

FULL STACK WEB DEVELOPMENT INTERNSHIP
DAILY ANALYSIS REPORT

WEEK 3
(14/4/2020 - 22/4/2020)

Amala Peter
3rd year Computer Science and Engineering Graduate
NSS College of Engineering, Palakkad

Asst.Prof Albert Sunny
IIT Palakkad

CONTENTS

Contents	Page No
Chapter 1 :FRONT END DEVELOPMENT	4
1.4 Javascript	4
1.4.9 Interaction:alert,prompt,confirm	4
1.4.10 for,while,forEach,Do while Loops	8
1.4.11 Javascript HTML DOM	13
1.4.12 Events & Listening to Events	18
1.4.13 setInterval & setTimeout	22
1.4.14 Date and Time	26
1.4.15 Arrow Functions	28
1.4.16 Math Object	30
1.4.17 Working with JSON	32
Chapter 2: BACKEND DEVELOPMENT	36
1.5 Node.Js	36
1.5.1 Introduction	36
1.5.2 Modules with Examples	37
1.5.3 Blocking vs Non-Blocking execution	38
1.5.4 Creating a custom Backend using Node.Js	40
1.5.5 Creating a custom Modules in Node using Node.Js	48
1.5.6 npm:The Node Package Manager	50
1.5.7 Express and Postman	52
1.5.8 Pug Template Engine	56
1.5.9 Raw HTML using Pug Template	60
Chapter 3: DATABASE DESIGNING	67
1.6 MongoDB	67
1.6.1 Introduction	67
1.6.2 Installation	68
1.6.3 Inserting Data	71
1.6.4 Searching/Querying Data	71
1.6.5 Deleting Data	72

1.6.6 Updating Data	72
1.6.7 MongoDB Compass & Installing Mongoose	74
1.6.8 Using Mongoose in Node.Js	75
CREATING FULLY RESPONSIVE WEBSITE	77
REFERENCES	101

1.4 Javascript

1.4.9 Interaction:alert,prompt,confirm

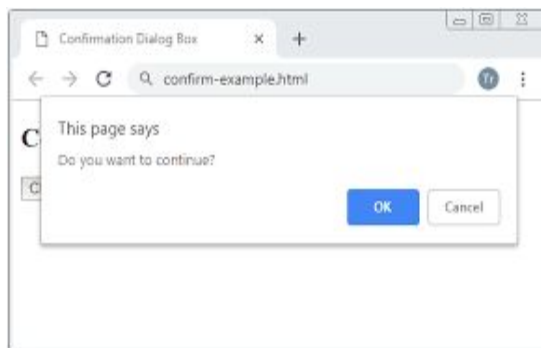
These are called user-interface functions / browser functions

Alert:-

Syntax:

```
alert(message);
```

This shows a message and pauses script execution until the user presses “OK”.The mini-window with the message is called a modal window. The word “modal” means that the visitor can’t interact with the rest of the page, press other buttons, etc. until they have dealt with the window.



Prompt:-

Syntax:-

The function prompt accepts two arguments:

```
result = prompt(title, [default]);
```

The text to show the visitor: default

It shows a modal window with a text message, an input field for the visitor, and the buttons OK/Cancel.The visitor may type something in the prompt input field and press

OK. Or they can cancel the input by pressing Cancel or hitting the Esc key. The call to prompt returns the text from the input field or null if the input was canceled.

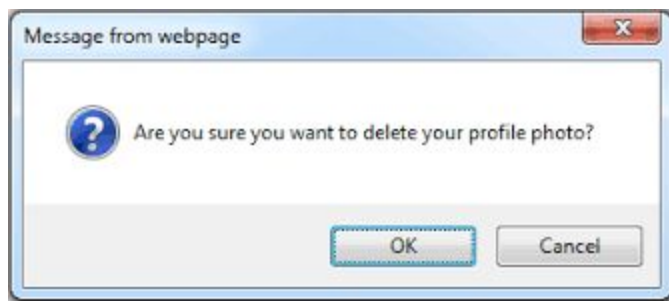


Confirm:-

Syntax:

```
result = confirm(question);
```

The function confirms a modal window with a question and two buttons: OK and Cancel. The result is true if OK is pressed and false otherwise.



Source Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Interaction using alert, confirm and prompt</title>
</head>
<body>
  <div class="container">
```

```
This is a page
</div>

<script>
  // Alert in in-browser JavaScript - Does not return anything
  // alert("This is a message");

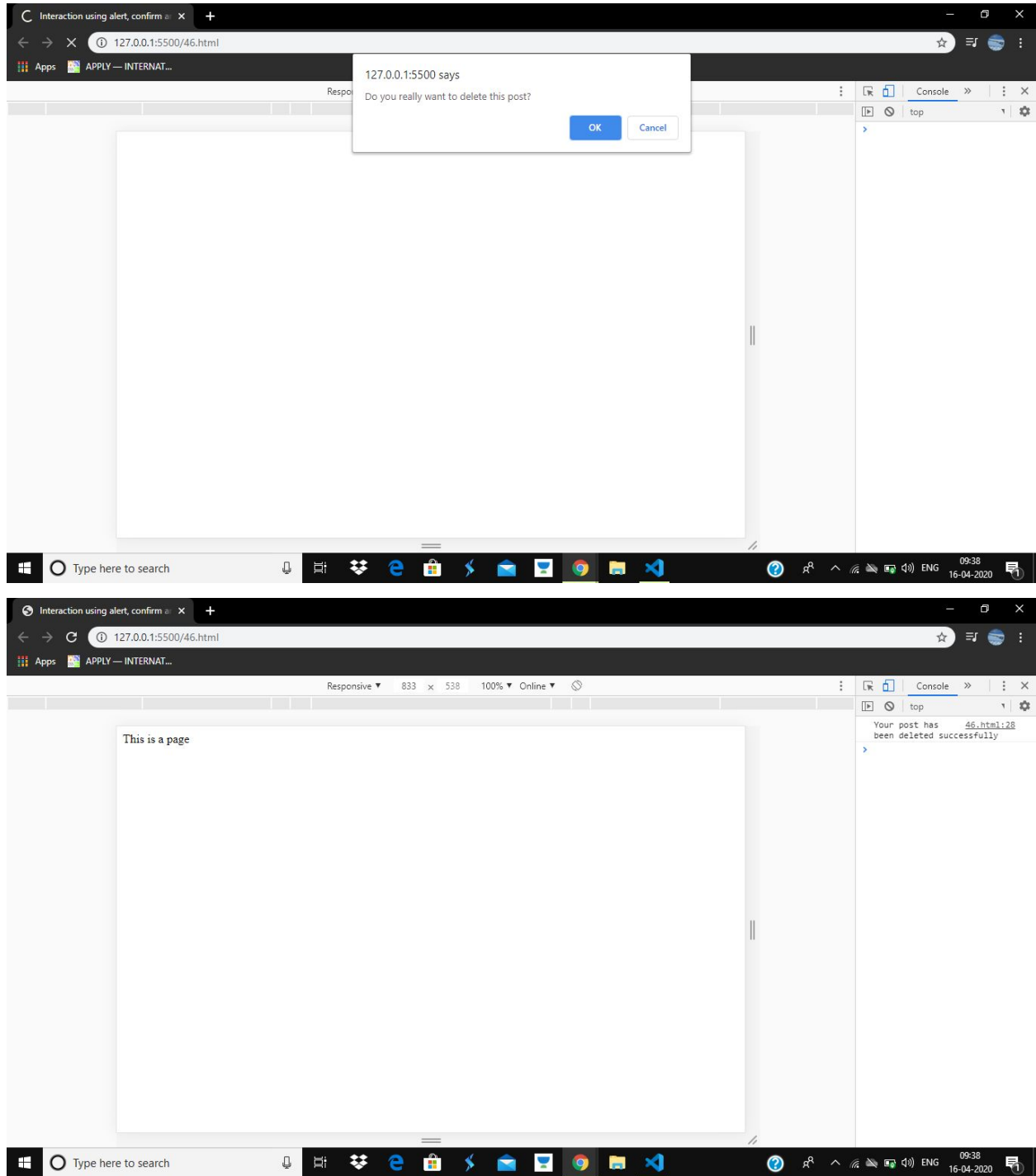
  // Prompt in JS
  // let name = prompt("What is your name?", "Guest");
  // console.log(name);

  // Confirm in JS
  let deletePost = confirm("Do you really want to delete this post?");

  // console.log(deletePost);
  if(deletePost){
    // Code to delete the post
    console.log("Your post has been deleted successfully");
  }
  else{
    // Code to cancel deletion of the post
    console.log("Your post has not been deleted");
  }

</script>
</body>
</html>
```

Output:-



1.4.10 for,while,forEach,Do while Loops

Loops can execute a block of code a number of times. Loops are handy, if you want to run the same code over and over again, each time with a different value. JavaScript supports different kinds of loops:

- for - loops through a block of code a number of times
- for/in - loops through the properties of an object
- for/of - loops through the values of an iterable object
- while - loops through a block of code while a specified condition is true
- do/while - also loops through a block of code while a specified condition is true

The For Loop:-

The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3) {  
    // code block to be executed  
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

The For/In Loop:-

The JavaScript for/in statement loops through the properties of an object:

```
var person = {fname:"John", lname:"Doe", age:25};  
var text = "";  
var x;  
for (x in person) {  
    text += person[x];  
}
```

The For/Of Loop:-

The JavaScript for/of statement loops through the values of an iterable objects

Syntax:

```
for (variable of iterable) {  
    // code block to be executed  
}
```

The While Loop:-

The while loop loops through a block of code as long as a specified condition is true.

Syntax:-

```
while (condition) {  
    // code block to be executed  
}
```

The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {  
    // code block to be executed  
}  
while (condition);
```

Source Code:-

```
<!DOCTYPE html>  
  
<html lang="en">  
  
<head>  
  
    <meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Loops</title>

</head>

<body>

  <div class="container">

    This is about loops

  </div>

  <script>

    console.log("This is amala");

    // let i = 0;

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

    //   console.log(i);

    // }

    let friends = ["Rohan", "Sanjeev", "Deepti", "Pooja", "SkillF"];

    for (let index = 0; index < friends.length; index++) {

      console.log("Hello friend, " + friends[index]);

    }

    // friends.forEach(function f(element){
```

```
// console.log("Hello Friend, " + element + " to modern JavaScript");

// });

// for (element of friends){

// console.log("Hello Friend, " + element + " to modern JavaScript again");

// }

let employee = {

    name: "Harry",

    salary: 2,

    channel: "erhie"

}

// Use this loop to iterate over objects in JavaScript

for(key in employee){

    console.log(`The ${key} of employee is ${employee[key]}`);

}

// while loop in js

let i = 0;

while(i < 4){
```

```
    console.log(`${i} is less than 4`);

    i++;

}

// do while loop in js

let j =34;

do{

    console.log(`${j} is less than 4 and we are inside do while loop`);

    j++;

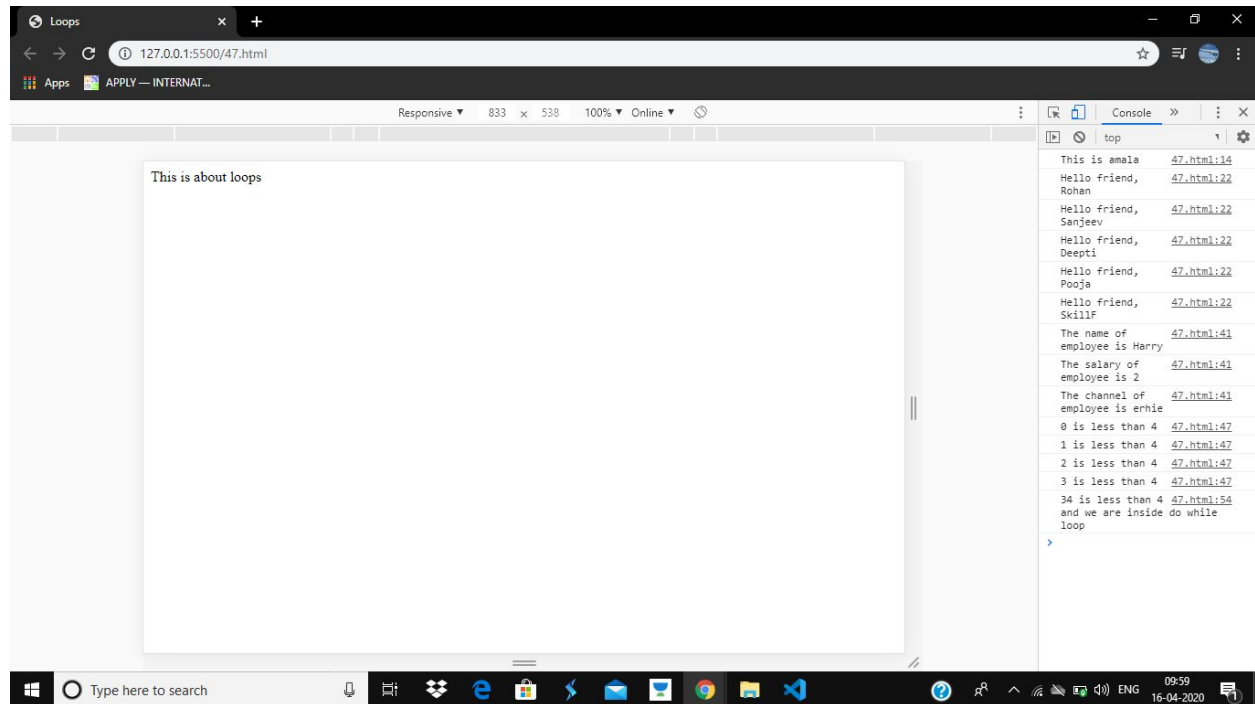
}while(j<4);

</script>

</body>

</html>
```

Output:-



1.4.11 Javascript HTML DOM

With the HTML DOM, JavaScript can access and change all the elements of an HTML document. When a web page is loaded, the browser creates a Document Object Model of the page. With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard. It defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

HTML DOM:- The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

HTML DOM methods are actions you can perform (on HTML Elements).HTML DOM properties are values (of HTML Elements) that you can set or change.HTML DOM can be accessed with JavaScript (and with other programming languages).

In the DOM, all HTML elements are defined as objects.The programming interface is the properties and methods of each object.

The getElementById Method:- The most common way to access an HTML element is to use the id of the element.

The innerHTML Property:- The easiest way to get the content of an element is by using the innerHTML property.The innerHTML property is useful for getting or replacing the content of HTML elements.

