Node.js

Software’s requirement

1. Node.js
2. VS Code

Javascript is used to develop websites that adds interactivity to achieve dynamic behaviour.

Javascript fundamentals

1. Functions
2. Variables
3. Operators
4. Loops
5. Conditional Statements
6. Objects
7. Arrays

Variables: You can use let or const keywords to create variables, avoid using var keyword

var x = 20; // not recommended

let y = 30; // block scoped, but you can modify let variables

const z = 40; // block scoped, but you can’t modify const variables

Functions: In Javascript you can create functions using function keyword

function getUsers() {

…..

}

function multiply(a, b) {  
 return a \* b;   
}

multiply(20, 30);

Arrays, forEach & map



forEach: It is only to iterate

map: it converts an array to another array.

Handling events in Javascript

DOM (Document Object Model) generates some events when you interact with the HTML element, then you can perform some task for those events which is called as event handling, you can call some functions based on the events.

document.getElementById(“id”): This is used to access an HTML element based on the id.

document.getElementsByTagName(“tag”): This is used to access all the elements having same tag name, it returns array of elements

document.getElementsByClassName(“class”): This is used to access all the elements having same class name, it returns array of elements

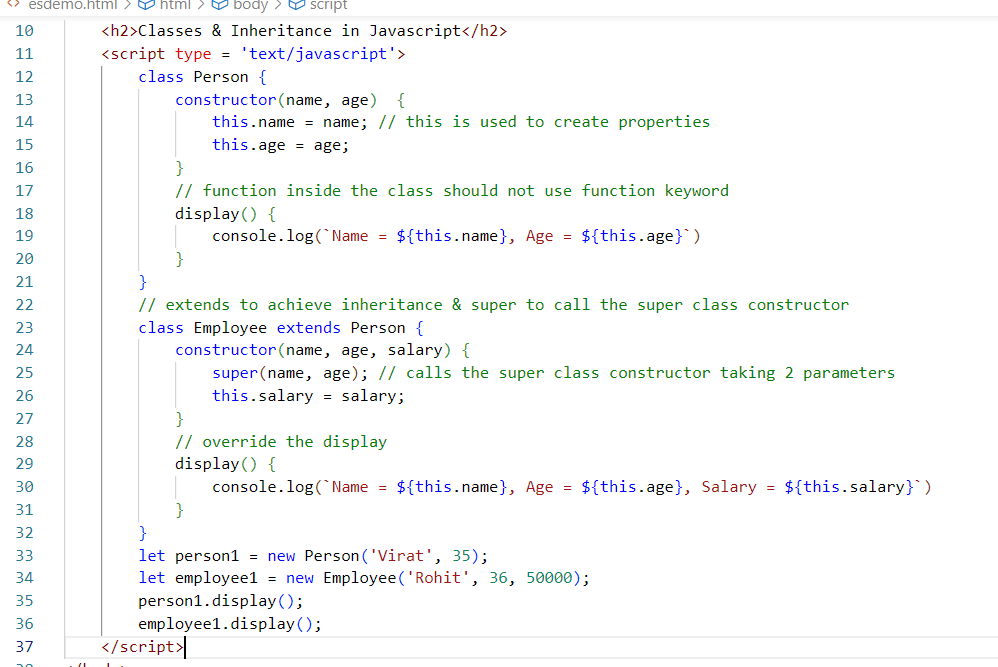
document.querySelector(“id or class or tag name”): This is also used to access the elements having either an id or class name or tag name

New features of Javascript

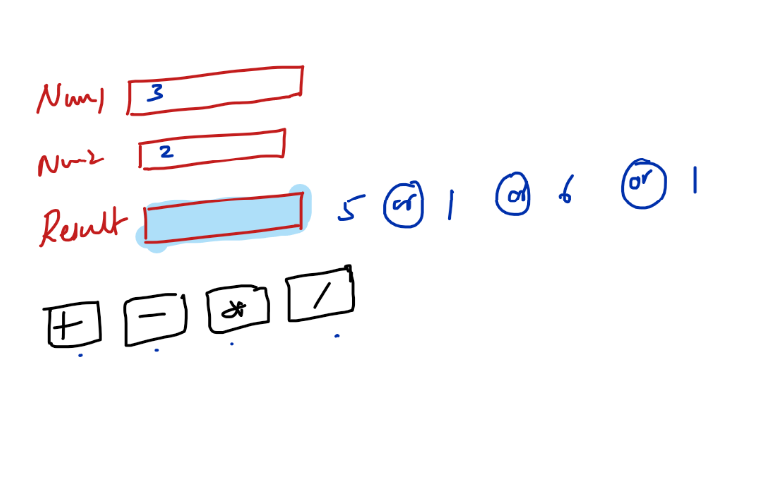
ECMA Script is the standard to specify the features for the Javascript, in ES6 they introduced lot of changes in the syntax to simplify the javascript writing

1. let, const keywords
2. class, super, extends keywords
3. arrow function – simplified form of writing the callbacks
4. template strings – back tick with ${expression}
5. rest & spread operators – to work with multiple parameters in an easy way
6. generators – to get multiple values from the function without completing the function execution
7. optional chain
8. exponential operators

Classes & Inheritance in new feature



Activity



Arrow functions

It simplifies writing callback functions by removing lot of things in the callback like function keyword, { } if its one line statement, return statement if its one line statement

Callback:

function(a, b) {   
 return a + b;  
}

Arrow function

(a, b) => a + b

(or)

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

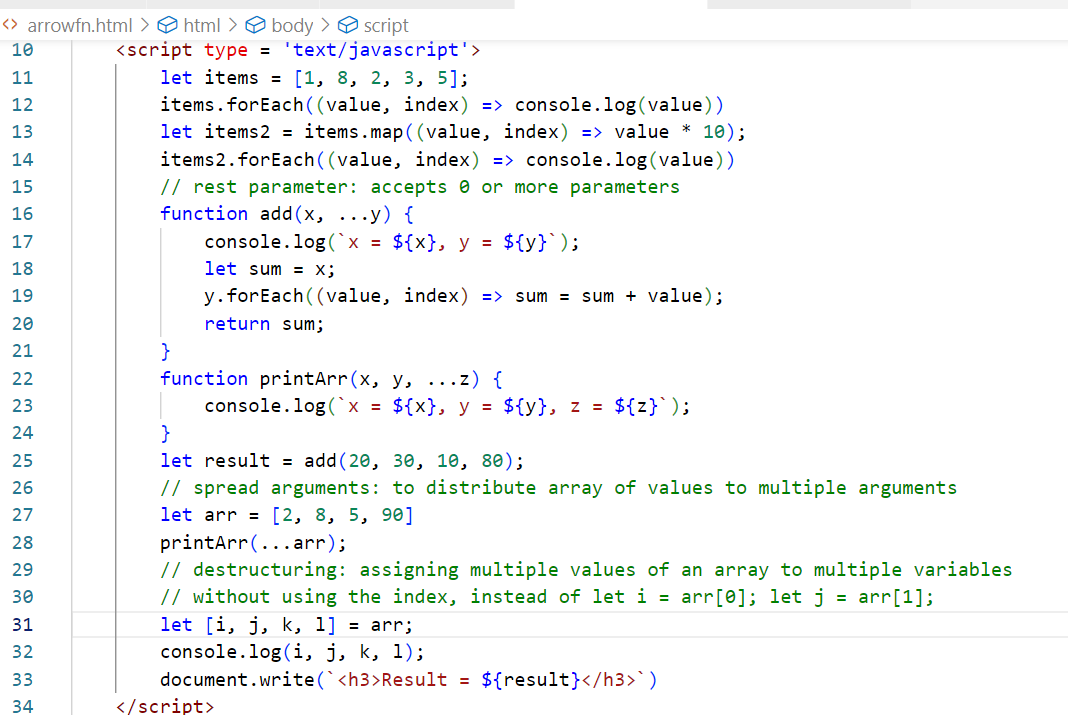
Using in forEach or map function

let items = [2, 1, 5, 4, 3];

items.forEach( (value, index) => console.log(value); }

let items2 = items.map( ( value, index) => value \* 10);

