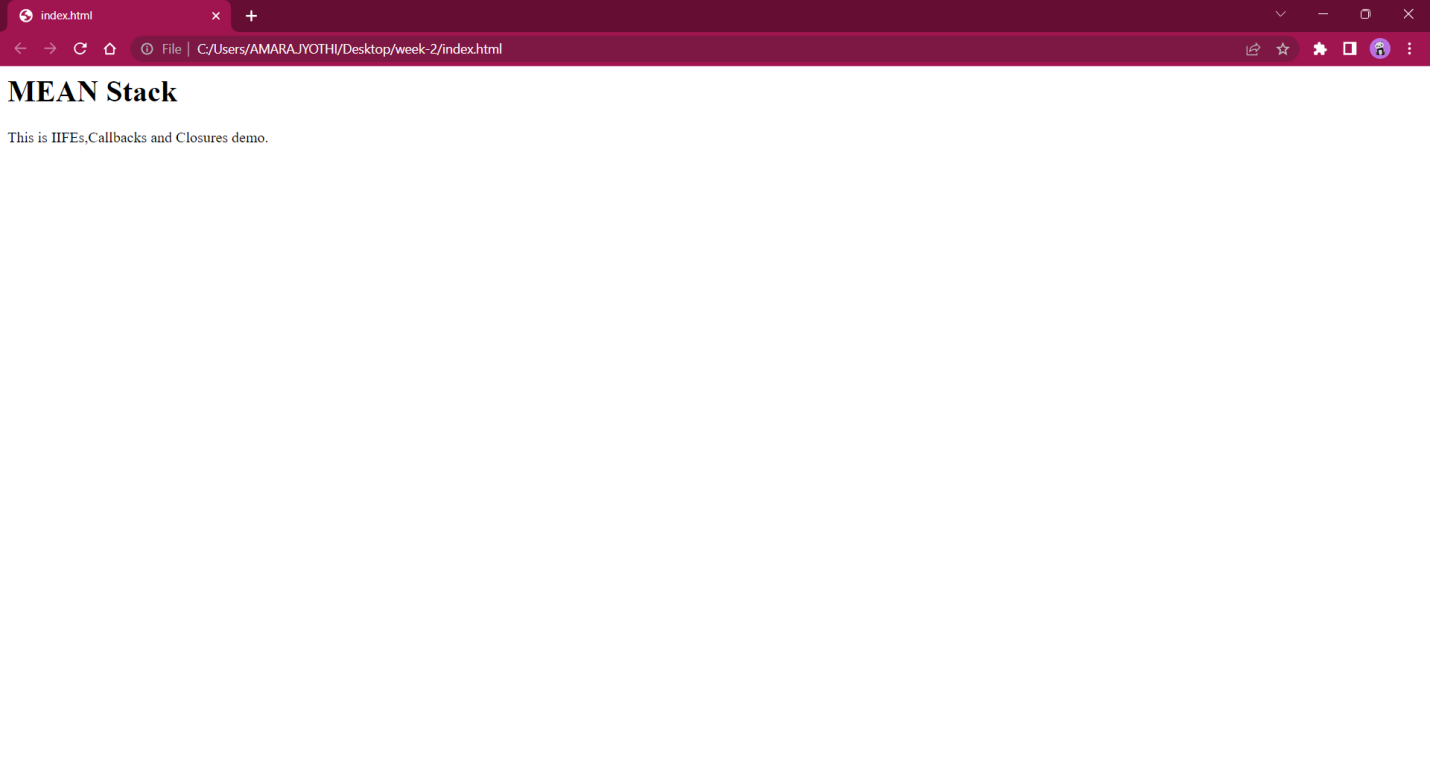
  console.log("\n");