Example:-

```
<html>  
<body>
```

```
<p id="demo"></p>
```

```
<script>  
document.getElementById("demo").innerHTML = "Hello World!";
```

```
</script>
```

```
</body>
```

```
</html>
```

Finding HTML Elements

Method	Description
<code>document.getElementById(id)</code>	Find an element by element id
<code>document.getElementsByTagName(name)</code>	Find elements by tag name
<code>document.getElementsByClassName(name)</code>	Find elements by class name

Changing HTML Elements

Property	Description
<code>element.innerHTML = new html content</code>	Change the inner HTML of an element
<code>element.attribute = new value</code>	Change the attribute value of an HTML element
<code>element.style.property = new style</code>	Change the style of an HTML element
Method	Description

<code>element.setAttribute(attribute, value)</code>	Change the attribute value of an HTML element
---	---

Adding and Deleting Elements

Method	Description
<code>document.createElement(element)</code>	Create an HTML element
<code>document.removeChild(element)</code>	Remove an HTML element
<code>document.appendChild(element)</code>	Add an HTML element
<code>document.replaceChild(new, old)</code>	Replace an HTML element
<code>document.write(text)</code>	Write into the HTML output stream

Source Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Manipulating DOM</title>
</head>
<body>
  <div id="main" class="container">
    <ul id="nav">
      <li>Home</li>
```



```
<li>About</li>
<li>Services</li>
<li>More About Us</li>
<li>Contact Us</li>
</ul>
</div>
<div class="container">
  Another Container
</div>
<script>
  let main = document.getElementById('main');
  console.log(main);
  let nav = document.getElementById('nav');
  console.log(nav);

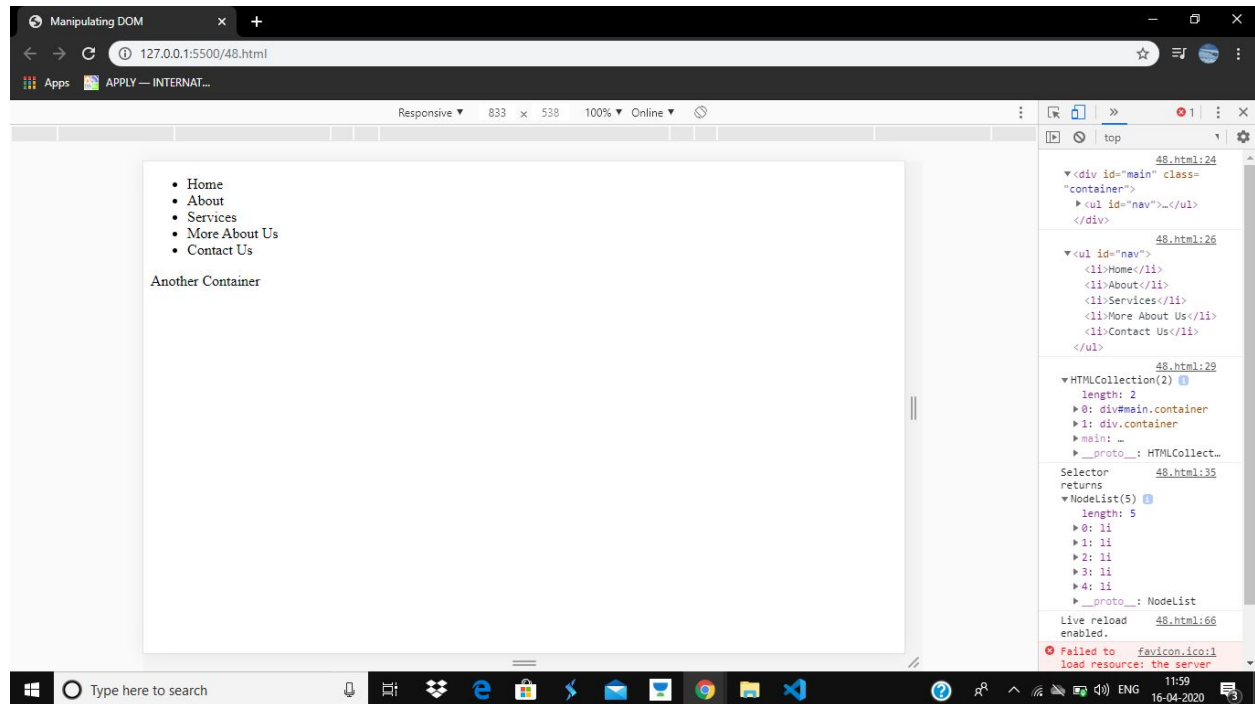
  let containers = document.getElementsByClassName('container');
  console.log(containers);

  // let sel = document.querySelector('#nav>li');
  // console.log("Selector returns ", sel)

  let sel = document.querySelectorAll('#nav>li');
  console.log("Selector returns ", sel)

</script>
</body>
</html>
```

Output:-



1.4.12 Events & Listening to Events

The `addEventListener()` method

The `addEventListener()` method attaches an event handler to the specified element. This method attaches an event handler to an element without overwriting existing event handlers. We can add many event handlers to one element.

When using the `addEventListener()` method, the JavaScript is separated from the HTML markup, for better readability and allows us to add event listeners even when you do not control the HTML markup.

Syntax

```
element.addEventListener(event, function, useCapture);
```

The first parameter is the type of the event (like "click" or "mousedown" or any other HTML DOM events). The second parameter is the function we want to call when the event occurs. The third parameter is a boolean value specifying whether to use event bubbling or event capturing. This parameter is optional.

Add an Event Handler to an Element

```
element.addEventListener("click", myFunction);

function myFunction() {
    alert ("Hello World!");
}
```

The removeEventListener() method

We can easily remove an event listener by using the removeEventListener() method. The removeEventListener() method removes event handlers that have been attached with the addEventListener() method

```
element.removeEventListener("mousemove", myFunction);
```

HTML DOM Event Objects:-

Drag, Drop, MouseClick etc...

Source Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Js Events</title>
</head>
<style>
#btn{
  cursor: pointer;
}
</style>
<body>
  <!-- Browser events:
  click
  contextmenu
  mouseover/mouseout
  mousedown/mouseup
```

```
mousemove
```

```
submit
```

```
focus
```

```
DOMContentLoaded
```

```
transitionend -->
```

```
<div class="container">
```

```
  <h1>This is my heading </h1>
```

```
    <p id="para">Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ut rem  
quidem quaerat eos minima, neque sapiente ipsam debitis tempore quae sequi autem  
sunt ea cupiditate? Saepe corporis, laboriosam facere distinctio sint qui unde ipsam  
molestias officiis totam ullam id illum iusto accusantium dicta aspernatur tempore atque  
rerum optio consectetur similique ad culpa veritatis. Dolores atque fuga, dignissimos vel  
velit minus, necessitatibus ipsum culpa recusandae architecto repellendus harum  
voluptatem placeat fugiat blanditiis minima temporibus vero quasi consectetur odio.  
Dignissimos veniam ad recusandae nemo quam quia eligendi. Cumque similique ut  
enim pariatur hic blanditiis reprehenderit maiores quibusdam ratione quisquam beatae  
laborum aperiam magnam iure debitis voluptas, molestiae animi est id eum? Quae  
maxime corrupti asperiores nemo at aperiam minima architecto incidunt necessitatibus.  
Minus explicabo similique quaerat! Tenetur quae amet sint quaerat at ad veniam,  
pariatur similique qui totam beatae ut eos, maiores minus assumenda voluptatem  
voluptas doloremque quia corporis illum eaque nemo ipsum labore alias. Magnam, vel  
quaerat qui excepturi est tenetur ab, esse asperiores porro beatae quasi nobis placeat.  
Natus in perferendis nam vitae, enim odio eveniet animi sunt ut nobis rem velit dicta  
possimus quo obcaecati deserunt assumenda, est molestias quam a. Repudiandae  
ipsam soluta eveniet quia assumenda error autem cum repellat eius nisi. Dolores,  
perferendis, velit nisi omnis dolore voluptatum nostrum quasi, sit odio aut quas quia  
minus rerum accusamus suscipit ex maiores commodi. Ipsum quisquam quo ab modi  
vitae animi eaque!</p>
```

```
</div>
```

```
<button id="btn" onclick="toggleHide()">Show/Hide</button>
```

```
<script>
```

```
  let para = document.getElementById('para');
```

```
  para.addEventListener('mouseover', function run(){
```

```
    console.log('Mouse Inside')
```

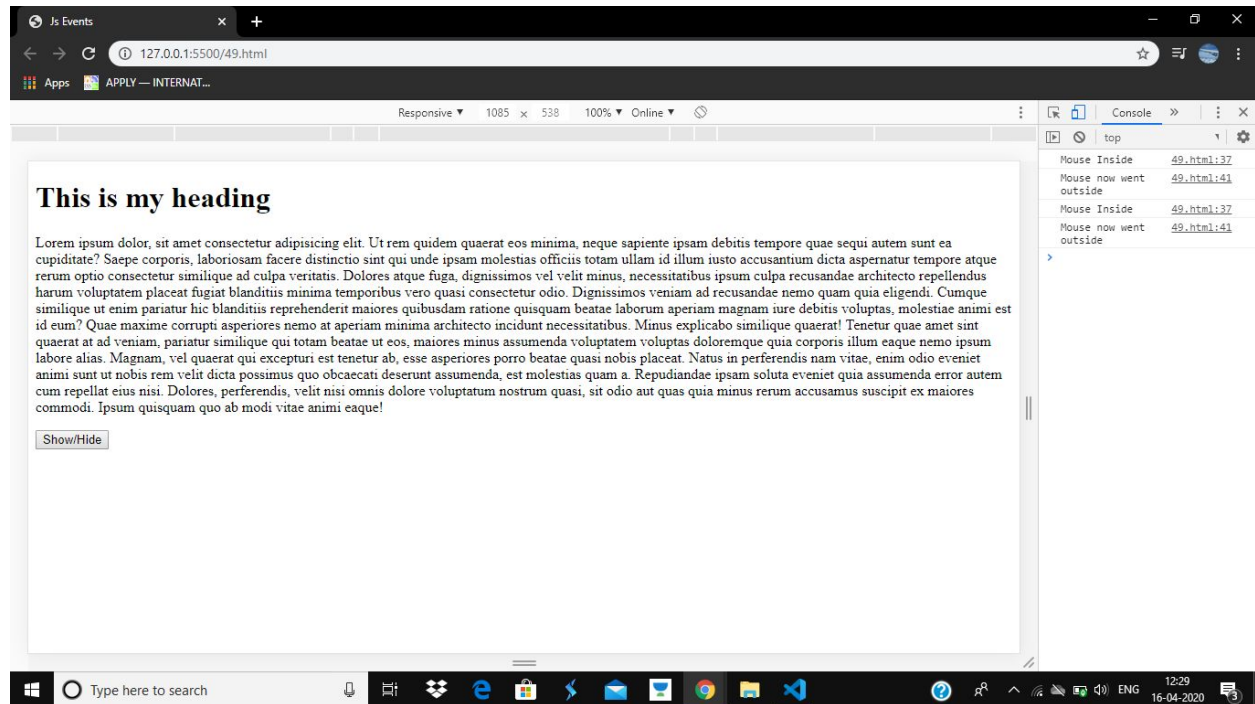
```
  });
```

```
para.addEventListener('mouseout', function run(){
    console.log('Mouse now went outside')
});

function toggleHide(){
    // let btn = document.getElementById('btn');
    let para = document.getElementById('para');
    if(para.style.display != 'none'){
        para.style.display = 'none';
    }
    else{
        para.style.display = 'block';
    }
}
</script>

</body>
</html>
```

Output:-



1.4.13 setInterval & setTimeout

The `setInterval()` :- method calls a function or evaluates an expression at specified intervals (in milliseconds). This will continue calling the function until `clearInterval()` is called, or the window is closed. The ID value returned by `setInterval()` is used as the parameter for the `clearInterval()` method.

1000 ms = 1 second.

Syntax:-

`setInterval(function, milliseconds, param1, param2, ...)`

The `clearInterval()` :- method clears a timer set with the `setInterval()` method. The ID value returned by `setInterval()` is used as the parameter for the `clearInterval()` method.

Example:-

```
myVar = setInterval("javascript function", milliseconds);
clearInterval(myVar);
```

The `setTimeout()` :- method calls a function or evaluates an expression after a specified number of milliseconds. The function is only executed once. If you need to repeat execution, use the `setInterval()` method.

Syntax

`setTimeout(function, milliseconds, param1, param2, ...)`

The `clearTimeout()` :- method clears a timer set with the `setTimeout()` method. The ID value returned by `setTimeout()` is used as the parameter for the `clearTimeout()` method. It prevents the function from running.

Syntax

`clearTimeout(id_of_settimeout)`

Example:-

```
var myVar;

function myFunction() {
  myVar = setTimeout(function(){ alert("Hello"); }, 3000);
}

function myStopFunction() {
  clearTimeout(myVar);
}
```

Source Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>JavaScript SetTimeOut and ClearTimeout</title>
</head>
<body>
```

```
<div class="container">
  Time now is <span id="time"></span>
</div>
<script>
console.log("This is tutorial ewk");
// setTimeout --> Alllows us to run the function once after the interval of time
// clearTimeout --> Alllows us to run the function repeatedly after the interval of time

function greet(name, byeText){
  console.log("Hello Good Morning " + name + " " + byeText);
}
// timeOut = setTimeout(greet, 5000, "Harry", "Take Care");
// console.log(timeOut);

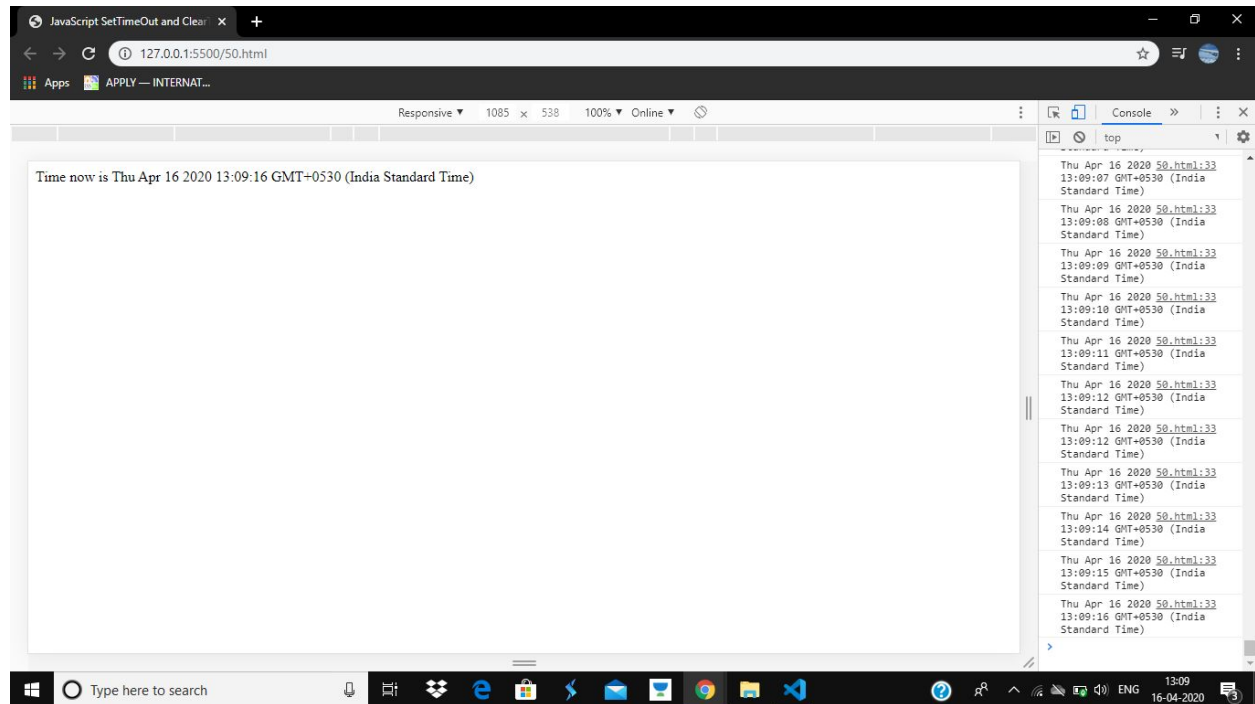
// clearTimeout(timeOut);

// setTimeout(greet(), 12000); --> Wrong as it is calling the function inside setTimeout

// intervalId = setInterval(greet, 1000, "Harry", "Good Night");
// clearInterval(intervalId);

function displayTime(){
  time = new Date();
  console.log(time);
  document.getElementById('time').innerHTML = time;
}
setInterval(displayTime, 1000);
</script>
</body>
</html>
```

Output:-



1.4.14 Date and Time

By default, JavaScript will use the browser's time zone and display a date as a full text string:

Thu Apr 16 2020 14:09:28 GMT+0530 (India Standard Time)

Creating Date Objects :- Date objects are created with the new Date() constructor. There are 4 ways to create a new date object:

```
new Date()  
new Date(year, month, day, hours, minutes, seconds, milliseconds)  
new Date(milliseconds)  
new Date(date string)
```

Date Methods

When a Date object is created, a number of methods allow you to operate on it. Date methods allow you to get and set the year, month, day, hour, minute, second, and millisecond of date objects, using either local time or UTC (universal, or GMT) time.

Source Code:-

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>JavaScript Dates</title>
</head>

<body>
  <div class="container">
    Current time is <span id="time"></span>
  </div>
  <script>
    console.log("This is about date and time ");
    let now = new Date();
    console.log(now);

    let dt = new Date(1000);
    console.log(dt);

    // let newDate = new Date("2029-09-30");
    // console.log(newDate)

    // let newDate = new Date(year, month, date, hours, minutes, seconds,
milliseconds);
    let newDate = new Date(3020, 4, 6, 9, 3, 2, 34);
    console.log(newDate);

    let yr = newDate.getFullYear();
    console.log("The year is ", yr);
```

```
let dat = newDate.getDate();
console.log("The date is ", dat);

let month = newDate.getMonth();
console.log("The month is ", month);

let hours = newDate.getHours();
console.log("The hours is ", hours);

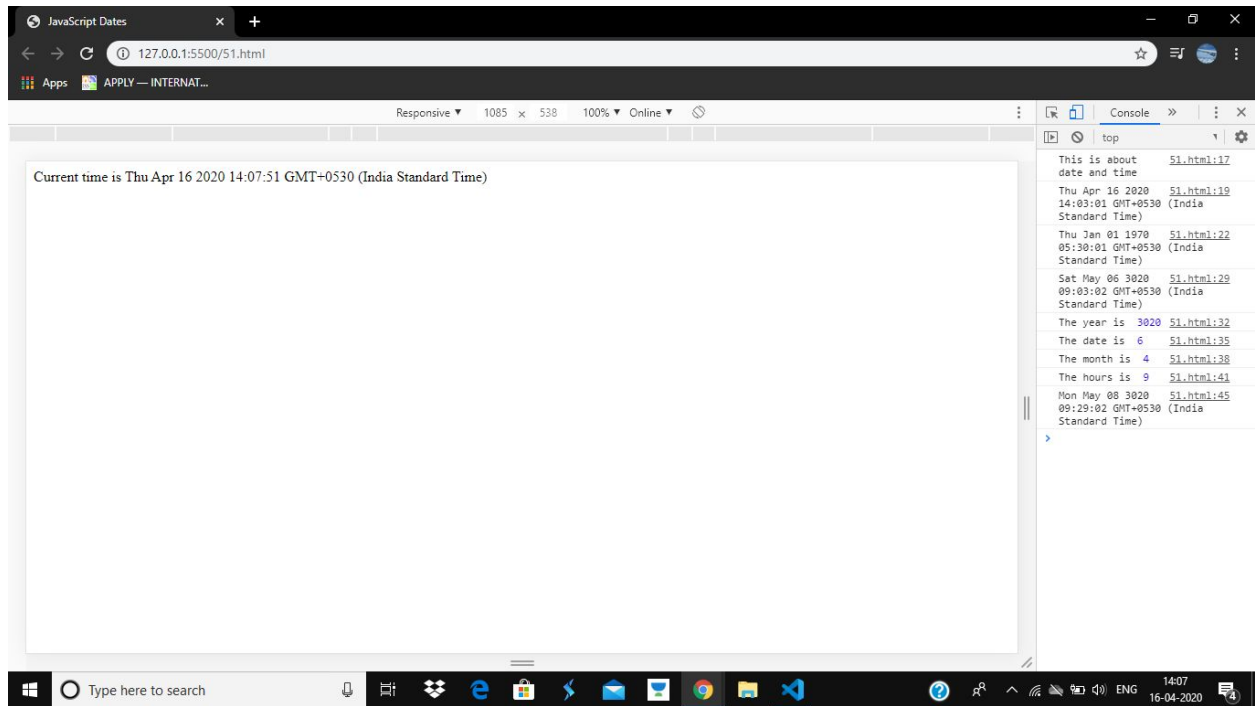
newDate.setDate(8);
newDate.setMinutes(29);
console.log(newDate)
setInterval(updateTime, 1000);

function updateTime() {
    time.innerHTML = new Date();
}

</script>
</body>

</html>
```

Output:-



1.4.15 Arrow Functions

Arrow functions allow us to write shorter function syntax

Without Arrow Function:-

```
hello = function() {  
  return "Hello World!";  
}
```

With Arrow Function:-

```
hello = () => {  
  return "Hello World!";  
}
```

Source Code:-

```
<!DOCTYPE html>  
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Arrow Functions</title>
</head>
<body>
  <div class="container">
    This is arrow functions tutorial.
  </div>
  <script>
    // Arrow function
    // let greet = ()=> {
    //   console.log('Good morning');
    // }

    let greet = () => console.log('Good morning');

    // let sum2 = (a, b)=>{
    //   return a+b;
    // };

    let sum2 = (a, b) => a+b;
    let half = a => a/2;

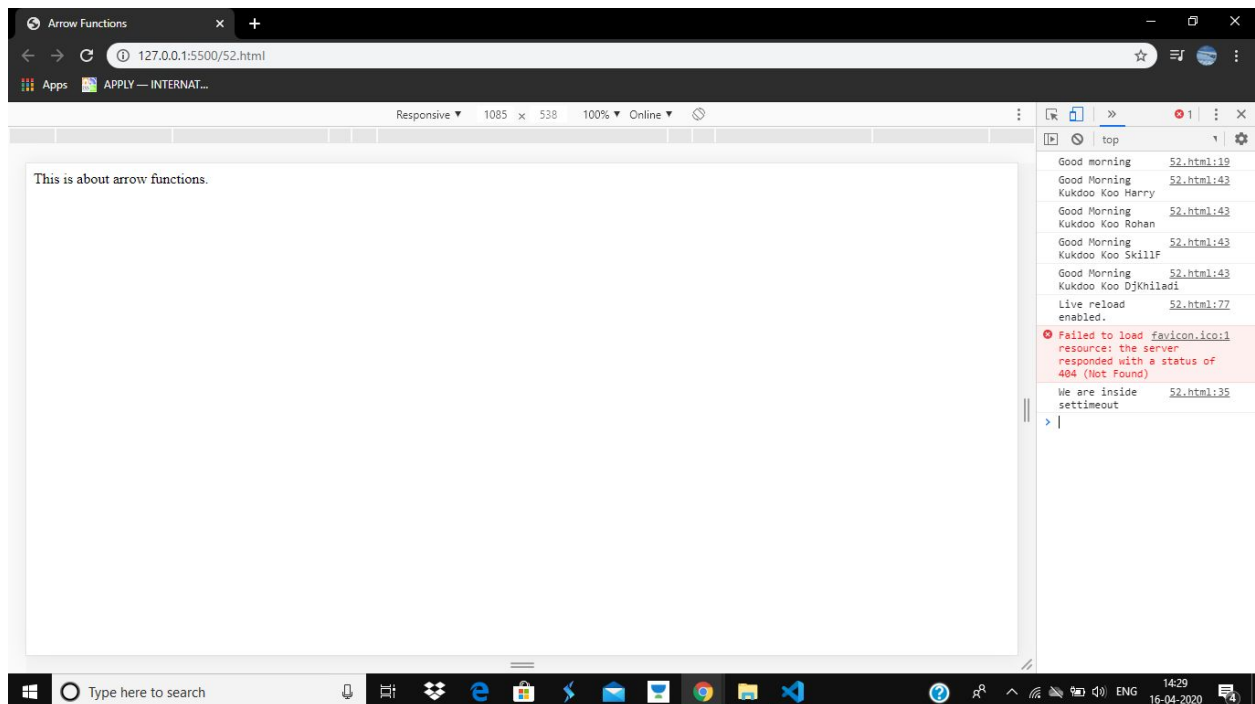
    // function greet() {
    //   console.log('Good morning');
    // }

    greet();
    setTimeout(() => {
      console.log("We are inside settimeout");
    }, 3000);

    let obj1={
      greeting: "Good Morning",
      names: ["Harry", "Rohan", "SkillF", "DjKhiladi"],
      speak(){
```

```
    this.names.forEach((student)=>{
      console.log(this.greeting + " Kukdoo Koo " + student);
    });
  }
}
obj1.speak();
</script>
</body>
</html>
```

Output:-



1.4.16 Math Object

The JavaScript Math object allows you to perform mathematical tasks on numbers.

Math.PI; // returns 3.141592653589793

Math.round() :- Math.round(x) returns the value of x rounded to its nearest integer:

`Math.round(4.7);` // returns 5

`Math.pow()` :- `Math.pow(x, y)` returns the value of x to the power of y:

`Math.pow(8, 2);` // returns 64

`Math.sqrt()` :- `Math.sqrt(x)` returns the square root of x:

`Math.sqrt(64);` // returns 8

`Math.abs()` :- `Math.abs(x)` returns the absolute (positive) value of x:

`Math.abs(-4.7);` // returns 4.7

`Math.ceil()` :- `Math.ceil(x)` returns the value of x rounded up to its nearest integer:

`Math.ceil(4.4);` // returns 5

`Math.floor()` :- `Math.floor(x)` returns the value of x rounded down to its nearest integer:

`Math.floor(4.7);` // returns 4

`Math.sin()` :- `Math.sin(x)` returns the sine (a value between -1 and 1) of the angle x (given in radians). If you want to use degrees instead of radians, you have to convert degrees to radians:

Angle in radians = Angle in degrees x PI / 180.

`Math.sin(90 * Math.PI / 180);` // returns 1 (the sine of 90 degrees)

`Math.min()` and `Math.max()` :- `Math.min()` and `Math.max()` can be used to find the lowest or highest value in a list of arguments:

`Math.min(0, 150, 30, 20, -8, -200);` // returns -200

`Math.max(0, 150, 30, 20, -8, -200);` // returns 150

`Math.random()` :- `Math.random()` returns a random number between 0 (inclusive), and 1 (exclusive):

```
Math.random(); // returns a random number
```

1.4.17 Working with JSON

JSON is a format for storing and transporting data. JSON is often used when data is sent from a server to a web page.

- JSON stands for JavaScript Object Notation
- JSON is a lightweight data interchange format
- JSON is language independent
- JSON is "self-describing" and easy to understand.

This JSON syntax defines an employees object: an array of 3 employee records (objects):

```
{
  "employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
  ]
}
```

The JSON Format Evaluates to JavaScript Objects. It is syntactically identical to the code for creating JavaScript objects. Because of this similarity, a JavaScript program can easily convert JSON data into native JavaScript objects.

JSON Syntax Rules

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

JSON Objects :- JSON objects are written inside curly braces. Just like in JavaScript, objects can contain multiple name/value pairs:


```
{"firstName":"John", "lastName":"Doe"}
```

JSON Arrays :- JSON arrays are written inside square brackets. Just like in JavaScript, an array can contain objects. Each object is a record of a person (with a first name and a last name).

```
"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]
```

Converting a JSON Text to a JavaScript Object

A common use of JSON is to read data from a web server, and display the data in a web page. For simplicity, this can be demonstrated using a string as input.

First, create a JavaScript string containing JSON syntax:

```
var text = '{ "employees" : [' +
  '{ "firstName":"John" , "lastName":"Doe" },' +
  '{ "firstName":"Anna" , "lastName":"Smith" },' +
  '{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

Then, use the JavaScript built-in function `JSON.parse()` to convert the string into a JavaScript object:

```
var obj = JSON.parse(text);
```

Finally, use the new JavaScript object in page:

```
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

Source Code:-

```
<!DOCTYPE html>
<html lang="en">
```

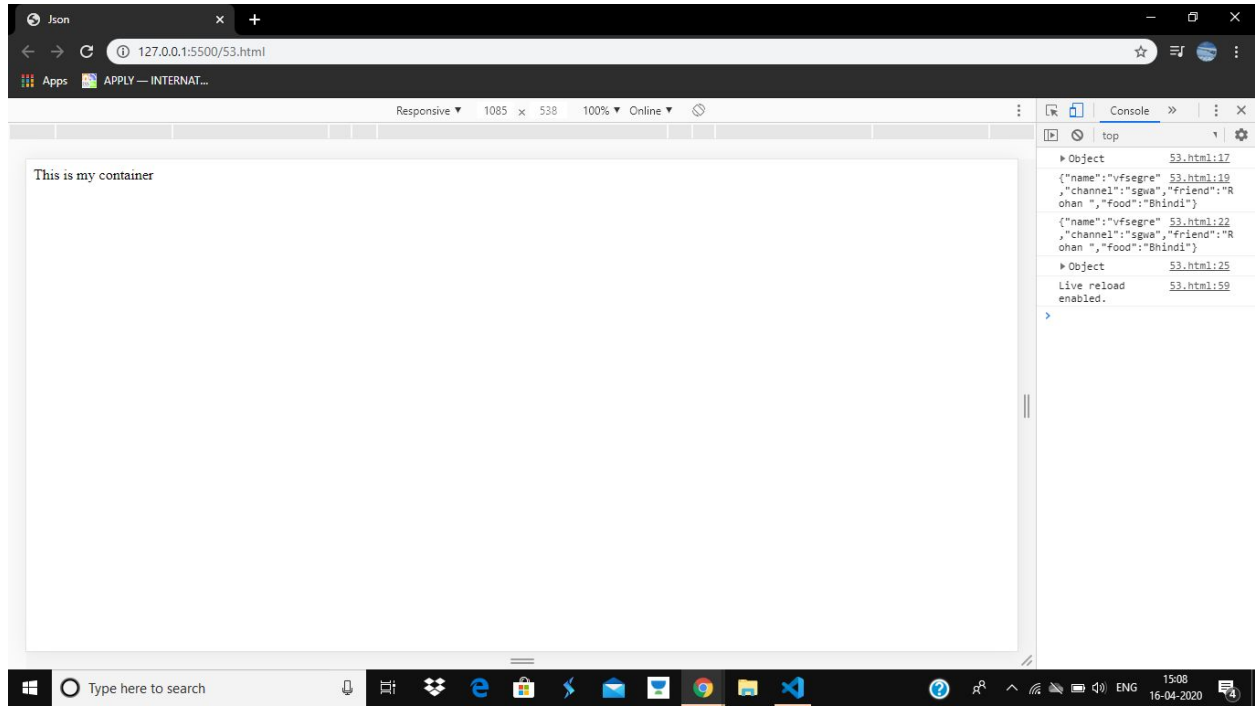
```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Json </title>
</head>
<body>
  <div class="container">This is my container</div>
  <script>
    let jsonObj = {
      name: "vfsegre",
      channel: "sgwa",
      friend: "Rohan ",
      food: "Bhindi"
    }
    console.log(jsonObj);
    let myJsonStr = JSON.stringify(jsonObj);
    console.log(myJsonStr);

    myJsonStr = myJsonStr.replace('Harry', 'Larry');
    console.log(myJsonStr)

    newJsonObj = JSON.parse(myJsonStr);
    console.log(newJsonObj)

  </script>
</body>
</html>
```

Output:-



Chapter 2: BACKEND DEVELOPMENT

1.5 Node.Js

1.5.1 Introduction

- Node.js is an open source server environment
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

A common task for a web server can be to open a file on the server and return the content to the client.

Node.js handles a file request:

1. Sends the task to the computer's file system.
2. Ready to handle the next request.
3. When the file system has opened and read the file, the server returns the content to the client.

Node.js eliminates the waiting, and simply continues with the next request. Node.js runs single-threaded, non-blocking, asynchronous programming, which is very memory efficient.

- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database

Node.js File

- Node.js files contain tasks that will be executed on certain events
- A typical event is someone trying to access a port on the server
- Node.js files must be initiated on the server before having any effect
- Node.js files have extension ".js"

1.5.2 Modules with Examples

We can consider modules to be the same as JavaScript libraries. A set of functions you want to include in your application.

Built-in Modules :- Node.js has a set of built-in modules which you can use without any further installation.