Rest, Spread & Destructuring



Rest: to pass 0 or more values

Spread: to distribute multiple values to multiple parameters

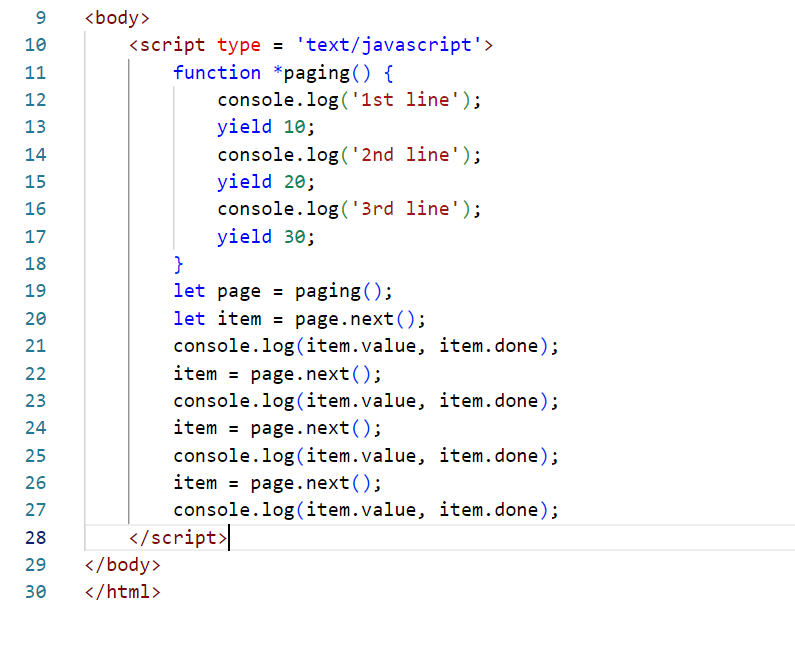
Destructuring: to assign values to multiple variables in a single line

Generators:

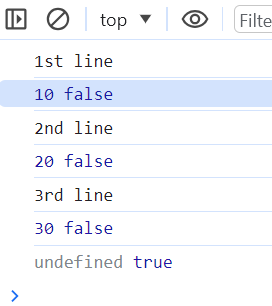
They can return multiple values on a single function call on installment

function \*foo() {  
 calling backend   
 yield data;  
 calling backend  
 yield data;  
 calling backend  
 yield data;  
}

let f = foo();  
f.next();  
f.next();  
f.next();



Output:



Optional chain

It is to access the nested property in an easier without checking for undefined condition.



Exponential operator

2 \* 3 = 6

2 \*\* 3 = 8

2 \*\* 4 = 16

Asynchronous actions

It initiates the action now but finishes later

setTimeout(callback, timing);

Ajax calls using XMLHttpRequest : call the backend api’s

let xhr = new XMLHttpRequest();

xhr.send( “api/posts” );

xhr.onreadystatechange = function() {   
 // updates the document when the response comes  
}

Promise: It is used to perform an asynchronous operation whose result could be either successful or failure

promise  
 .then( callback )  
 .catch( callback )

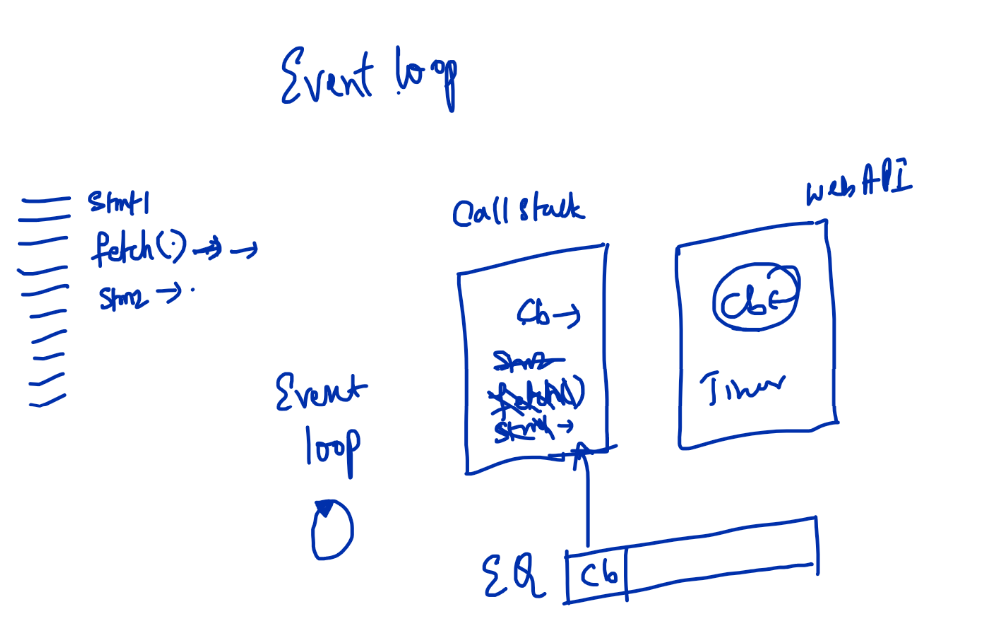
Node.js

* It provides a runtime environment to run the javascript without using browser settings
* It can be used as a tool for any kind of project

Benefits of Node.js

1. It helps to create any kind of project
2. It provides an HTTP environment so that you don’t need any external servers to host a website, earlier developers used external servers like apache tomcat, jboss, weblogic
3. Front-end developers can easily switch to write backend programs using node.js without learning a new programming language
4. It can interact with various databases like SQL & NoSQL
5. It provides NPM (Node Package Manager) from which you can download third party javascript libraries
6. NPM also provides libraries for many popular front-end technologies like React.js, Angular, Vue.js
7. Node.js can be used to develop a Fullstack application using a common language Javascript, popular fullstacks like MEAN, MERN, MEVN all use Node.js as an environment
   1. MEAN – MongoDB/MySQL, Express, Angular, Node
   2. MERN – MongoDB/MySQL, Express, React, Node
   3. MEVN – MongoDB/MySQL, Express, Vue.js, Node
8. Node.js uses Non-Blocking asynchronous callbacks to perform IO operations like reading from the network, accessing a database or a file system, instead of blocking the thread and wasting CPU cycles waiting.

Event Loop architecture



Step1: Main code is pushed to the callstack

Step2: stmt1 is pushed to the callstack & will be cleared

Step3: fetch() is pushed to the callstack & will be cleared, however it will have a callback that is handled in the WebAPI

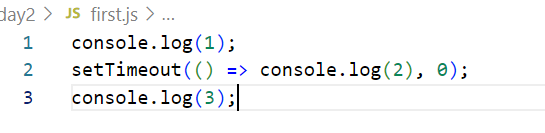
Step4: stmt2() is pushed to the callstack & will be cleared

Step5: If the asynchronous operation is completed then it will be added to the event queue and will be pushed in FIFO basis

Note: Node.js is different than the browser, because it executes javascript in the backend, you can’t use any inbuilt objects of browsers in node.js like, document, window, alert, sessionStorage, localStorage

To run the javascript you can use : node filename.js

first.js



Here the callstack executes all the 3 lines one by one, then the callback of setTimeout is added to the event queue & pushed to the callstack for execution, because of that we see callback function console.log() later.

REPL: Read Eval Print & Loop: It is a CLI provided by node.js to run some scripts to test the code, Mongodb uses this REPL in its terminal to enable admins to directly interact with the database using javascript functions

package.json: It is an heart of node.js, it contains the entire application configurations like dependencies, commands, versions and so on.

npm init: This creates package.json

npm init -fy: This also create package.json, but it doesn’t ask information of your project

Node.js modules

Modules are reusable code that you can use in any javascript file, you can reusable objects, functions, classes, variables by importing & exporting, in Node.js we have 3 types of modules

1. Local module: Which is created within the project
2. Core module: These are part of Node.js
3. Third party module: These are not part of Node.js or Local module, they are downloaded from the internet.