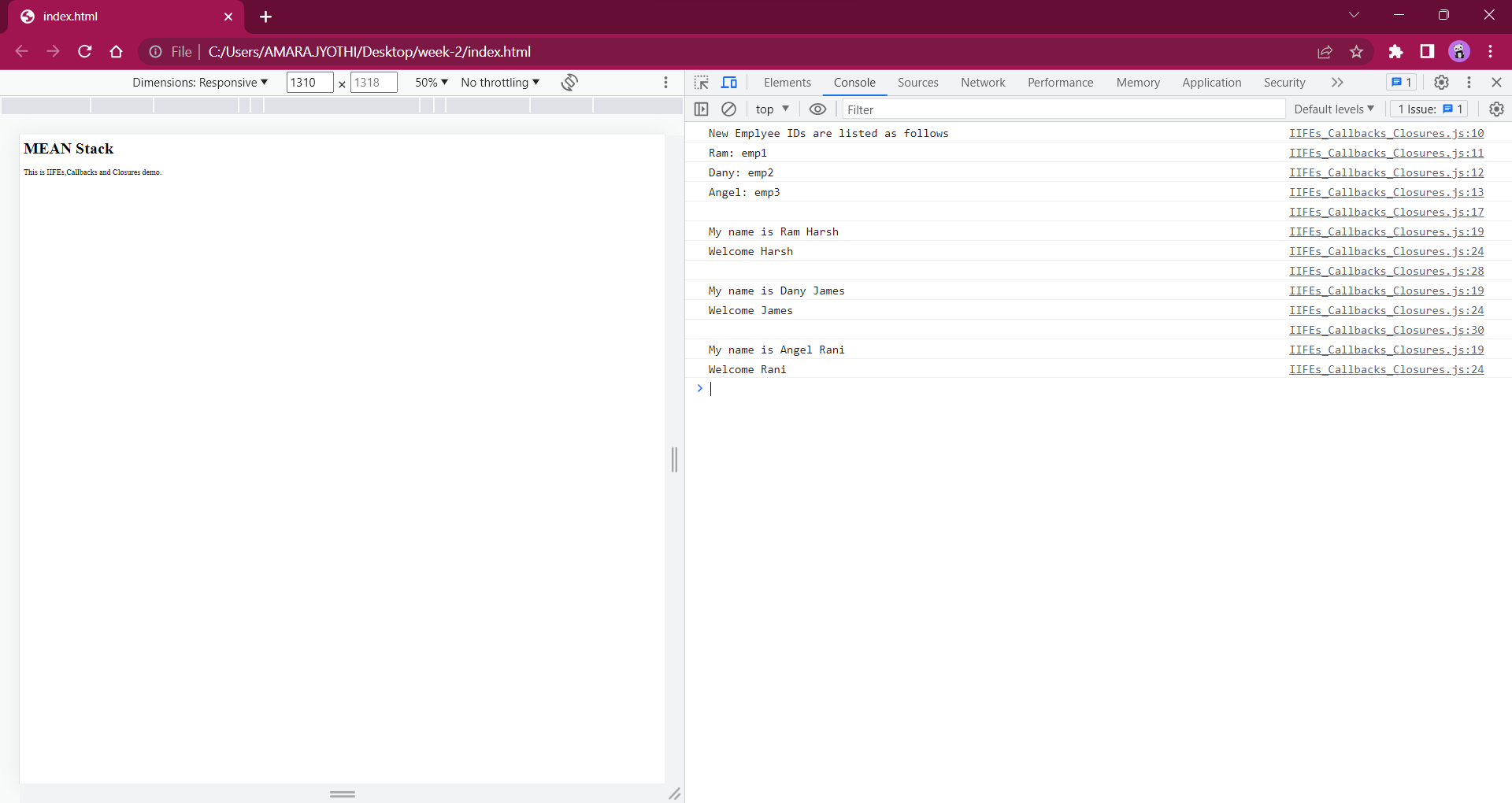
  fullName("Dany", "James", greeting);

  console.log("\n");

  fullName("Angel", "Rani", greeting);

Output :





4. Maps and Classes…

Source Code :

**Index.html :**

<html>

    <body>

        <h1>MEAN Stack</h1>

        <p> This is Maps and Classes demo. </p>

        <script src="maps-and-classes.js"></script>

    </body>

</html>

**Maps-and-classes.js :**

var map1 = new Map();

map1.set("first name", "Jhony");

map1.set("last name", "Stella");

map1.set("friend 1","Bindu")

    .set("friend 2","Arya");

console.log(map1);

console.log("map1 has friend 3 ? " + map1.has("friend 3"));

console.log("get value for key = friend 3 - "+ map1.get("friend 3"));

console.log("delete element with key = friend 2 - " + map1.delete("friend 2"));

map1.clear();

console.log(map1);

class Employee

{

    constructor(id,name)

    {

      this.id=id;

      this.name=name;

    }

    detail()

    {

  document.writeln(this.id+" "+this.name+"<br>")

    }

  }

//passing object to a variable

var e1=new Employee(101,"Michael");

var e2=new Employee(102,"Bob");

e1.detail();

e2.detail();

Output :