Include Modules :- To include a module, use the `require()` function with the name of the module:

```
var http = require('http');
```

Now the application has access to the HTTP module, and is able to create a server:

```
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.end('Hello World!');  
}).listen(8080);
```

Creating our Own Modules and is easy to include them in our applications.

```
exports.myDateTime = function () {  
  return Date();  
};
```

Including my Own Module

```
var http = require('http');  
var dt = require('./myfirstmodule');  
  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.write("The date and time are currently: " + dt.myDateTime());  
  res.end();  
}).listen(8080);
```

Source Code:-

```
const fs = require("fs");  
let text = fs.readFileSync("a.txt", "utf-8");
```

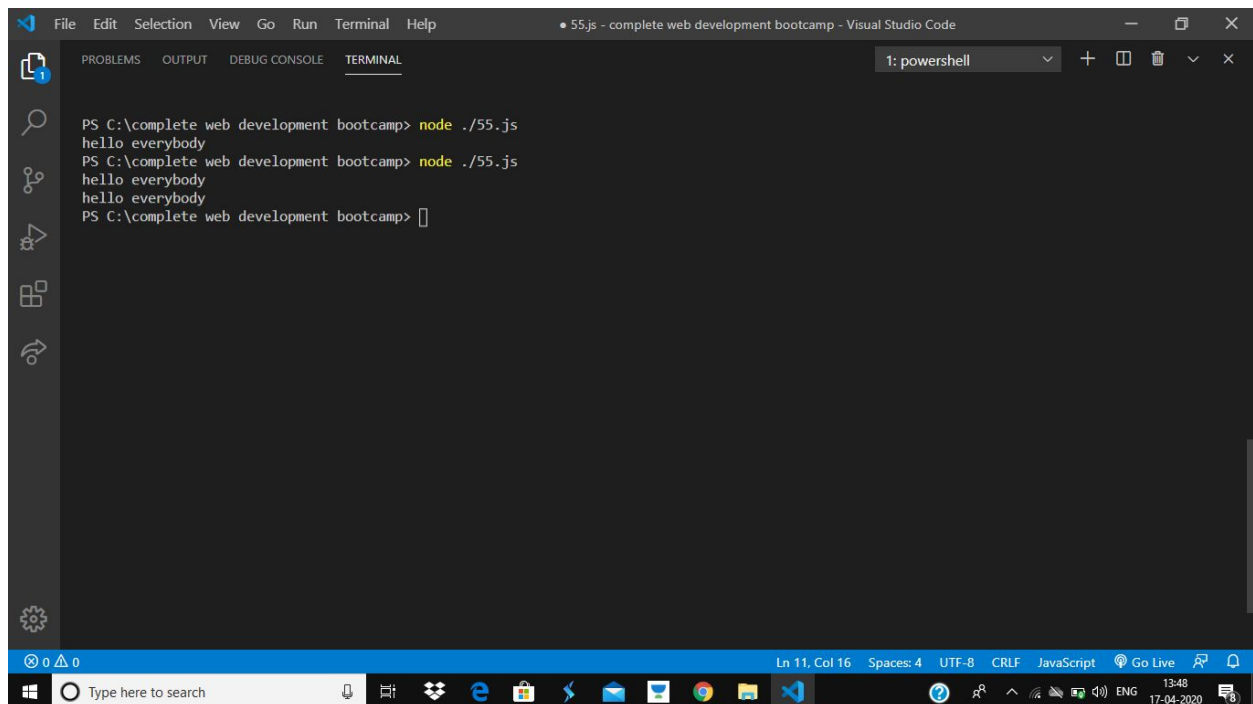
```
text = text.replace("everyone", "everybody");

console.log(text);

fs.writeFileSync("rohan.txt", text);

let a=fs.readFileSync("rohan.txt","utf-8");
console.log(a);
```

Output:-

A screenshot of the Visual Studio Code interface. The terminal window is open, showing the execution of a Node.js script. The command prompt is 'PS C:\complete web development bootcamp> node ./55.js'. The output of the script is 'hello everybody', which is displayed three times, corresponding to three separate executions of the command. The terminal window title is '1: powershell'. The Visual Studio Code status bar at the bottom shows 'Ln 11, Col 16', 'Spaces: 4', 'UTF-8', 'CRLF', 'JavaScript', and 'Go Live'. The Windows taskbar is visible at the bottom of the screen.

1.5.3 Blocking vs Non-Blocking execution

Blocking is when the execution of additional JavaScript in the Node.js process must wait until a non-JavaScript operation completes. This happens because the event loop is unable to continue running JavaScript while a blocking operation is occurring.

In Node.js, JavaScript that exhibits poor performance due to being CPU intensive rather than waiting on a non-JavaScript operation, such as I/O, isn't typically referred to as

blocking. Synchronous methods in the Node.js standard library that use libuv are the most commonly used blocking operations. Native modules may also have blocking methods.

All of the I/O methods in the Node.js standard library provide asynchronous versions, which are non-blocking, and accept callback functions. Some methods also have blocking counterparts, which have names that end with Sync.

JavaScript execution in Node.js is single threaded, so concurrency refers to the event loop's capacity to execute JavaScript callback functions after completing other work. Any code that is expected to run in a concurrent manner must allow the event loop to continue running as non-JavaScript operations, like I/O, are occurring.

As an example, let's consider a case where each request to a web server takes 50ms to complete and 45ms of that 50ms is database I/O that can be done asynchronously. Choosing non-blocking asynchronous operations frees up that 45ms per request to handle other requests. This is a significant difference in capacity just by choosing to use non-blocking methods instead of blocking methods.

Blocking/synchronous execution:-

```
const fs = require('fs');
const data = fs.readFileSync('/file.md'); // blocks here until file is read
```

NonBlocking/Asynchronous execution:-

```
const fs = require('fs');
fs.readFile('/file.md', (err, data) => {
  if (err) throw err;
});
```

Source Code:-

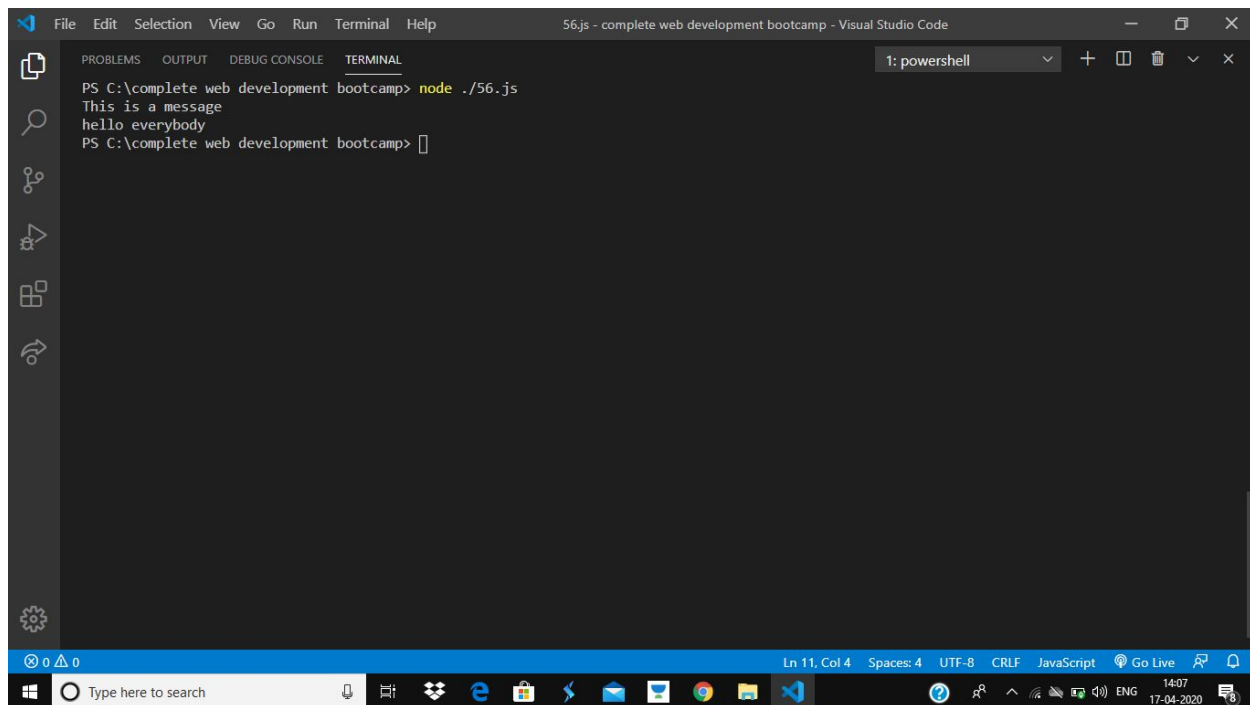
```
// Synchronous or blocking
// - line by line execution

// Asynchronous or non-blocking
// - line by line execution not guaranteed
```

```
// - callbacks will fire

const fs = require("fs");
fs.readFile("rohan.txt", "utf-8", (a, data)=>{
  console.log(data);
});
console.log("This is a message");
```

Output:-

A screenshot of the Visual Studio Code interface. The main editor window shows a file named '56.js' with the following JavaScript code:

```
const fs = require("fs");
fs.readFile("rohan.txt", "utf-8", (a, data)=>{
  console.log(data);
});
console.log("This is a message");
```

 The 'TERMINAL' pane at the bottom shows the command prompt output:

```
PS C:\complete web development bootcamp> node ./56.js
This is a message
hello everybody
PS C:\complete web development bootcamp>
```

 The status bar at the bottom indicates the current file is '56.js', line 11, column 4, with 4 spaces, UTF-8 encoding, CRLF line endings, and JavaScript language mode. The system tray at the very bottom shows the date and time as 17-04-2020, 14:07.

1.5.4 Creating a custom Backend using Node.Js

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.

Sample Code:-

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;
```



```
const server = http.createServer((req, res) => {  
  res.statusCode = 200;  
  res.setHeader('Content-Type', 'text/plain');  
  res.end('Hello World');  
});  
  
server.listen(port, hostname, () => {  
  console.log(`Server running at http://${hostname}:${port}/`);  
});
```

In this case, I have made 4 html pages, such as

home.html

about.html

services.html

contactus.html

And merged them with node.js. Upon calling, each page pops up.

Source Code:-

Node.js code

```
const http = require('http');  
const fs = require('fs');  
  
const hostname = '127.0.0.1';  
const port = 4000;  
const home = fs.readFileSync('home.html')  
const about = fs.readFileSync('about.html')  
const services = fs.readFileSync('services.html')  
const contactus = fs.readFileSync('contactus.html')  
  
const server = http.createServer((req, res) => {  
  console.log(req.url);  
  url = req.url;  
  
  res.statusCode = 200;  
  res.setHeader('Content-Type', 'text/html');
```

```
    if(url == '/'){
        res.end(home);
    }
    else if(url == '/about'){
        res.end(about);
    }
    else if(url == '/services'){
        res.end(services);
    }
    else if(url == '/contactus'){
        res.end(contactus);
    }
    else{
        res.statusCode = 404;
        res.end("<h1>404 not found</h1>");
    }
});

server.listen(port, hostname, () => {
    console.log(`Server running at http://${hostname}:${port}/`);
});
```

Home.html:-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>home</title>
    <style>
        ul{
            display: flex;
            text-decoration: none;
            padding: 29px;
            background-color: burlywood;
            width: auto;
            height: fit-content;
```

```
    }
    li{
      display: flex;
      text-decoration: none;
      padding: 32px 51px 27px 215px;
      margin: -46px;

    }

  </style>
</head>
<body>
  <div>
    <nav>
      <ul>
        <li class="item"><a href ='home'>Home</a></li>
        <li class="item"><a href ='about'>about</a></li>
        <li class="item"><a href ='services'>services</a></li>
        <li class="item"><a href ='contactus'>contactus</a></li>

      </ul>
    </nav>
  </div>
  <div class="container">
    <h1>
      this is home page
    </h1>
  </div>
</body>
</html>
```

about.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>home</title>
<style>
  ul{
    display: flex;
    text-decoration: none;
    padding: 29px;
    background-color: burlywood;
    width: auto;
    height: fit-content;

  }
  li{
    display: flex;
    text-decoration: none;
    padding: 32px 51px 27px 215px;
    margin: -46px;

  }

</style>
</head>
<body>
  <div>
    <nav>
      <ul>
        <li class="item"><a href ='home'>Home</a></li>
        <li class="item"><a href ='about'>about</a></li>
        <li class="item"><a href ='services'>services</a></li>
        <li class="item"><a href ='contactus'>contactus</a></li>

      </ul>
    </nav>
  </div>
  <div class="container">
    <h1>
      this is aboutpage
    </h1>
  </div>
</body>
```

```
</html>
```

services.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>home</title>
  <style>
    ul{
      display: flex;
      text-decoration: none;
      padding: 29px;
      background-color: burlywood;
      width: auto;
      height: fit-content;

    }
    li{
      display: flex;
      text-decoration: none;
      padding: 32px 51px 27px 215px;
      margin: -46px;

    }

  </style>
</head>
<body>
  <div>
    <nav>
      <ul>
        <li class="item"><a href ='home'>Home</a></li>
        <li class="item"><a href ='about'>about</a></li>
        <li class="item"><a href ='services'>services</a></li>
        <li class="item"><a href ='contactus'>contactus</a></li>
```

```
        </ul>
    </nav>
</div>
<div class="container">
    <h1>
        this is servicespage
    </h1>
</div>
</body>
</html>
```

contactus.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>home</title>
    <style>
        ul{
            display: flex;
            text-decoration: none;
            padding: 29px;
            background-color: burlywood;
            width: auto;
            height: fit-content;

        }
        li{
            display: flex;
            text-decoration: none;
            padding: 32px 51px 27px 215px;
            margin: -46px;

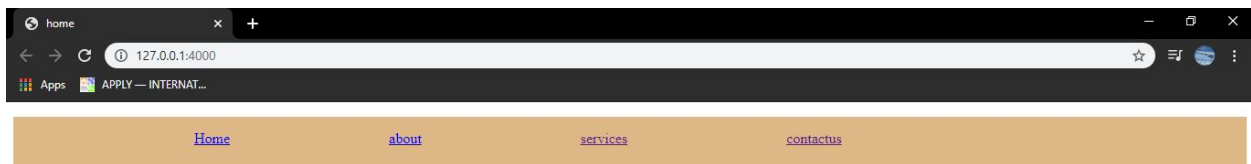
        }

    </style>
</head>
```

```
<body>
  <div>
    <nav>
      <ul>
        <li class="item"><a href ='home'>Home</a></li>
        <li class="item"><a href ='about'>about</a></li>
        <li class="item"><a href ='services'>services</a></li>
        <li class="item"><a href ='contactus'>contactus</a></li>

      </ul>
    </nav>
  </div>
  <div class="container">
    <h1>
      this is contact us page
    </h1>
  </div>
</body>
</html>
```

Output:-



this is home page



After clicking services:-



this is services page



1.5.5 Creating a custom Modules in Node using Node.Js

A module is basically a javascript file that exposes internally scoped functions using the exports variable. The module.exports or exports is a special object which is included in every JS file in the Node.js application by default. module is a variable that represents current module and exports is an object that will be exposed as a module. So, whatever you assign to module.exports or exports, will be exposed as a module.

Message.js

```
module.exports = 'Hello world'; //or exports = 'Hello world';
```

app.js

```
var msg = require('./Messages.js');  
console.log(msg);
```


Export Object :- Exports an object. So, we can attach properties or methods to it.

Message.js

```
exports.SimpleMessage = 'Hello world';
```

app.js

```
var msg = require('./Messages.js');  
console.log(msg.SimpleMessage);
```

Source Code:-

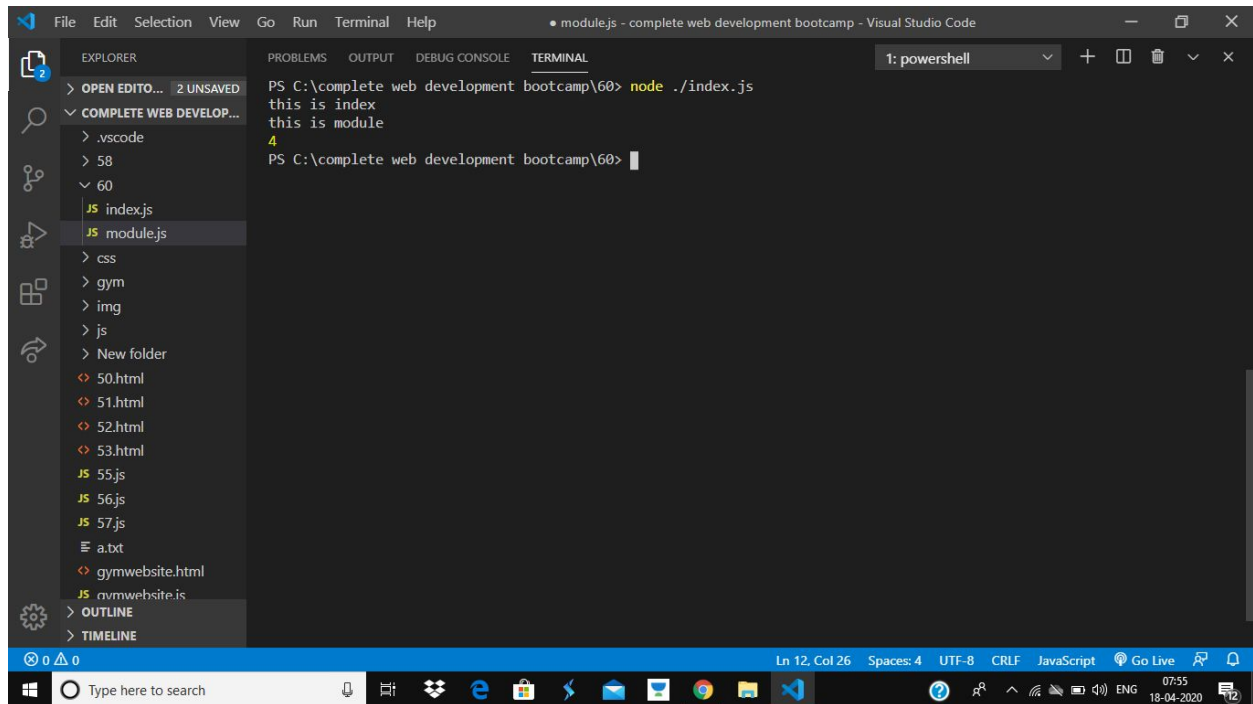
Index.js