Local Modules: These are some variables or classes or functions which you can export in one file & import in another file

a.js

module.exports.add = function(x, y) {   
 return x + y;  
}  
module.exports.sub = function(x, y) {   
 return x – y;  
}

b.js

let add = require(‘./a.js’).add;

let sub = require(‘./a.js’).sub;

add(20, 30);

sub(30, 40);

The above code in a.js exports add & sub, and b.js imports them, however the syntax is little bit complex because of old javascript syntax, we can use export & import in an easier way but we need to add type : module in package.json, then we can write a.js & b.js as

a.js

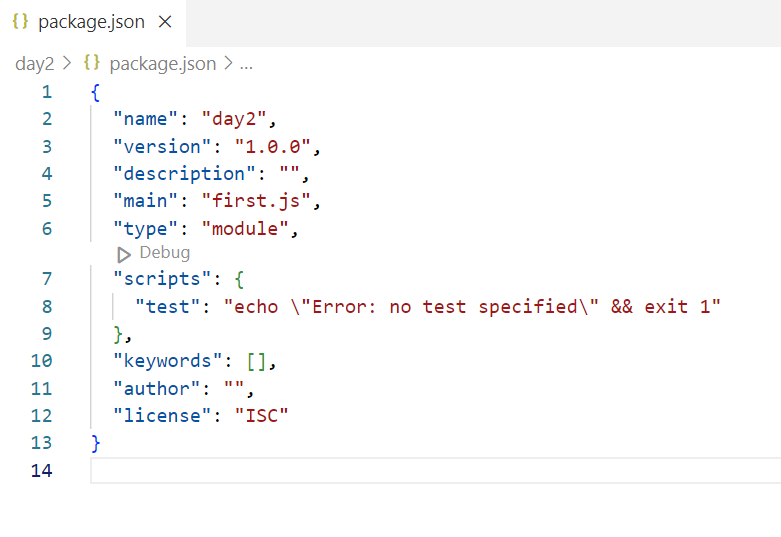
export function add(x, y) {   
 return x + y;   
}  
export function sub(x, y) {   
 return x – y ;   
}

b.js

import { add, sub } from ‘./a.js’;

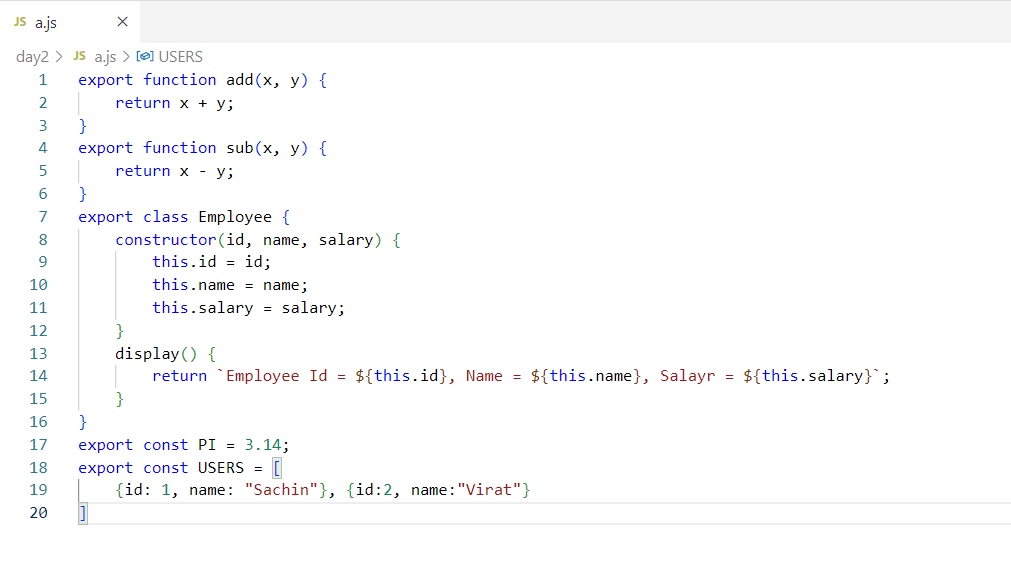
// aliasing: if we have same function in 2 or js file

import { add as a} from ‘./a.js’;  
import { add as b } from ‘./x.js’;  
a(); // calls add of a.js  
b(); // calls add of b.js



“type”:”module”: to make use of newer import & export syntax.

a.js: to export functions, variables, arrays & classes



Core Modules: In node.js there are many core modules which you can import, like

* os: It provides platform details like architecture, platform, version and so on
* http: It is to create server
* fs: It is to interact with the files to read/write data
* url: It is to query the url string when you have a client & server programs

OS Module



Third party module

These modules we need to install using npm

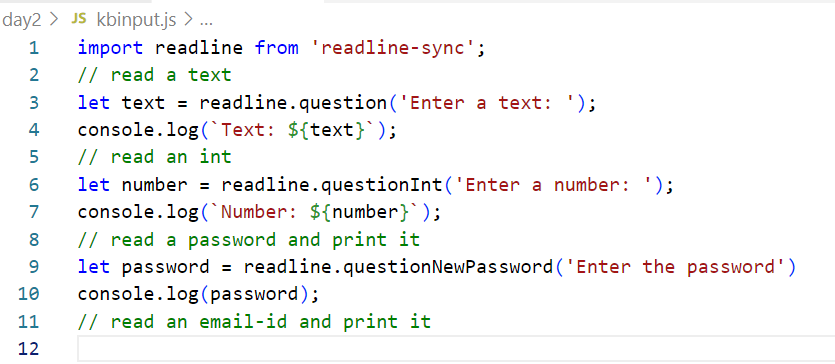
readline-sync: It is to take user input in a terminal based application

npm install readline-sync --save

or

npm i readline-sync

This gives various methods to read int, String, password, email and so on.



Array methods:

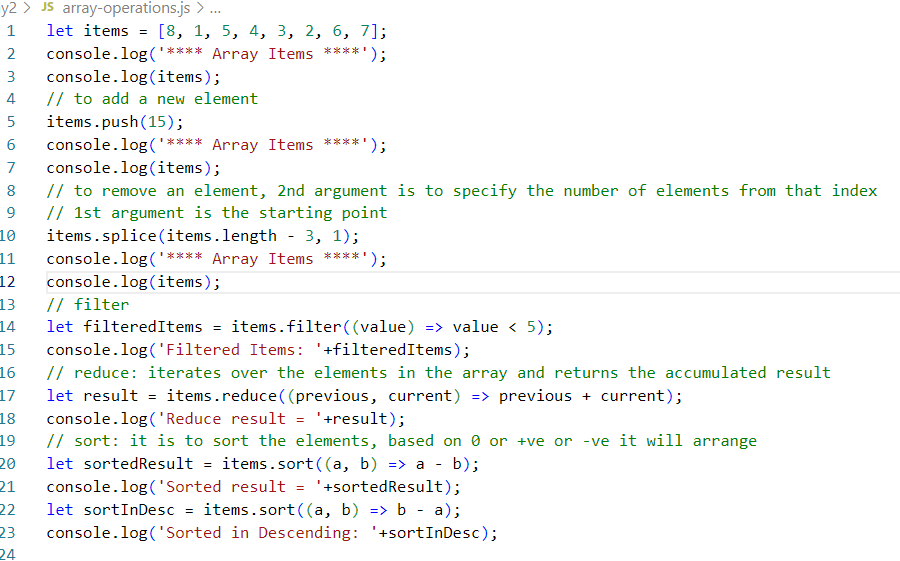
1. forEach: Iterating
2. map: transforming to another array
3. push: to add a new element in an array
4. splice: to remove an element from an array
5. filter: iterates and creates a new array by filtering on some condition
6. sort: iterates & sorts the array
7. reduce: iterates & accumulates the iterated value in a single result

Note: You can apply these methods on simple or complex types

filter(callbackFn(item)): It is used to iterate the elements and filter the element on a condition and store it in another array

reduce(callbackFn(previous, current)): It iterates the elements in the array and returns the accumulated result, and the accumulated result is provided as an argument in the next call to the callback function

sort(callbackFn(a, b)): It sorts based on the numbers compared with a & b, it could be +ve, -ve or 0



Activity:

Create 2 files

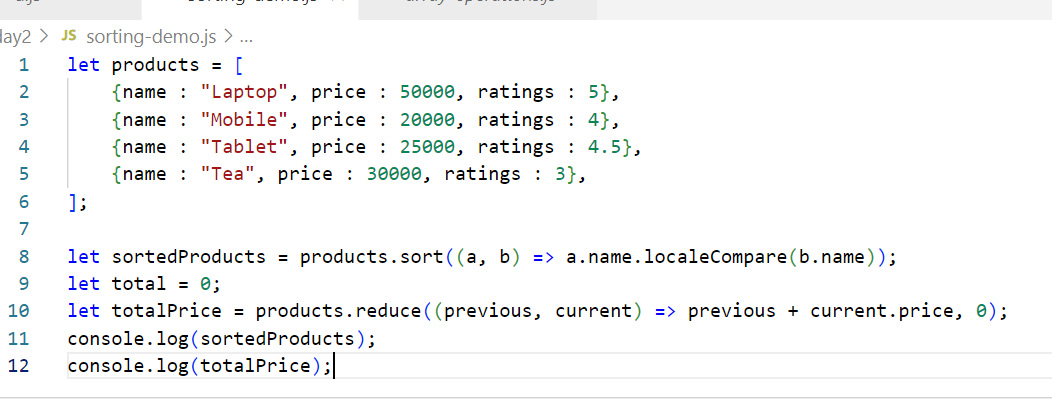
1 file will have an array to store product details like name, price, ratings, discount, all these properties must be part of a Product object, in the same file create methods that can perform following operations

1. store(product): Should accept a product object & store in an array
2. getAllProducts(): Should return all the products to the client in a sorted order based on the name
3. getAllProductsByPrice(order): Should return all the products in a sorted order by price based on the input, if order is asc then return in ascending order else return in descending order
4. getAllProductsByRatings(order): Should return all the products in a sorted order by ratings based on the input that could be either asc or desc
5. getTotalPrice(): Should return total price by accumulating all the product price

another file must call these 5 methods by taking input at runtime, these 5 methods keep in a loop such that based on some input you can call these 5 methods ex: 1: store 2: getAllProducts 3: getAllProductsByPrice 4: getAllProductsByRatings, 5: getTotalPrice, 6: Exit you must exit the loop on the input 6, until you can be in a loop to perform all the above operations.

Note: Create Product class with 4 parameters constructor that takes name, price, ratings & discount

How to sort when the data is in string form & How to calculate the total price of all the products



File Handling

fs module is present in node.js to read/write files, there are two types of methods you get

1. Synchronous read & write methods
2. Asynchronous read & write methods

import fs from ‘fs’;

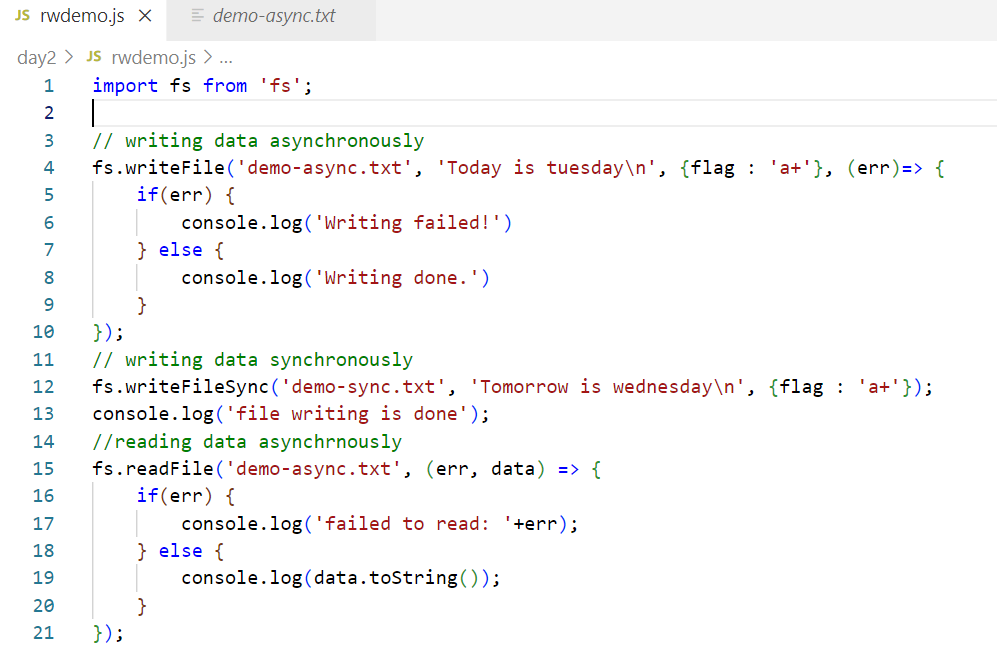
// Synchronous read/write

fs.writeFileSync(‘filename’, data);   
fs.readFileSync(‘filename’); // returns the buffer stream/byte/binary data which you can convert it into string

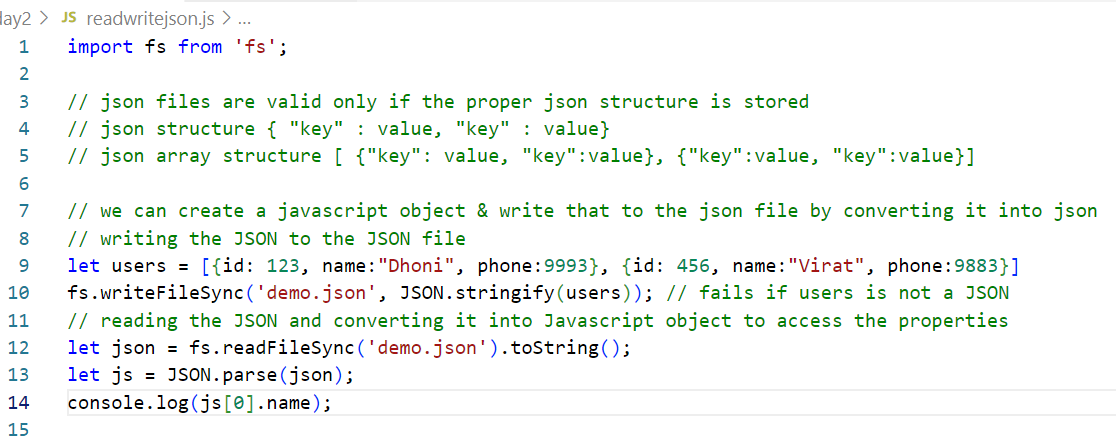
// Asynchronous read/write

fs.writeFile(‘filename’, data, callback);

fs.readFile(‘filename’, callback)



Reading/Writing JSON



HTTP module

You can use this module to create server instances, start the server & also handle the request from the client.

import http from ‘http’; // es6 syntax

let http = require(‘http’); // older syntax

How to create a server instance using http module

let server = http.createServer( callback );

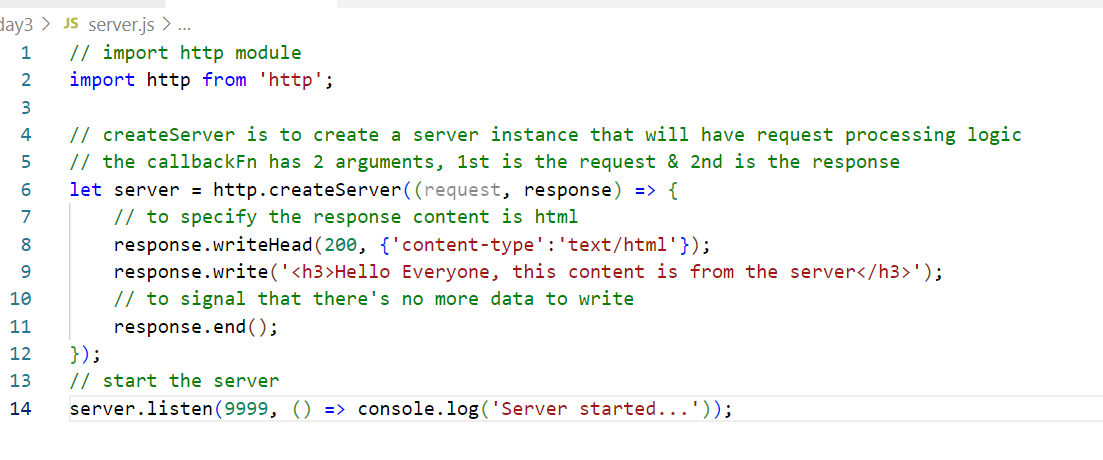
callback which is executed when the request comes to the server