```
console.log("this is index");  
const average=require("./module")  
console.log(average([3,4,5]));
```

Module.js

```
console.log("this is module");  
  
function average(arr) {  
  let sum = 0;  
  let av=0;  
  arr.forEach(element => {  
    sum=sum+element;  
  });  
  av=sum/arr.length;  
  return av;  
}  
module.exports = average;
```

Output:-



1.5.6 npm: The Node Package Manager

npm originally short for Node Package Manager is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. The package manager and the registry are managed by npm, Inc.

npm can manage packages that are local dependencies of a particular project, as well as globally-installed JavaScript tools. When used as a dependency manager for a local project, npm can install, in one command, all the dependencies of a project through the package.json file. In the package.json file, each dependency can specify a range of valid versions using the semantic versioning scheme, allowing developers to auto-update their packages while at the same time avoiding unwanted breaking changes. npm also provides version-bumping tools for developers to tag their packages with a particular version. npm also provides the package-lock.json file which has the entry of the exact version used by the project after evaluating semantic versioning in package.json.

To install npm:

```
npm init
```

To install dependencies:

```
Npm install slugify
```

To install devdependencies:

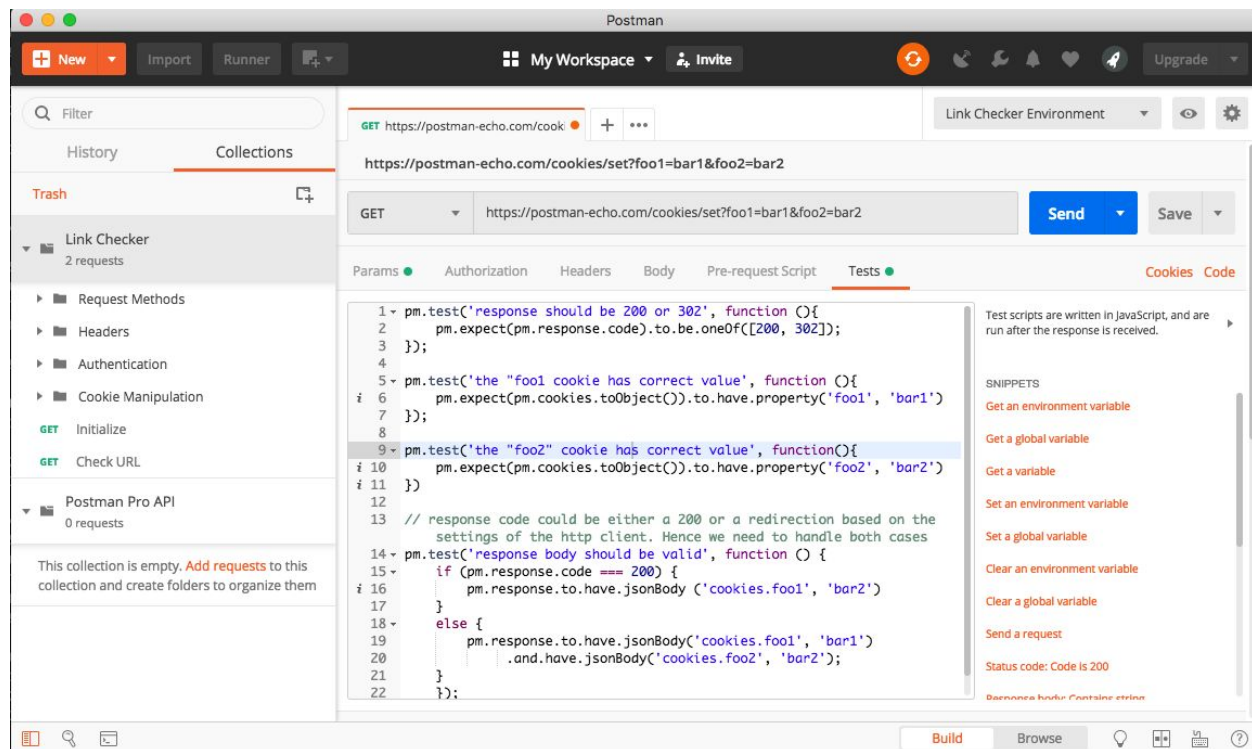
```
npm install nodemon -- globally
```

1.5.7 Express and Postman

Postman:-

Postman contains a powerful runtime based on Node.js that allows you to add dynamic behavior to requests and collections. This allows you to write test suites, build requests that can contain dynamic parameters, pass data between requests, and a lot more. You can add JavaScript code to execute during 2 events in the flow:

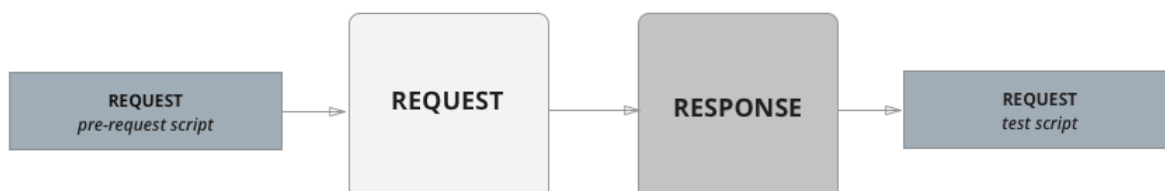
1. Before a request is sent to the server, as a pre-request script under the Pre-request Script tab.
2. After a response is received, as a test script under the Tests tab.



Execution order of scripts

In Postman, the script execution order for a single request looks like this:

- A pre-request script associated with a request will execute before the request is sent
- A test script associated with a request will execute after the request is sent



The Postman Sandbox is a JavaScript execution environment that is available to you while writing prerequisite and test scripts for requests (both in Postman and Newman). Whatever code you write in these sections is executed in this sandbox.

Express:-

Express was initially released in November 2010 and is currently on version 4.17.1 of the API.

Based on the number of high profile companies that use Express, the number of people contributing to the codebase, and the number of people providing both free and paid for support, then yes, *Express* is a popular framework.

Opinionated frameworks are those with opinions about the "right way" to handle any particular task. They often support rapid development *in a particular domain* (solving problems of a particular type) because the right way to do anything is usually well-understood and well-documented. However they can be less flexible at solving problems outside their main domain, and tend to offer fewer choices for what components and approaches they can use.

Unopinionated frameworks, by contrast, have far fewer restrictions on the best way to glue components together to achieve a goal, or even what components should be used. They make it easier for developers to use the most suitable tools to complete a particular task, albeit at the cost that you need to find those components yourself.

Express is unopinionated. We can insert almost any compatible middleware you like into the request handling chain, can structure the app in one file or multiple files, and use any directory structure, may sometimes feel that we have too many choices!

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received the application works out what action is needed based on the URL pattern and possibly associated information contained in POST data or GET data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template.

Express provides methods to specify what function is called for a particular HTTP verb (GET, POST, SET, etc.) and URL pattern ("Route"), and methods to specify what template ("view") engine is used, where template files are located, and what template to use to render a response. You can use Express middleware to add support for cookies, sessions, and users, getting POST/GET parameters, etc.

```
var express = require('express');
var app = express();
app.get('/', function(req, res) {
  res.send('Hello World!');
});
app.listen(3000, function() {
  console.log('Example app listening on port 3000!');
});
```

Importing express:-

```
var express = require('express');
var app = express();
```

Source Code:-

```
const express = require("express");

const app = express();

const port = 80;

app.get("/", (req, res)=>{

  res.send("home page");

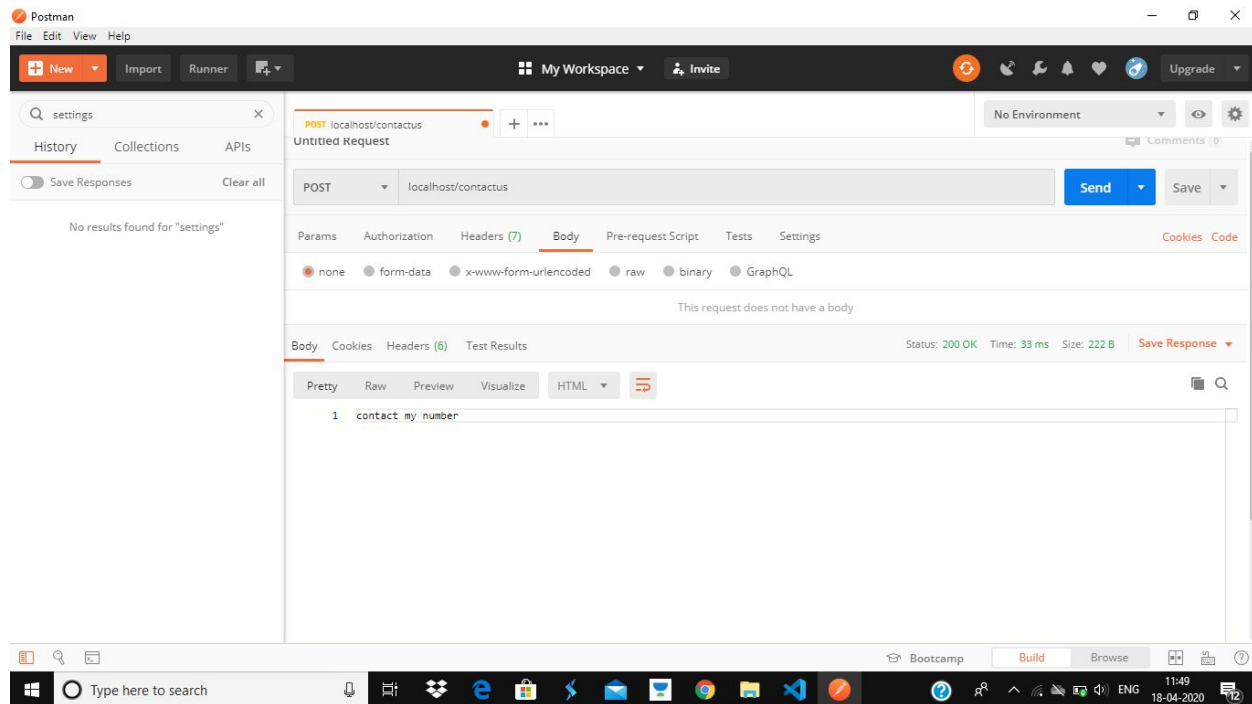
});

app.get("/about", (req, res)=>{

  res.send("about page");
```

```
});  
  
app.get("/services", (req, res)=>{  
  
    res.send("services page");  
  
});  
  
app.get("/contactus", (req, res)=>{  
  
    res.send("contact us page");  
  
});  
  
app.post("/contactus", (req, res)=>{  
  
    res.send("contact my number");  
  
});  
  
app.listen(port, ()=>{  
  
    console.log(` successfull on port ${port}`);  
  
});
```

Postman:-



1.5.8 Pug Template Engine

A template engine enables you to use static template files in your application. At runtime, the template engine replaces variables in a template file with actual values, and transforms the template into an HTML file sent to the client. This approach makes it easier to design an HTML page.

Some popular template engines that work with Express are Pug, Mustache, and EJS. The Express application generator uses Jade as its default, but it also supports several others.

To render template files, set the following application setting properties, set in app.js in the default app created by the generator:

- views, the directory where the template files are located. Eg: app.set('views', './views'). This defaults to the views directory in the application root directory.
- view engine, the template engine to use. For example, to use the Pug template engine: app.set('view engine', 'pug').

After the view engine is set, we don't have to specify the engine or load the template engine module in your app; Express loads the module internally, as shown below (for the above example).

```
app.set('view engine', 'pug')
```

Create a Pug template file named index.pug in the views directory, with the following content:

```
html
  head
    title= title
  body
    h1= message
```

Then create a route to render the index.pug file. If the view engine property is not set, must specify the extension of the view file. Otherwise, you can omit it.

```
app.get('/', function (req, res) {
  res.render('index', { title: 'Hey', message: 'Hello there!' })
})
```

When we make a request to the home page, the index.pug file will be rendered as HTML.

Source Code:-

index.js

```
const express = require("express");
const path=require("path");
const app = express();
const port = 80;
//static files
app.use('static',express.static('static'))
//set template engine as pug
app.set('view engine','pug')
```

```
app.set('views', path.join(__dirname, 'views'))
//pug endpoint
app.get("/demo", (req, res)=>{
    res.render('demo', { title: 'Hey everyone', message: 'Hello world!' })
});

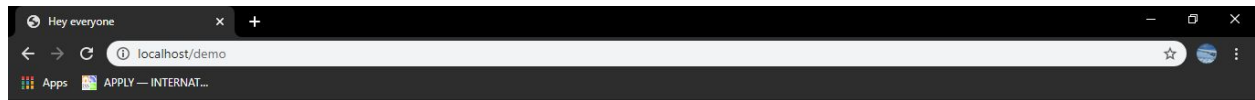
app.get("/", (req, res)=>{
    res.send("home page");
});
app.get("/about", (req, res)=>{
    res.send("about page");
});
app.get("/services", (req, res)=>{
    res.send("services page");
});
app.get("/contactus", (req, res)=>{
    res.send("contact us page");
});
app.post("/contactus", (req, res)=>{
    res.send("contact my number");
});

app.listen(port, ()=>{
    console.log(` successfull on port ${port}`);
});
```

demo.pug

```
html
  head
    title= title
  body
    h1= message
```

Output:-



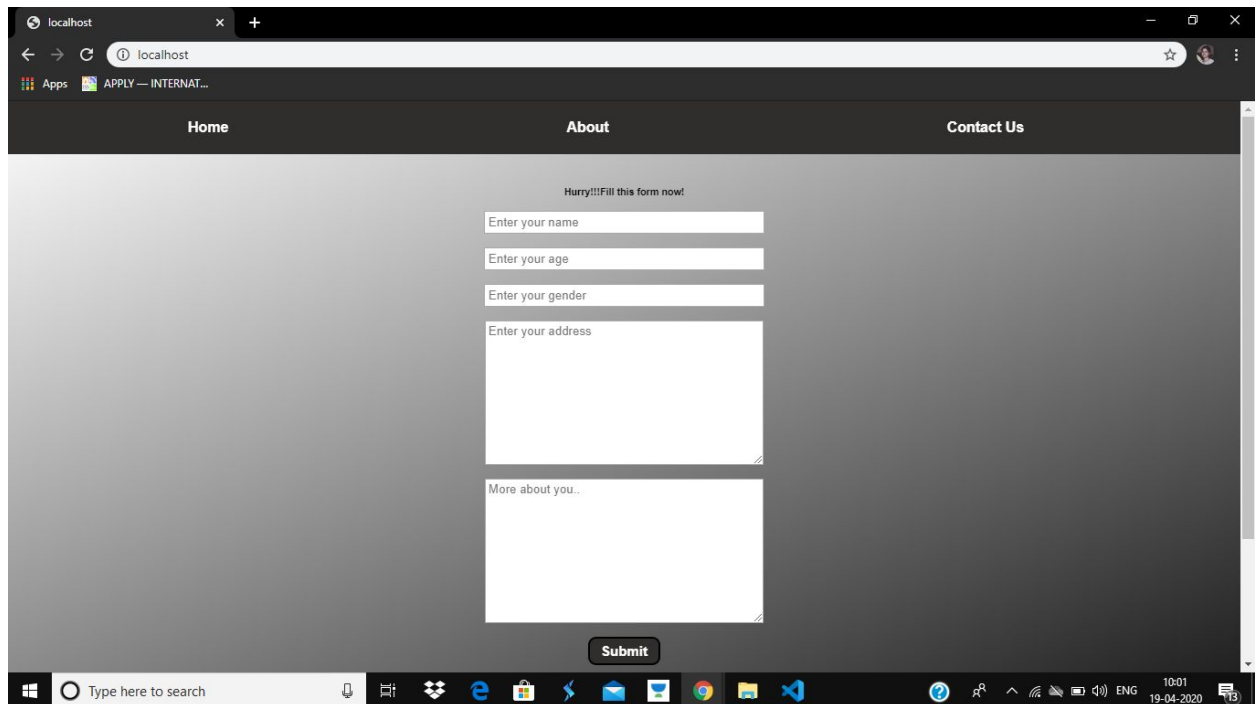
Hello world!



1.5.9 Raw HTML using Pug Template

I have created an HTML Login page using pug template with the help of express and visualised it in localhost:80.

Model of The Template created:



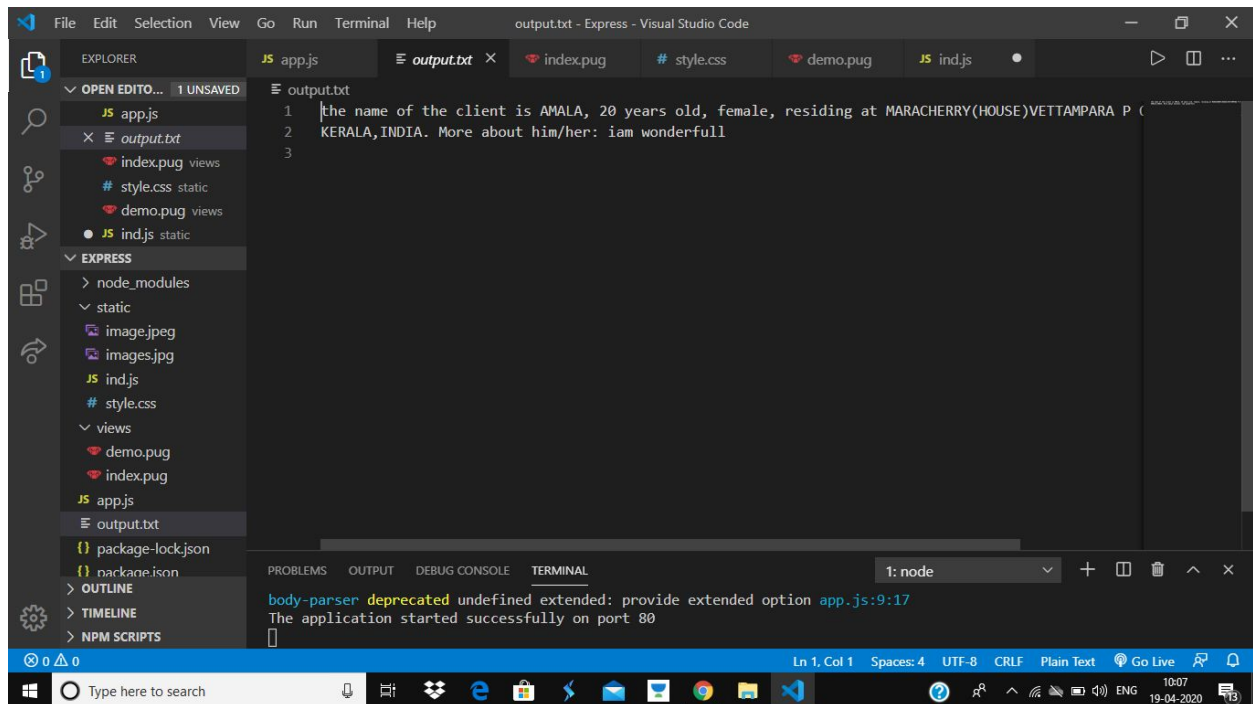
The screenshot shows a web browser window with the address bar set to 'localhost'. The page has a dark header with three navigation links: 'Home', 'About', and 'Contact Us'. The main content area has a light gray background and contains a login form. The form is titled 'Hurry!!!Fill this form now!' and includes the following fields:

- Enter your name
- Enter your age
- Enter your gender
- Enter your address
- More about you...

A 'Submit' button is located at the bottom of the form. The Windows taskbar is visible at the bottom of the screen, showing the search bar and various application icons. The system clock in the bottom right corner indicates the time is 10:01 on 19-04-2020.

After Pressing Submit button all the details entered by you will be recorded in output.txt. For eg:I have entered my details in the login page,

Screenshot of output.txt:



Source code:

app.js

```
const express = require("express");
const path = require("path");
const fs = require("fs");
const app = express();
const port = 80;

// EXPRESS SPECIFIC STUFF
app.use('/static', express.static('static')) // For serving static files
app.use(express.urlencoded())

// PUG SPECIFIC STUFF
app.set('view engine', 'pug') // Set the template engine as pug
```

```
app.set('views', path.join(__dirname, 'views')) // Set the views directory

// ENDPOINTS
app.get('/', (req, res)=>{
  const con = "This is the best content on the internet so far so use it wisely"
  const params = {'title': 'My first Backend Design', "content": con}
  res.status(200).render('index.pug', params);
})

app.post('/', (req, res)=>{
  name = req.body.name
  age = req.body.age
  gender = req.body.gender
  address = req.body.address
  more = req.body.more

  let outputToWrite = `the name of the client is ${name}, ${age} years old, ${gender}, residing at ${address}. More about him/her: ${more}`
  fs.writeFileSync('output.txt', outputToWrite)
  const params = {'message': 'Your form has been submitted successfully'}
  res.status(200).render('index.pug', params);
})

// START THE SERVER
app.listen(port, ()=>{
  console.log(`The application started successfully on port ${port}`);
});
```

index.pug

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <!-- <title>Using plain html in Pug</title> -->
  <title>#{title}</title>

  style
    include ../static/style.css
</head>
<body>
  <nav>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/">About</a></li>
      <li><a href="/">Contact Us</a></li>
    </ul>
  </nav>
  <div class="container">

    <h6>Hurry!!!Fill this form now!</h6>
    <!-- <p>
      | #{content}
      | This is a plain html using pug
    </p> -->

    <form action="/" method="post" id="contact">
      <input type="text" name="name" id="name" placeholder="Enter
your name">
      <input type="text" name="age" id="age" placeholder="Enter your
age">
      <input type="text" name="gender" id="gender"
placeholder="Enter your gender">
      <textarea name="address" id="address" cols="30" rows="10"
placeholder="Enter your address"></textarea>
```

```
        <textarea name="more" id="more" cols="30" rows="10"
placeholder="More about you.."></textarea>
        <button class="btn">Submit</button>
    </form>
</div>
</body>
</html>
```

style.css

```
*{
    font-family: Arial, Helvetica, sans-serif;
    margin: 0;
    padding: 0;
}

body::before{

    content: "";
    width: 100%;
    height: 800px;
    z-index: -1;
    top: 0px;
    position: absolute;
    background: linear-gradient(-23deg, black, transparent);
}

nav{
    background-color: #302d2d;
    padding: 12px
}

ul{
    display: flex;
    flex-direction: row;
}
```



```
li{
  list-style: none;
  padding: 8px 185px;
}

a{
  text-decoration: none;
  color: white;
  font-weight: bold;
}

h2{
  text-align: center;
  color: black;
}

.container{
margin: 35px 0px;
text-align: center;
}

input, textarea{
  display: block;
  padding: 3px;
  margin: 15px auto;
  width: 22%;
}

.btn{
  margin: 4px auto;
  display: block;
  padding: 5px 12px;
  background: #302d2d;
  color: white;
  font-weight: bold;
  font-size: 15px;
```

```
border: 2px solid black;  
border-radius: 9px;  
cursor: pointer;  
outline: none;  
}
```

Chapter 3: DATABASE DESIGNING

1.6 MongoDB

1.6.1 Introduction

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database:-

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection:-

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document:-

A document is a set of key-value pairs. Documents have a dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection

Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)

Why Use MongoDB?

- Document Oriented Storage – Data is stored in the form of JSON style documents.
- Index on any attribute
- Replication and high availability
- Auto-Sharding
- Rich queries
- Fast in-place updates
- Professional support by MongoDB

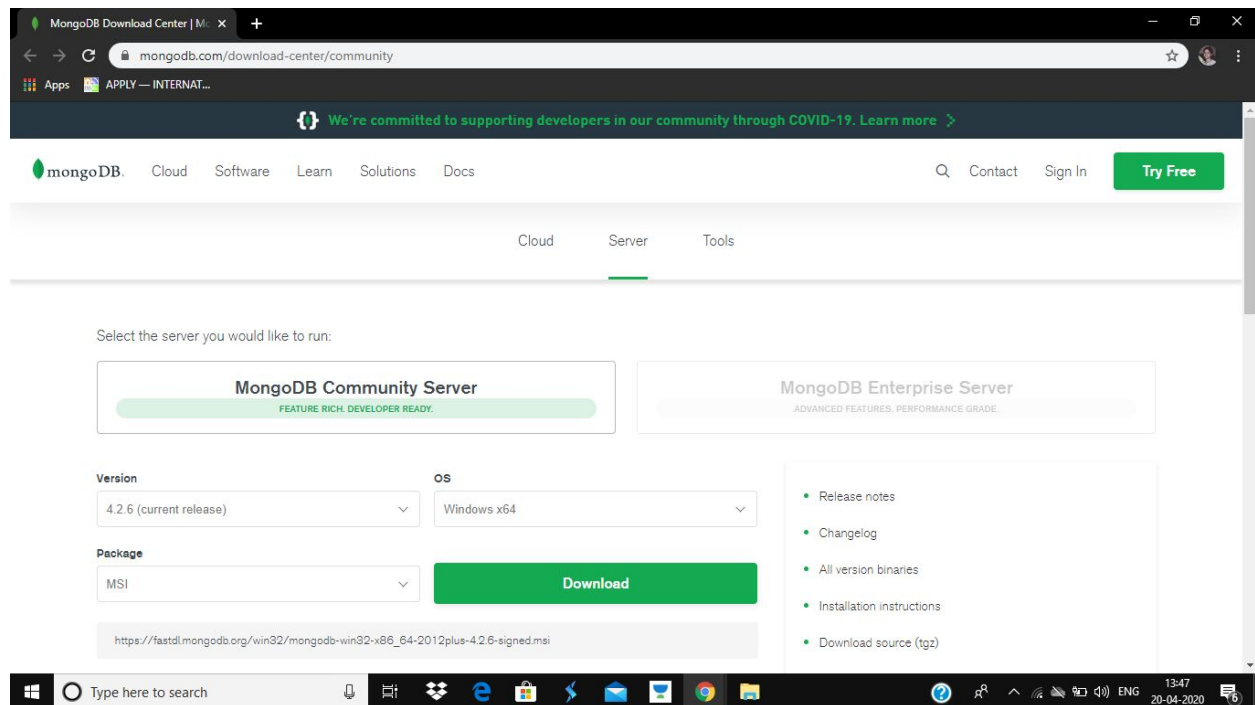
Advantages of MongoDB over RDBMS

- Schema less – MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear.
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Tuning.
- Ease of scale-out – MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

1.6.2 Installation

Install MongoDB On Windows

To install MongoDB on Windows, first download the latest release of MongoDB from <https://www.mongodb.com/download-center>.



Now install the downloaded file, by default, it will be installed in the folder

C:\Program Files\.

MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is c:\data\db. So you need to create this folder using the Command Prompt. Execute the following command sequence.

```
C:\>md data
C:\>md data\db
```

Then you need to specify set the dbpath to the created directory in mongod.exe. For the same, issue the following commands.

In the command prompt, navigate to the bin directory current in the MongoDB installation folder. Suppose my installation folder is C:\Program Files\MongoDB

```
C:\Users\XYZ>d:cd C:\Program Files\MongoDB\Server\4.2\bin
C:\Program Files\MongoDB\Server\4.2\bin>mongod.exe --dbpath "C:\data"
```

This will show a waiting for connection message on the console output, which indicates that the mongod.exe process is running successfully.

Now to run MongoDB, you need to open another command prompt and issue the following command.

```
C:\Program Files\MongoDB\Server\4.2\bin>mongo.exe
MongoDB shell version v4.2.1
connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mong
odb
Implicit session: session { "id" :
UUID("4260beda-f662-4cbe-9bc7-5c1f2242663c") }
MongoDB server version: 4.2.1
>
```

This will show that MongoDB is installed and running successfully.

1.6.3 Inserting Document

Create Database:-

The use Command

Is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

Syntax

Basic syntax of use DATABASE statement is as follows –

```
use DATABASE_NAME
```

Example

If you want to use a database with name <mydb>, then use DATABASE statement would be as follows –

```
>use mydb
switched to db mydb
```

Insert Document:-

The insert() Method

To insert data into a MongoDB collection, you need to use MongoDB's insert() or save() method.

Syntax

The basic syntax of insert() command is as follows –

```
>db.COLLECTION_NAME.insert(document)
```

```
> use amala
switched to db amala
> db.items.insertOne({name: "Samsung 30s", price: 22000, rating: 4.5, qty: 233, sold: 98})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("5e9d5f987e2a5470672853b4")
}
>
> db.items.insertMany([{name: "Samsung 30s", price: 22000, rating: 4.5, qty: 233, sold: 98}, {name: "Moto 30s", price: 29000, rating: 3.5, qty: 133, sold: 598}, {name: "Realme 80s", price: 129000, rating: 2.5, qty: 633, sold: 98, isBig: true}])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("5e9d5f987e2a5470672853b5"),
    ObjectId("5e9d5f987e2a5470672853b6"),
    ObjectId("5e9d5f987e2a5470672853b7")
  ]
}
```

1.6.4 Searching/Querying Data

The find() Method

To query data from the MongoDB collection, you need to use MongoDB's find() method.

Syntax

The basic syntax of find() method is as follows –

```
>db.COLLECTION_NAME.find()
```

find() method will display all the documents in a non-structured way.

Example

Assume we have created a collection named mycol as –

```
> use sampleDB
switched to db sampleDB
> db.createCollection("mycol")
{ "ok" : 1 }
>
```

```
> use amala
switched to db amala
> db.items.find({rating: 3.5})
{ "_id" : ObjectId("5e9d5e92bfff4eafa1acb985"), "name" : "Moto 30s", "price" : 29000, "rating" : 3.5, "qty" : 133, "sold" : 598 }
{ "_id" : ObjectId("5e9d5f987e2a5470672853b6"), "name" : "Moto 30s", "price" : 29000, "rating" : 3.5, "qty" : 133, "sold" : 598 }
> db.items.find({rating: {$gte: 3.5}})
{ "_id" : ObjectId("5e9d5e92bfff4eafa1acb983"), "name" : "Samsung 30s", "price" : 22000, "rating" : 4.5, "qty" : 233, "sold" : 98 }
{ "_id" : ObjectId("5e9d5e92bfff4eafa1acb984"), "name" : "Samsung 30s", "price" : 22000, "rating" : 4.5, "qty" : 233, "sold" : 98 }
{ "_id" : ObjectId("5e9d5e92bfff4eafa1acb985"), "name" : "Moto 30s", "price" : 29000, "rating" : 3.5, "qty" : 133, "sold" : 598 }
{ "_id" : ObjectId("5e9d5f987e2a5470672853b4"), "name" : "Samsung 30s", "price" : 22000, "rating" : 4.5, "qty" : 233, "sold" : 98 }
{ "_id" : ObjectId("5e9d5f987e2a5470672853b5"), "name" : "Samsung 30s", "price" : 22000, "rating" : 4.5, "qty" : 233, "sold" : 98 }
{ "_id" : ObjectId("5e9d5f987e2a5470672853b6"), "name" : "Moto 30s", "price" : 29000, "rating" : 3.5, "qty" : 133, "sold" : 598 }
> db.items.find(
...   $or:[{rating: {$lt: 3.5}}, {price:{$gt: 114000}}]
... )
{ "_id" : ObjectId("5e9d5e92bfff4eafa1acb986"), "name" : "Realme 80s", "price" : 129000, "rating" : 2.5, "qty" : 633, "sold" : 98, "isBig" : true }
{ "_id" : ObjectId("5e9d5f987e2a5470672853b7"), "name" : "Realme 80s", "price" : 129000, "rating" : 2.5, "qty" : 633, "sold" : 98, "isBig" : true }
>
```

1.6.5 Deleting Data

db.collection.deleteOne deletes the first document that matches the filter.

db.collection.deleteMany deletes the set of documents that matches the filter.

```
> use amala
> db.items.deleteOne({price: 22000})
> db.items.find()
> db.items.deleteMany({price: 129000})
> db.items.find()
```

1.6.6 Updating Data

Update() is actually a deprecated feature now.