How to start the server

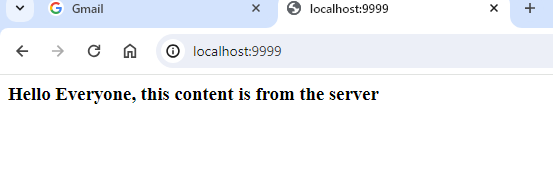
server.listen( port, callback )

callback is executed once the server starts

server.js



Output:



nodemon: It is a library that is used to reload the program when you make changes

nodemon filename.js can be used instead of node filename.js

Configure a command in package.json so that it will run the main file of your application

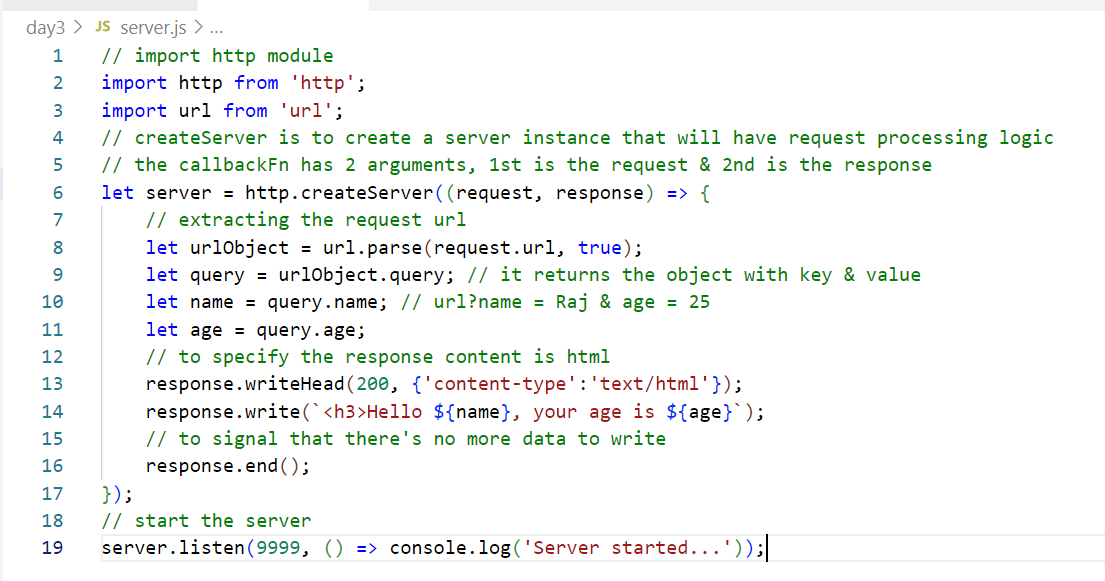
package.json

{  
 …  
 “scripts”: {   
 “start” : nodemon index.js,  
 “test” : nodemon test.js,  
   
 }  
}

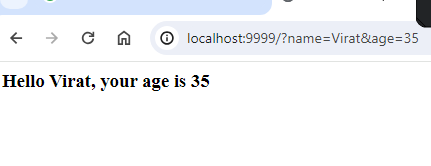


Reading the request parameters from the url

www.abc.com/?key=value&key=value



Output:



Express Server:

It is used to create RESTful webservices

Webservices: It is an online service that can be accessed by any programming languages, there are two types

1. SOAP (Simple Object Access Protocol) webservice: It uses XML to exchange the data
2. ReSTful (Representational State Transfer) webservice: It uses JSON/XML/Text/CSV to exchange the data

Guidelines while creating ReSTful webservices

1. URL : A webservice must have an unique URL to perform some operations
2. HTTP methods: A webservice must be mapped with HTTP methods to perform some operations

Note: We can have 2 webservices with the same URL but with different HTTP methods

/products : GET >> getProducts()

/products : PUT >> updateProduct()

/ products: POST >> orderProduct()

/ products : GET >> getProductsByName(); // error, because we already have GET mapped to the same URL

In Node.js to create webservcies we need to use a module called express

Express module creates an express server with then same node.js architecture & can perform all the CRUD operations with a filesystem / database

Note: express module is not inbuilt, we must download

npm install express –save

How to create a simple hello world webservice

import express from ‘express’;

let app = express(); // object that provides http methods like get, post, put, delete

app.get(‘/greet’, callback) ;

callback: it is going to handle the request and generate the response, the callback takes 2 parameters 1st is request 2nd is response

i.e.,

app.get(‘/greet’, (request, response) => {   
 // code goes here like interacting with the fs, databases like mongodb/mysql/oracle  
} );

Path parameters: It is a dynamic value that a client can mention in the url & it is mandatory, it is sent in the url as below

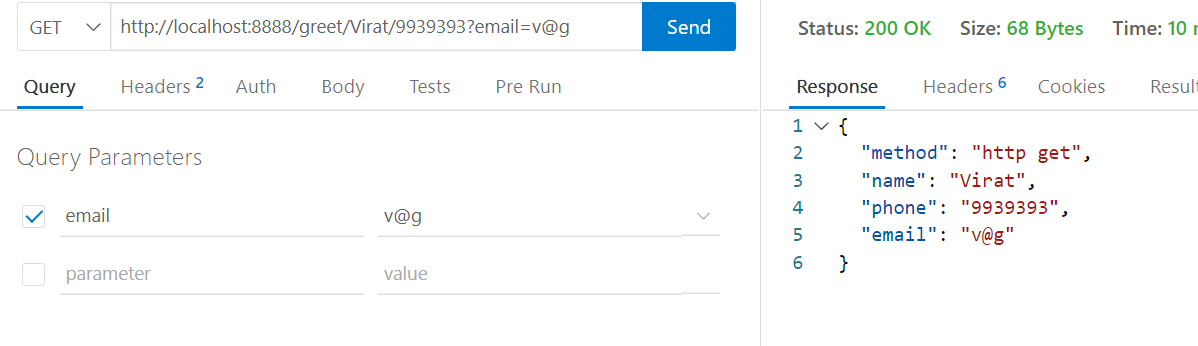
foo/100  
foo/200

Query parameters: it is a dynamic value that can a client can mention in the url & it is optional, it is sent in the url as below

foo/sales/?name=Virat  
foo/sales/?name=Rohit



Output:



Client can send the data in two ways

1. Request URL:
   1. Path parameter
   2. Query parameters
2. Request Body:
   1. Form data

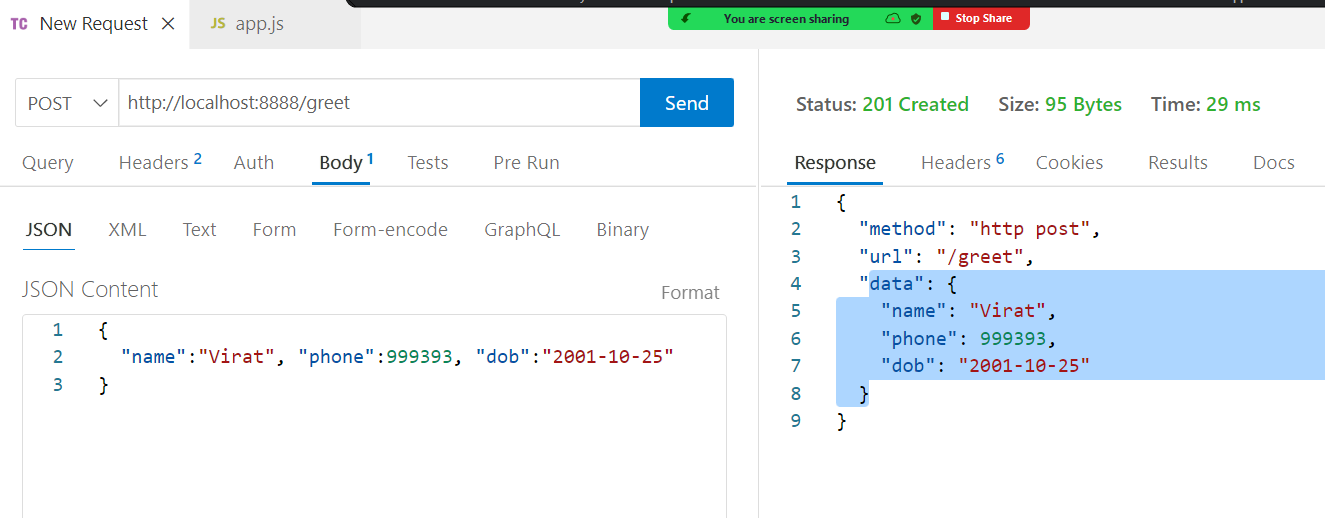
How to read the data coming from the request body

app.use(express.json()); // converts JSON to Javascript

let body = request.body;



Output:



Mongodb: it is a NoSQL database, it doesn’t need any structure

How to perform CRUD operations in mongodb

We have inbuilt javascript functions like

insertOne( json ): Create

find( json ): Retrieve

updateOne( json ): Update

deleteOne( json ): Delete

db.insertOne( { “key” : value, “key”: value });

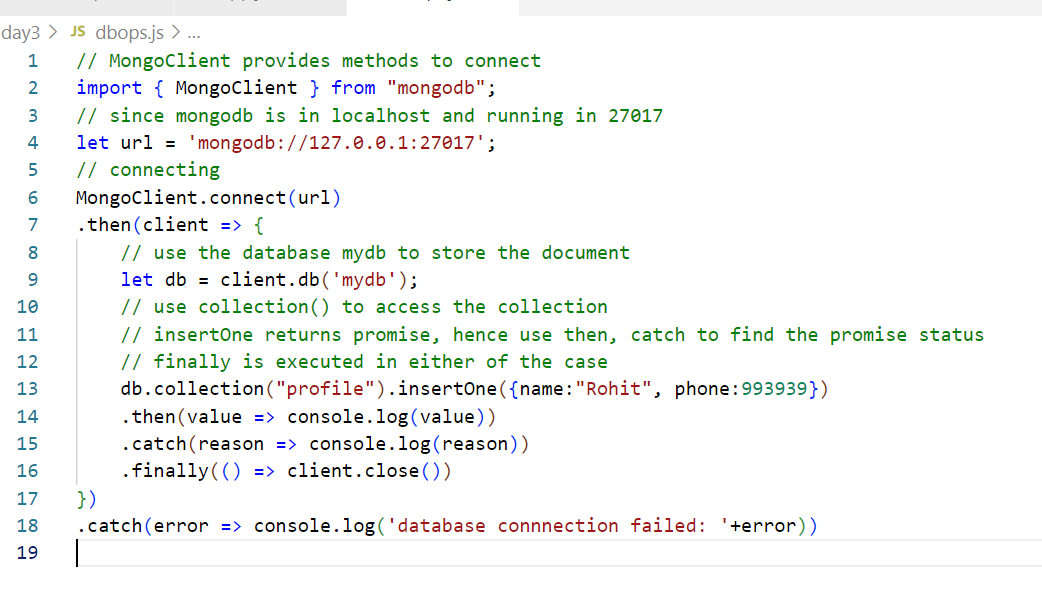
db.profile.updateOne({Filter}, { $set : { JSON } });

How to make node.js application to interact with the database

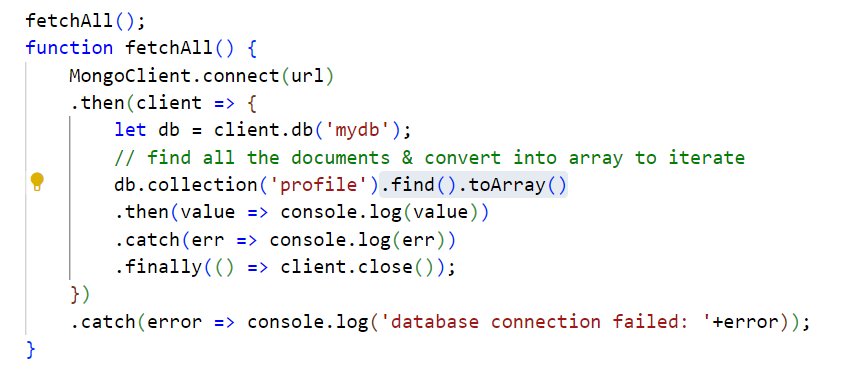
npm gives database modules to interact with any database like mongodb, mysql, postgresql.

npm install mongodb –save

Storing the document



Find all the documents



Activity:

Using express.js create webservices to perform CRUD operations on the profile collection present in the mongodb

1. Post: The JSON data coming from the client should be stored in the database
2. Get: This has to get all the documents from the database and return to the client
3. Put: This has to take phone number & \_id value from the client using path parameter and update the phone number based on the \_id value

In Thunder client you need to pass phone number and \_id value in the URL, show the response of the update in the Thunder client

1. Delete: This to take \_id from the client using path parameter and delete the matching document in the database, show the response of the delete in the thunder client
2. Get: This has to get a single document that matches to the \_id, pass the \_id from the URL & show the document matches to the \_id or show the error message if \_id is not found in the thunder client

Note: Show all the response data in JSON format, in the thunder client you must able to see it

Datastructure

In Javascript we have predefined data-structures

1. Arrays
2. Objects

We can create our own datastructure’s in Javascript if our application expects that, there are many datastructures which are useful in the applications

1. Stack
2. Queue
3. LinkedList
4. Map

Stack: It uses Last In First Out pattern

Queue: It uses Fist In First Out pattern

LinkedList: It maintains elements in the form of list, each element is called as node, which will have data & the location of previous & next node except the first & the last node

Map: It maintains elements in key value pairs

Implementing a Stack

We need to provide operations like

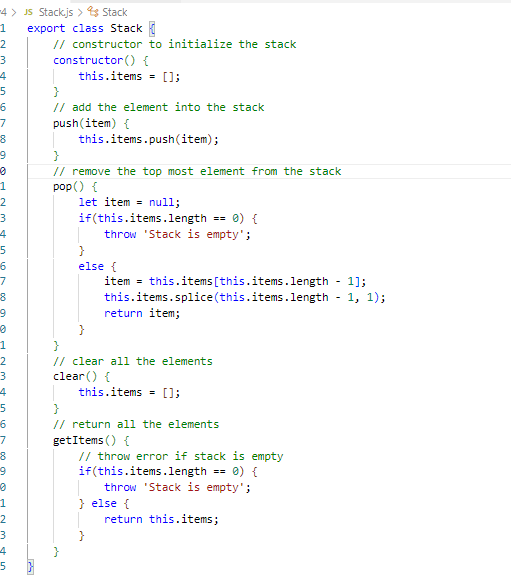
1. push: To add the element at the bottom of the stack
2. pop: To remove the elements from the top of the stack
3. clear: To remove all the elements from the stack
4. size: To return the size of the stack
5. getItems: To return all the elements

Note: We need to make use of an array to implement Stack, in Javascript arrays are dynamic in nature

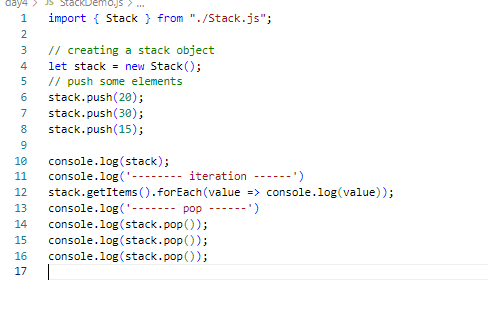
Creating a stack

export class Stack {   
 constructor() { this.items = [ ]; }  
 push(item) { this.items.push(item); }  
 pop() { … return topMostElement }  
}

Stack.js



StackDemo.js



Map: It is used to store element in key value pairs, in new feature of Javascript Map is available, which was not there earlier

let map = new Map();

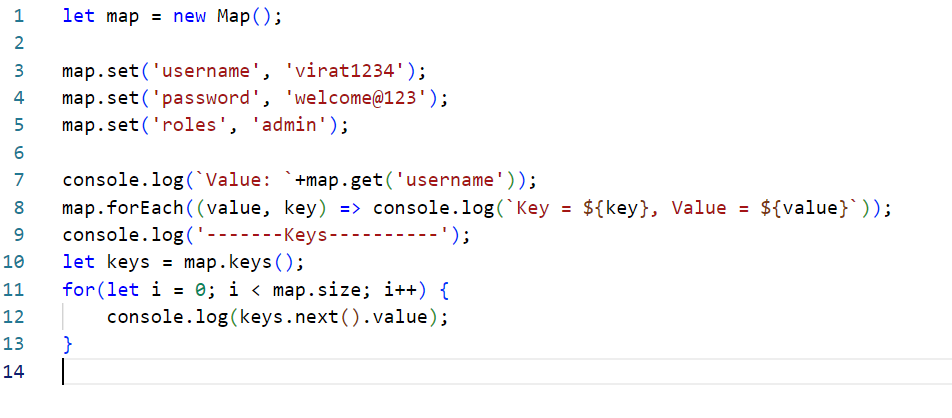
map.set(“key”, value);  
map.set(“key”, value);

Note: Map maintains uniqueness, where key can’t be duplicate

map.get(“key”) returns the value of that key

map.keys(): returns all the keys present in the map

MapDemo.js



Activity: Create a Queue.js and maintain the elements in FIFO order, so that the first element added must be the first one to be removed, create add(), remove(), peek(), clear(), getItems()

add(): Adds the element   
remove(): remove the element on FIFO order  
peek(): returns the element at the top of the queue  
getItems(): returns all the elements in the queue.