UpdateOne and updateMany both use update only.

But updateOne() will only update one document within the collection based on The filter.

If you want to update all the documents within the collection based on the filter you need to updateMany().

```
use amala
db.items.find()
db.items.updateOne({name: "Moto 30s"}, {$set: {price: 2}})
db.items.find()
db.items.updateMany({name: "Moto 30s"}, {$set: {price: 3, rating: 1}})
```

1.6.7 MongoDB Compass & Installing Mongoose

MongoDB Compass is a simple-to-use, sophisticated GUI that allows any user within your organization to visualize and explore your data with ad-hoc queries in just a few clicks – all with zero knowledge of the MongoDB query language.

MongoDB Compass is the GUI for MongoDB. Compass allows you to analyze and understand the contents of your data without formal knowledge of MongoDB query syntax. In addition to exploring your data in a visual environment, you can also use Compass to optimize query performance, manage indexes, and implement document validation.

Installing mongoose:-

First be sure you have MongoDB and Node.js installed.

Next install Mongoose from the command line using npm:

```
$ npm install mongoose
```

1.6.8 Using Mongoose in Node.Js

Now say we like fuzzy kittens and want to record every kitten we ever meet in MongoDB. The first thing we need to do is include mongoose in our project and open a connection to the test database on our locally running instance of MongoDB.

```
// getting-started.js
```

```
var mongoose = require('mongoose');
```

```
mongoose.connect('mongodb://localhost/test', {useNewUrlParser: true});
```

We have a pending connection to the test database running on localhost. We now need to get notified if we connect successfully or if a connection error occurs:

```
var db = mongoose.connection;
```

```
db.on('error', console.error.bind(console, 'connection error:'));
```

```
db.once('open', function() {
```

```
    // we're connected!
```

```
});
```

Once our connection opens, our callback will be called. For brevity, let's assume that all following code is within this callback.

With Mongoose, everything is derived from a Schema. Let's get a reference to it and define our kittens.

```
var kittySchema = new mongoose.Schema({
```

```
    name: String
```

```
});
```

So far so good. We've got a schema with one property, name, which will be a String. The next step is compiling our schema into a Model.

```
var Kitten = mongoose.model('Kitten', kittySchema);
```

A model is a class with which we construct documents. In this case, each document will be a kitten with properties and behaviors as declared in our schema. Let's create a kitten document representing the little guy we just met on the sidewalk outside:

```
var silence = new Kitten({ name: 'Silence' });
```

```
console.log(silence.name); // 'Silence'
```

Kittens can meow, so let's take a look at how to add "speak" functionality to our documents:

```
// NOTE: methods must be added to the schema before compiling it with  
mongoose.model()
```

```
kittySchema.methods.speak = function () {
```

```
  var greeting = this.name
```

```
    ? "Meow name is " + this.name
```

```
    : "I don't have a name";
```

```
  console.log(greeting);
```

```
}
```

```
var Kitten = mongoose.model('Kitten', kittySchema);
```

Functions added to the methods property of a schema get compiled into the Model prototype and exposed on each document instance:

```
var fluffy = new Kitten({ name: 'fluffy' });
```

```
fluffy.speak(); // "Meow name is fluffy"
```

We have talking kittens! But we still haven't saved anything to MongoDB. Each document can be saved to the database by calling its save method. The first argument to the callback will be an error if any occurred.

```
fluffy.save(function (err, fluffy) {  
  if (err) return console.error(err);  
  fluffy.speak();  
});
```

Say time goes by and we want to display all the kittens we've seen. We can access all of the kitten documents through our Kitten model.

```
Kitten.find(function (err, kittens) {  
  if (err) return console.error(err);  
  console.log(kittens);  
})
```

CREATION OF FULLY RESPONSIVE WEBSITE USING HTML,CSS,JAVASCRIPT,NODE.JS,MONGODB



WE ASSURE YOU

MORDERN EQUIPMENT

Exercise equipment is getting a streamlined makeover, and some of the best fitness tools out there can even pass as cool sculptures or everyday decor.

GOOD TRAINING

Be consistent. Be diligent. Exert yourself. Don't leave any doubt in your mind that you tried as hard as you could. Most importantly: Trust the system.

HEALTHY DIET PLAN

A healthy eating plan gives your body the nutrients it needs every day while staying within your daily calorie goal for weight loss

THINK BEFORE YOU ACT

Regular exercise helps control weight when use with a balanced diet. Regular physical activity can help you prevent or manage a wide range of health problems and concerns, including stroke, metabolic syndrome, type 2 diabetes, depression, and certain types of cancer, arthritis and falls. Regular training helps reduce stress and improves moods. Regular weight training can improve muscle mass.

About

What is a gym?

A gym - physical exercises and activities performed inside, often using equipment, especially when done as a subject at school. Gymnasium is a large room with equipment for exercising the body and increasing strength or a club where you can go to exercise and keep fit.

A gym is a gymnasium, also known as health club and fitness centre. Gymnasiums have moved away just being a location for gymnastics. Where they had gymnastics apparatus such as bar bells, parallel bars, jumping boards and running path etc.

Choosing the right gym for you

Most gyms have a wide range of ages and levels of fitness. Don't buy into the preconception that it will be full of supreme athletes!

Personal training

Staff members tend to be qualified personal trainers or the club will tend to be able to put you in contact with a personal trainer who will devise a training plan and dietary advice to achieve your goals.

Staff

Members of staff do tend to be fit and healthy, smartly dressed in corporate clothing. They should be happy to explain and demonstrate whenever necessary. They should have knowledge across a wide range of exercise disciplines and many are also personal trainers. When you are on your initial visit look out for if the staff are helping and advising all members or only willing to advise clients who have paid for additional personal training session. You should check on staff qualifications and also view feedback from previous clients. Some memberships come with a specific training schedule designed to achieve your goals.

Services

What can I expect?

Free weights

Gyms have a main work out area which tends to be divided into a free weights section including dumbbells, bar bells. They may also have exercise machines with free weights or more commonly exercise machines will have their own section.

Cardiovascular

The Cardiovascular area will have machines including treadmills (running machines), rowing machines, exercise bikes (stationary).

Classes

Classes are group sessions and separate room from the cardio and free weights. Examples of group class exercise include aerobics, Spin (cycling on a stationary bike), boxercise, high intensity training (HIT) and yoga.

Staff members tend to be qualified personal trainers or the club will tend to be able to put you in contact with a personal trainer who will devise a training plan and dietary advice to achieve your goals.

Facilities

Facilities can vary greatly from gym to gym. Not just on a broad range of cardiovascular equipment including:

Treadmills, Rowers, stationary bikes, cross trainers, versa climbers (step machines), steppers, and rowing machines.

Resistance training equipment including:

Fixed weight machines, dumbbells, barbells and weight plates.

Also clean toilets, changing and shower rooms and exercise mats, stability balls and space to exercise.

When choosing your gym you should check there is a number of the equipment you feel you will use more. If you are going to focus on free weight training (dumbbells etc) check they have multiple sets. On weight machines check they have setting from minimal weight to maximum weight to suit your capabilities. Also check have they been maintained and in good condition. If you are going to mainly use of cardio machines again check they have been well maintained, check they have adequate number of machines compared to gym members.

Larger gyms also may have studio classes, sports and injury therapy. Sports shop, juice bars, cafe and restaurant. Swimming pool, sauna, Jacuzzi, steam rooms. Tennis, squash and badminton courts. Some gyms also offer child care facilities.



GET FIT GYM

[Home](#)[About](#)[Services](#)[Class Info](#)[Registration](#)

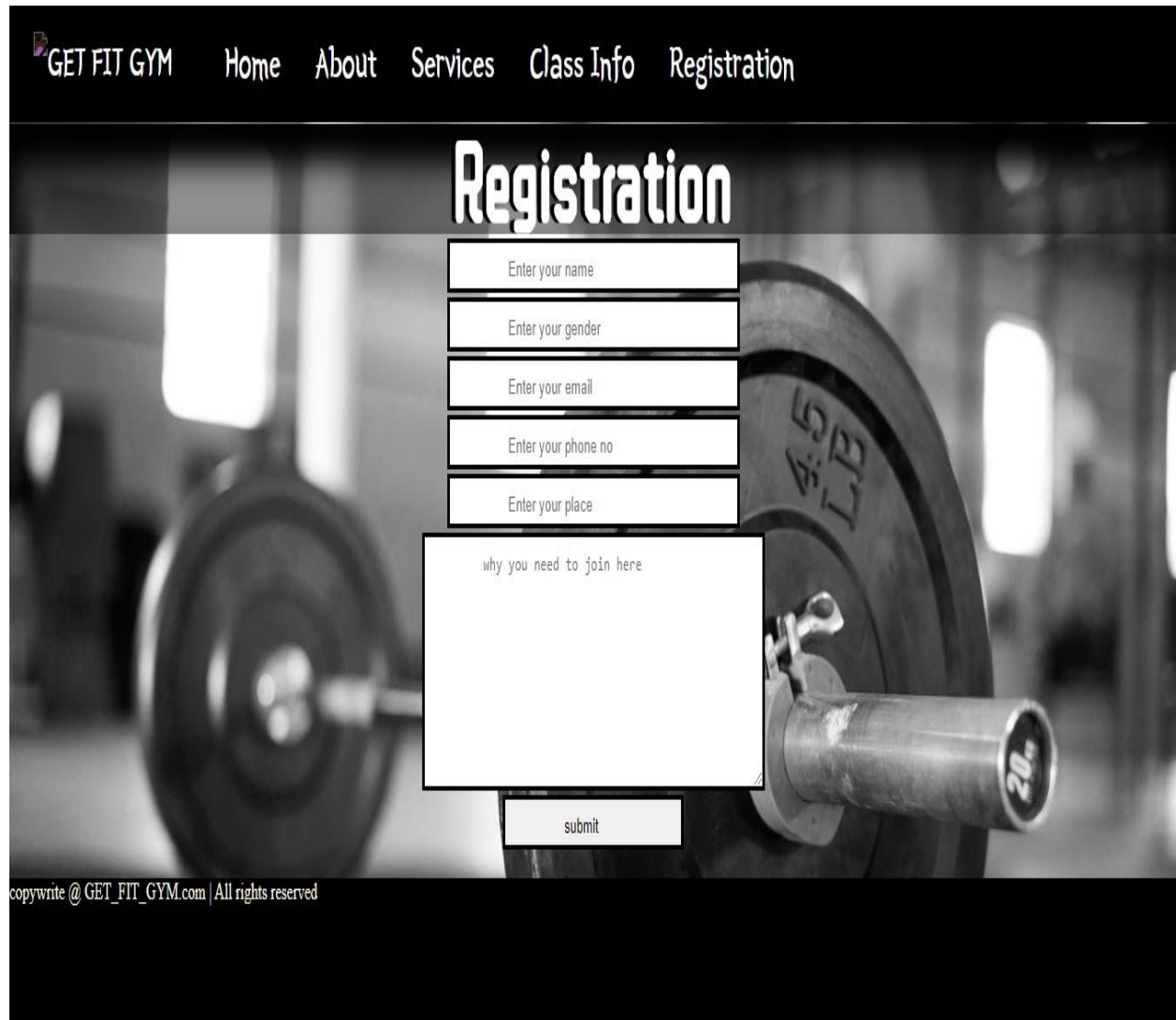
Class Info

Accessibility

You may want to take into account the distance from your home and or work. As the further away it is the less convenient it may become.

Opening hours

Many have different preferences for when they like to train from early birds at 7 and 7 am till late at night when the gyms empty and you have no distractions. Opening hours is again something you may need to consider depending on your other commitments, child care, working hours etc. Also check weekend opening hours. Gym opening and closing times vary greatly with some of the larger open 6am till 11pm. Some gyms are now 24 hours! Also check holiday open hours.



The image shows a registration form for 'GET FIT GYM' overlaid on a background image of a gym. The form is centered and consists of several input fields and a submit button. The background image shows a close-up of a barbell with a 20lb weight plate. The text '45 LB' is visible on the weight plate. The form has a dark header with the gym's name and navigation links. The main title 'Registration' is in a large, bold, white font. The input fields are white with black text prompts. The submit button is a light gray rectangle.

GET FIT GYM Home About Services Class Info Registration

Registration

Enter your name

Enter your gender

Enter your email

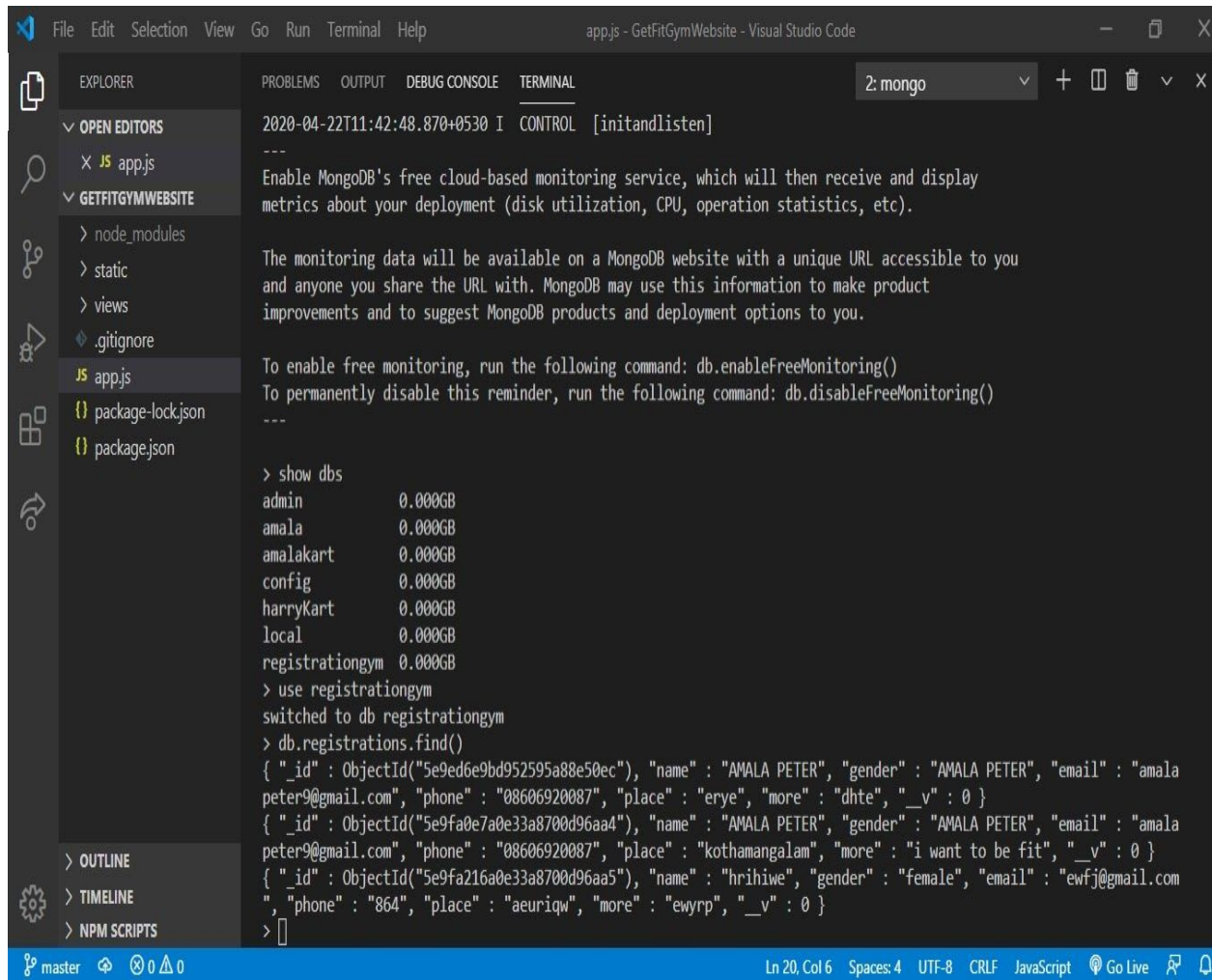
Enter your phone no

Enter your place

why you need to join here

submit

copywrite @ GET_FIT_GYM.com | All rights reserved



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal is running a MongoDB shell session. The left sidebar shows the Explorer view with the file structure of the 'appjs' project. The terminal output shows the following commands and results:

```
2020-04-22T11:42:48.870+0530 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> show dbs
admin          0.000GB
amala          0.000GB
amalakart      0.000GB
config         0.000GB
harryKart      0.000GB
local          0.000GB
registrationgym 0.000GB
> use registrationgym
switched to db registrationgym
> db.registrations.find()
{ "_id" : ObjectId("5e9ed6e9bd952595a88e50ec"), "name" : "AMALA PETER", "gender" : "AMALA PETER", "email" : "amala
peter9@gmail.com", "phone" : "08606920087", "place" : "erye", "more" : "dhate", "__v" : 0 }
{ "_id" : ObjectId("5e9fa0e7a0e33a8700d96aa4"), "name" : "AMALA PETER", "gender" : "AMALA PETER", "email" : "amala
peter9@gmail.com", "phone" : "08606920087", "place" : "kothamangalam", "more" : "i want to be fit", "__v" : 0 }
{ "_id" : ObjectId("5e9fa216a0e33a8700d96aa5"), "name" : "hrihiwe", "gender" : "female", "email" : "ewfj@gmail.com",
"phone" : "864", "place" : "aauriqw", "more" : "ewyrrp", "__v" : 0 }
>
```

The status bar at the bottom indicates the current file is 'appjs' at line 20, column 6, with 4 spaces, UTF-8 encoding, CRLF line endings, and JavaScript language mode.