Testing Node.js with Mocha & Chai

Mocha: is a testing framework for Javascript, it can test both synchronous & asynchronous code in Javascript.

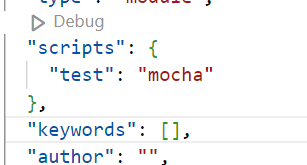
Chai: is an assertion library which tests your code with expected & actual results

npm install mocha chai –save-dev

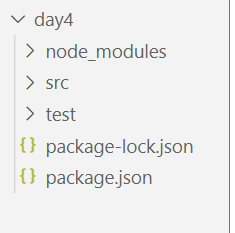


Note: We need to mention in the package.json that mocha is the testing tool node must use when you use npm test, mocha needs all the test cases to be present in a test folder

package.json



Keep the project structure in a way where all the main scripts goes inside src & test scripts goes inside test folder.



src/Greetings.js

function greet(name) {   
 return “Welcome “+name;  
}

test/Greetings.spec.js



>> Run using >> npm test

Node.js with front-end technologies

Node.js helps in providing necessary tools to implement UI’s at the front-end that can be launched using node server, some of the popular front-end technologies like React.js, Angular framework can be developed with the help of node modules

React & Angular framework are used to develop single page applications, where only part of the page will be reloaded when something needs to be updated in the front-end.

React is from Facebook

Angular is from Google

Angular framework is used to develop front-end applications which is single page application, it can be used to develop web, mobile applications.

Angular uses HTML & Typescript(Extended version of Javascript)

Components: These are reusable UI’s that you can develop independently as well reuse in other components as well.

Google has provided angular-cli toolkit to create & run angular projects

Toolkit has following features

1. Downloads a project template with a standard structure with src, test, package.json and etc
2. Will have compilers to compile the project automatically
3. Will have live reloading feature, your browser will be updated automatically when you make changes
4. Will have important commands to use in the project like start, test, and many more.

npm install -g @angular/cli

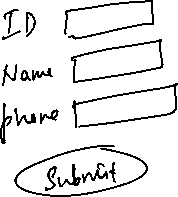
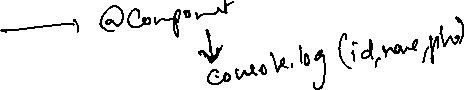
npm install @angular/cli

How to display the component data in the HTML

{{ property }}  
{{ property.nestedProperty }}

Activity:

Create a component that will have form controls to enter id, name & phone, and pass these data to the component class which will print these details in the console.log



FormsModule: It helps you to store the form data in an object and pass to the component class

Add this FormsModule in the AppModule

AppModule will have everything of your application like components, pipes,

Accessing the API’s from the angular

HttpClient object is used to access the API’s, angular can supply this object to the constructor of the class automatically, but to make that happen we must add HttpClientModule in the AppModule

Service: It is a reusable class that will have business logics that can be accessed by multiple components,

Service class can have HttpClient & multiple components can use this service class object to call the API’s

Note: Angular also takes care of creating the object of the service classes & supply to the constructor of the class automatically.

@Injectable()  
export class ProfileService {   
 // Angular supplies the HttpClient object to the constructor  
 constructor(private \_http : HttpClient) { }   
  
 fetch() { \_http.get(url); }  
 store(data) { \_http.post(url, json); }  
}

@Component({})  
export class ProfileFormComponent {   
 //Angular supplies the ProfileService object to the constructor  
 constructor(private \_service : ProfileService) { }   
  
 handleForm(profile: any) {   
 \_service.store(data)  
 }  
}

Activity:

Create a component that will allow you to enter an id whose value will be from 1 to 10, based on these input you must able to display the user details like name, email, address, phone in the component, call the API present in the JSON placeholder that can accept the id from 1 to 10.



Node.js Socket IO

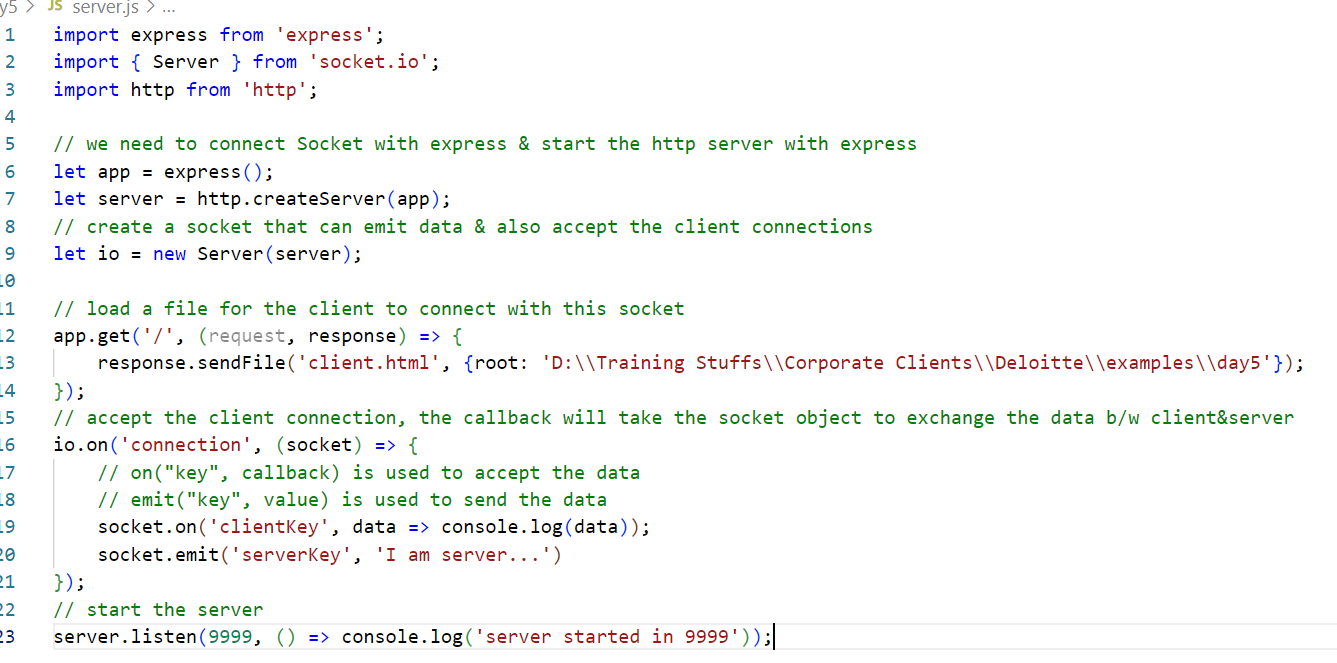
It is used mainly to share the data between the client & server without using request & response, Many online games, chatting programs use socket.io to share the data.