SOURCE CODE:

app.js

```
const express = require("express");
const path = require("path");
const app = express();
var mongoose = require('mongoose');
const bodyparser = require('body-parser')
mongoose.connect('mongodb://localhost/registrationym', {useNewUrlParser:
true});
const port = 8000;

//define mongoose schema

var registrationSchema = new mongoose.Schema({
  name: String,
  gender: String,
  email: String,
  phone: String,
  place: String,
  more: String
});

var registration = mongoose.model('Registration', registrationSchema);

// EXPRESS SPECIFIC STUFF
app.use('/static', express.static('static')) // For serving static files
app.use(express.urlencoded())

// PUG SPECIFIC STUFF
app.set('view engine', 'pug') // Set the template engine as pug
app.set('views', path.join(__dirname, 'views')) // Set the views directory
```

```
// ENDPOINTS
app.get('/', (req, res)=>{
  const params = { }
  res.status(200).render('home.pug', params);
})

app.get('/registration', (req, res)=>{
  const params = { }
  res.status(200).render('registration.pug', params);
})

app.post('/registration', (req, res)=>{
  var myData = new registration(req.body)
  myData.save().then(()=>{
    res.send("This Item has been saved to the database")
  }).catch(()=>{
    res.status(400).send("Item is not saved to the database")
  });

  //res.status(200).render('registration.pug');
})

app.get('/about', (req, res)=>{
  const params = { }
  res.status(200).render('about.pug', params);
})

app.get('/services', (req, res)=>{
  const params = { }
  res.status(200).render('services.pug', params);
})

app.get('/classinfo', (req, res)=>{
  const params = { }
  res.status(200).render('classinfo.pug', params);
})

// START THE SERVER
app.listen(port, ()=>{
  console.log(`The application started successfully on port ${port}`);
});
```

base.pug

```
doctype html
html
head
  title GET FITNESS GYM
block scripts
block style
body
  nav#navbar
    ul
      div#logo
        img(src="/static/img/mylogo.jpg")
        | GET FIT GYM
      li #[a(href="/") Home]
      li #[a(href="/about") About]
      li #[a(href="/services") Services]
      li #[a(href="/classinfo") Class Info]
      li #[a(href="/registration") Registration]

block content

footer#footer
  | copywrite @ GET_FIT_GYM.com | All rights reserved
```

home.pug

```
extends base.pug

block scripts
  script(src='/static/index.js')

block style
  style
```

```
include ../static/style.css

block content
  section#introSection
    div GET FIT GYM
    div.small Are you READY for The Fitness Challenge!?!??

  section#missionSection
    h2 WE ASSURE YOU
    div.card
      h3 MORDERN EQUIPMENT
      div.card-box
        div.card-content
          p Exercise equipment is getting a streamlined
makeover, and some of the best fitness tools out there can even pass as
cool sculptures or everyday decor.
        div.card
          h3 GOOD TRAINING
          div.card-box
            div.card-content
              p Be consistent. Be diligent. Exert yourself. Don't
leave any doubt in your mind that you tried as hard as you could. Most
importantly: Trust the system.
            div.card
              h3 HEALTHY DIET PLAN
              div.card-box
                div.card-content
                  p A healthy eating plan gives your body the nutrients
it needs every day while staying within your daily calorie goal for weight
loss

    section#sponsorsSection
      h2 THINK BEFORE YOU ACT
      p Regular exercise helps control weight when use with a balanced
diet. Regular physical activity can help you prevent or manage a wide
range of health problems and concerns, including stroke, metabolic
syndrome, type 2 diabetes, depression, and certain types of cancer,
```

arthritis and falls. Regular training helps reduce stress and improves moods. Regular weight training can improve muscle mass.

style.css

```
@import
url('https://fonts.googleapis.com/css?family=Yeon+Sung|ZCOOL+QingKe+HuangY
ou&display=swap');
*{
    margin: 0;
    box-sizing: border-box;
    padding: 0;
}

#logo{
    display: flex;
    color: white;
    align-items: center;
    margin: 0 14px;
    font-size: 28px;
    margin-right: 40px;
}

#logo img{
height: 42px;
    filter: invert(1);
}

#navbar{
    background-color: black;
    padding: 18px 14px;
    font-family: 'Yeon Sung' ;
}

#navbar ul{
```

```
display: flex;
flex-direction: row;
/* justify-content: center; */
align-items: center;
}

#navbar li{
    list-style: none;
}

#navbar li a{
    color: white;
    font-size: 30px;
    text-decoration: none;
    padding: 13px 20px;
}

#navbar li a:hover{
    color: darkgrey;
}

#introSection{
    padding-bottom: 40px;
    color: white;
    display: flex;
    flex-direction: column;
    justify-content: center;
    font-family: 'ZCOOL QingKe HuangYou';
    font-size: 70px;
    font-weight: bolder;
    align-items: center;
    background: center/cover no-repeat url("/static/mybg.jpg") ;
    height: 425px;
    background-color: red;
    text-shadow: 20px -19px 8px black;
}
```



```
#introSection div.small{
    font-size: 52px;
    padding: 5px;
}

#missionSection{
    height: 326px;;
    width: 67%;
    margin: auto;
}

#missionSection h2{
    font-size: 36px;
    font-family: 'Yeon Sung' ;
    text-align: center;
    padding-top: 12px;
    text-shadow: darkgrey;
}

.card{
    display: inline-block;
    border: 2px solid black;
    border-radius: 10px;
    width: 28%;
    margin: 23px 20px;
    height: 209px;
}

.card h3{
    text-align: center;
    margin: 0 12px;
    margin-top: 24px;
    font-family: Arial, Helvetica, sans-serif;
    text-shadow: darkgrey;
}
```

```
.card-box{
  display: flex;
  align-items: center;
  justify-content: space-between;
  font-family: Arial, Helvetica, sans-serif;
}

.card-content{
  padding: 0 10px;
  font-family: Arial, Helvetica, sans-serif;
  text-align: justify;
}

.card-content p{
  font-size: 19px;
  font-family: Arial, Helvetica, sans-serif;
}

#sponsorsSection{
  height: 300px;
  background-color: darkgrey;
}

#sponsorsSection h2{
  font-size: 36px;
  font-family: 'Yeon Sung' ;
  text-align: center;
  padding-top: 12px;
}

#sponsorsSection p{
  font-size: 36px;
  font-family: 'Yeon Sung' ;
```

```
text-align: center;
padding-top: 12px;
font-family: Arial, Helvetica, sans-serif;
text-align: justify;
}

#footer{

    color: beige;
    background-color: black;
    height: 100px;

}
```

about.pug

```
extends base.pug

block scripts
  script(src='/static/index.js')

block style
  style
    include ../static/style.css
    include ../static/styleAbout.css

block content
  div.container
    h1 About
    h3 What is a gym?
    p A gym - physical exercises and activities performed inside, often using equipment, especially when done as a subject at school. Gymnasium is a large room with equipment for exercising the body and increasing strength or a club where you can go to exercise and keep fit.
```

A gym is a gymnasium, also known as health club and fitness centre. Gymnasiums have moved away just being a location for gymnastics. Where they had gymnastics apparatus such as bar bells, parallel bars, jumping boards and running path etc.

Choosing the right gym for you

Most gyms have a wide range of ages and levels of fitness. Don't buy into the preconception that it will be full of supreme athletes!

Personal training

Staff members tend to be qualified personal trainers or the club will tend to be able to put you in contact with a personal trainer who will devise a training plan and dietary advice to achieve your goals.

Staff

Members of staff do tend to be fit and healthy, smartly dressed in corporate clothing. They should be happy to explain and demonstrate whenever necessary. They should have knowledge across a wide range of exercise disciplines and many are also personal trainers. When you are on your initial visit look out for if the staff are helping and advising all members or only willing to advise clients who have paid for additional personal training session. You should check on staff qualifications and also view feedback from previous clients. Some memberships come with a specific training schedule designed to achieve your goals.

style>About.css

```
@import
url('https://fonts.googleapis.com/css?family=Yeon+Sung|ZCOOL+QingKe+HuangY
ou&display=swap');
```

```
h1{
  text-align: center;
  font-family: 'ZCOOL QingKe HuangYou';
  margin:1px 0;
  color:white;
  font-size: 73px;
  background-color:#1916167a;
  box-shadow: inset 0px 20px 20px 6px black;
  text-shadow: -5px 0px 0px black;
}
h3{
  text-align: left;
  font-family: 'ZCOOL QingKe HuangYou';
  margin:57px 57px;
  color:black;
  font-size: 30px;
}
.container p{
  margin: 22px 57px;
}
}
```

services.pug

```
extends base.pug

block scripts
  script(src='/static/index.js')

block style
  style
    include ../static/style.css
    include ../static/styleServices.css
```

```
block content
```

```
div.container
```

```
h1 Services
```

```
h3 What can I expect?
```

```
p Free weights
```

p Gyms have a main work out area which tends to be divided into a free weights section including dumbbells, bar bells. They may also have exercise machines with free weights or more commonly exercise machines will have their own section.

```
h3 Cardiovascular
```

p The Cardiovascular area with have machines including treadmills (running machines), rowing machines, exercise bikes (stationary).

```
h3 Classes
```

p Classes are group sessions and separate room from the cardio and free weights. Examples of group class exercise include aerobics, Spin (cycling on a stationary bike), boxercise, high intensity training (HIT) and yoga.

```
h3 Personal training
```

p Staff members tend to be qualified personal trainers or the club will tend to be able to put you in contact with a personal trainer who will devise a training plan and dietary advice to achieve your goals.

```
h3 Facilities
```

p Facilities can vary greatly from gym to gym. Not just on a broad range of cardiovascular equipment including:

p Treadmills, Rowers, stationary bikes, cross trainers, versa climbers (step machines), steppers, and rowing machines.

```
h3 Resistance training equipment including:
```

p Fixed weight machines, dumbbells, barbells and weight plates.

p Also clean toilets, changing and shower rooms and exercise mats, stability balls and space to exercise.

p When choosing your gym you should check there is a number of the equipment you feel you will use more. If you are going to focus on free weight training (dumbbells etc) check they have multiple sets. On weight machines check they have setting from minimal weight to maximum weight to suit your capabilities. Also check have they been maintained and in good condition. If you are going to mainly use of cardio machines again check they have been well maintained, check they have adequate number of machines compared to gym members.

p Larger gyms also may have studio classes, sports and injury therapy. Sports shop, juice bars, cafe and restaurant. Swimming pool, sauna, Jacuzzi, steam rooms. Tennis, squash and badminton courts. Some gyms also offer child care facilities.

styleServices.css

```
@import
url('https://fonts.googleapis.com/css?family=Yeon+Sung|ZCOOL+QingKe+HuangY
ou&display=swap');

h1{
  text-align: center;
  font-family: 'ZCOOL QingKe HuangYou';
  margin:1px 0;
  color:white;
  font-size: 73px;
  background-color:#1916167a;
  box-shadow: inset 0px 20px 20px 6px black;
  text-shadow: -5px 0px 0px black;
```

```
}  
h3{  
  text-align: left;  
  font-family: 'ZCOOL QingKe HuangYou';  
  margin:57px 57px;  
  color:black;  
  font-size: 30px;  
}  
.container p{  
  margin: 22px 57px;  
}  
}
```

classinfo.pug

```
extends base.pug  
  
block scripts  
  script(src='/static/index.js')  
  
block style  
  style  
    include ../static/style.css  
    include ../static/styleClassinfo.css  
  
block content  
  div.container  
    h1 Class Info  
    h3 Accessibility  
      p You may want to take into account the distance from your home  
and or work. As the further away it is the less convenient it may become.  
  
    h3 Opening hours  
  
      p Many have different preferences for when they like to train from  
early birds at 7 and 7 am till late at night when the gyms empty and you
```


have no distractions. Opening hours is again something you may need to consider depending on your other commitments, child care, working hours etc. Also check weekend opening hours. Gym opening and closing times vary greatly with some of the larger open 6am till 11pm. Some gyms are now 24 hours! Also check holiday open hours.

styleClassinfo.css

```
@import
url('https://fonts.googleapis.com/css?family=Yeon+Sung|ZCOOL+QingKe+HuangY
ou&display=swap');

h1{
    text-align: center;
    font-family: 'ZCOOL QingKe HuangYou';
    margin:1px 0;
    color:white;
    font-size: 73px;
    background-color:#1916167a;
    box-shadow: inset 0px 20px 20px 6px black;
    text-shadow: -5px 0px 0px black;
}
h3{
    text-align: left;
    font-family: 'ZCOOL QingKe HuangYou';
    margin:57px 57px;
    color:black;
    font-size: 30px;
}
.container p{
    margin: 22px 57px;
}
```

registration.pug

```
extends base.pug

block scripts
  script(src='/static/index.js')

block style
  style
    include ../static/style.css
    include ../static/styleRegistration.css

block content
  div.container
    h1 Registration
    form(action="/registration",method="post",class="myform")
      input(type="text",class="myinput",name="name",placeholder="Enter your name")

      input(type="text",class="myinput",name="gender",placeholder="Enter your gender")

      input(type="email",class="myinput",name="email",placeholder="Enter your email")

      input(type="phone",class="myinput",name="phone",placeholder="Enter your phone no")

      input(type="text",class="myinput",name="place",placeholder="Enter your place")

      textarea(class="myinput",cols="30",rows="10",name="more",placeholder="why you need to join here")
      button(class="btn",type="submit")
        | submit
```

styleRegistration.css

```
@import
url('https://fonts.googleapis.com/css?family=Yeon+Sung|ZCOOL+QingKe+HuangY
ou&display=swap');
body{
    background-image: url("/static/mybg.jpg");
    background-repeat: no-repeat;
    background-size: cover;
    z-index: 0;
}
h1{
    text-align: center;
    font-family: 'ZCOOL QingKe HuangYou';
    margin:1px 0;
    color:white;
    font-size: 73px;
    background-color:#1916167a;
    box-shadow: inset 0px 20px 20px 6px black;
    text-shadow: -5px 0px 0px black;
}
.myform{
}
.container{
    height:80vh;
}
.myinput{
    display: block;
    margin: 3px auto;
    padding: 10px 95px 5px 68px;
    border: 3px solid black;
}
.btn{
    display:block;
    margin: 3px auto;
    padding: 10px 95px 5px 68px;
```

```
border: 3px solid black;  
cursor: pointer;  
  
}
```

REFERENCES

Frontend development:

javascript

www.w3schools.com

www.wikipedia.com

www.bootstrap.com

Backend Development:

Node.Js

www.w3schools.com

www.wikipedia.com

www.bootstrap.com

Database Design

www.postman.com

www.express.com

www.mongodb.com

www.mongoose.com

Web Hosting

www.digitalocean.com