Event Emitter: It is a way to send the data to the another program

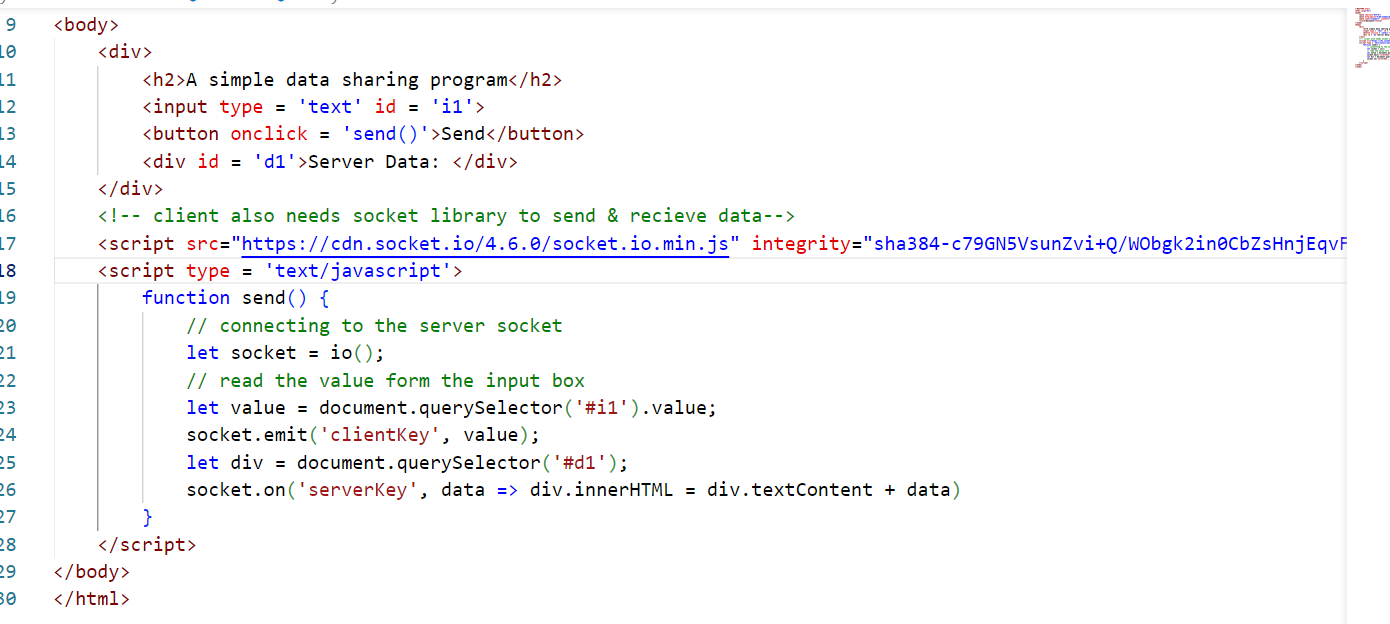
npm install socket.io

We need to install express also because socket.io uses express as well

Server program



client.html



1st Run server.js using node server.js

then From browser type : <http://localhost:9999/> to load client.html

React.js  
It is a Javascript library to develop single page applications, it uses HTML & JSX to create applications for web & mobiles

JSX: it is an extended form of Javascript that simplifies writing HTML code

Javascript

let content = “<div>Some content “+expression+”</div>”;

JSX

let content = <div>Some content {expression}</div>

Components: These are reusable UI”s which you can independently create & maintain, components are the building blocks to develop SPA

Note: Component names must begin in upper case letter i.e., the first letter

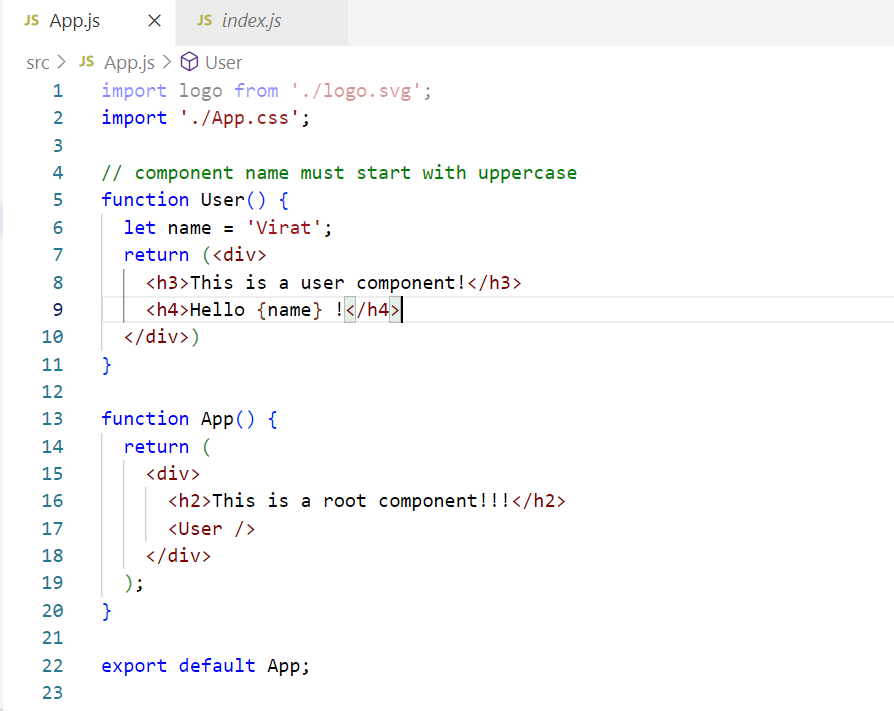
function User( ) {  
 // some scripts  
 return (<div>HTML Content {expression} </div>  
}

<User />: This shows the HTML content and its dynamic data as well

create-react-app: It is a toolkit used to create react projects

npx create-react-app application-name

App.js



In React you can handle events using callbacks

let handleClick = () => {   
 // some code  
}

In React event names must use camel case & begin with lowercase

<button onClick = {handleClick}>Click</button>

Component states

States are used to modify the component data, it is created using a useState() hook, which is a react inbuilt hook function

let [name, setName] = useState(‘’);

Note: Normal variables wouldn’t re-render if updated, hence we will use states, which are re-rendered when modified

setName() is a function to modify the data,

name is a variable to store the data

let [age, setAge] = useState(‘’);

let [phone, setPhone] = useState(‘’);

setName(‘Raj’); // now name = Raj  
setAge(25); // age = 25  
setPhone(99999); // phone = 99999

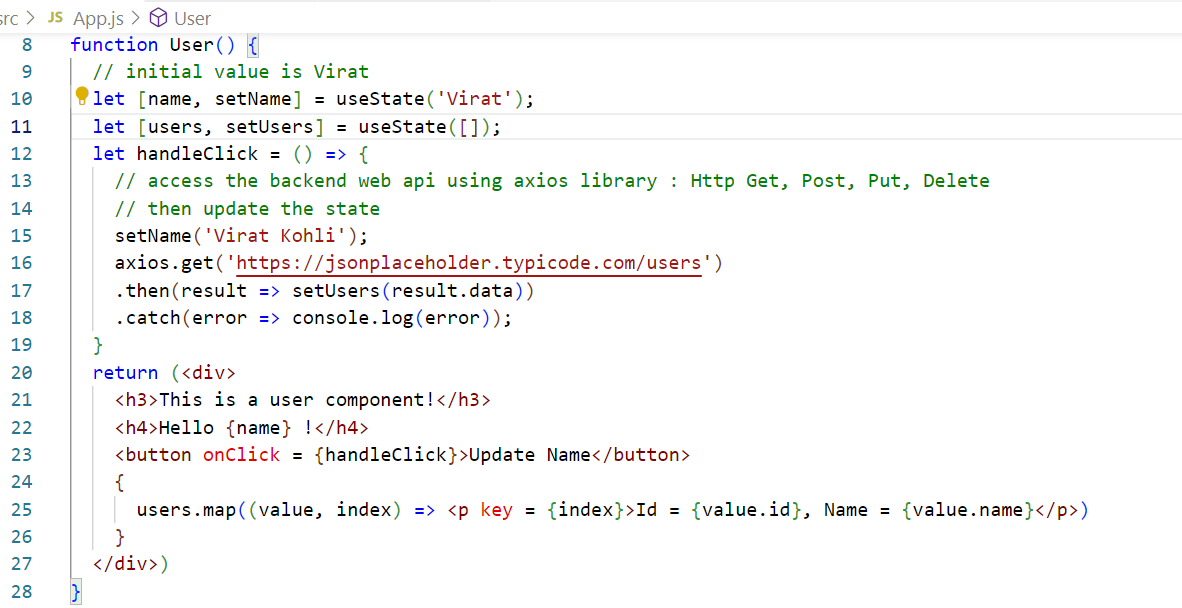
You need to import useState

import { useState } from ‘react’;

axios: It is a library used to call api’s, in react its not available by default you must install it using npm install axios –save.

axios.get(url).then( callback ). catch( callback )

axios.post(url, data).then( callback ). catch ( callback )



Routers: It helps you to navigate from one component to another component, we have this both in React & Angular