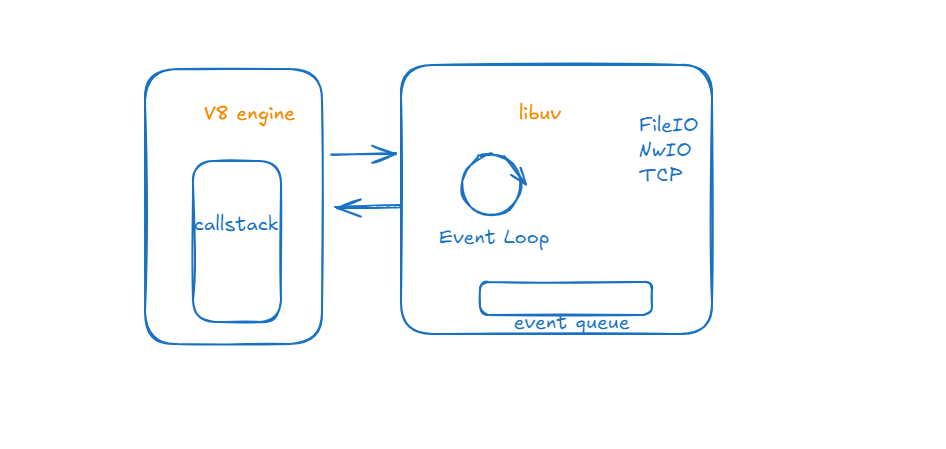
Node.js

What is Node.js

* Node.js is a runtime environment which can run the Javascript code outside the browser
* Before Node.js, you could run the Javascript only using browser, Javascript was used only at the front-end to provide UI logics like adding effects to the UI, making the UI content dynamic
* With the help of Node.js you can now write Javascript for the backend applications that can interact with various resources like files, databases, buffers and so on

Node.js architecture



Node.js has 2 main components

V8 engine: It is Google high performance engine which converts Javascript code to Machine code.

Libuv: It mainly handles asynchronous IO operations

What is IO

IO refers to the programs interaction with the

* System Disks
* Network
* Read/Write files
* Talking to the DB
* Making HTTP Requests

These programs are slow compared to the CPU work or accessing memory

What are Synchronous & Asynchronous operations

* Synchronous operations: These are the operations that are executed sequentially one at a time

let result = connect(“www.foo.com/xyz”);

console.log(result);  
console.log(“Done”);

Note: The program runs sequentially by first printing the result & then Done

* Asynchronous operations: These are the operations which are independent from the primary program flow & this will let primary program to continue without waiting for the operations to complete

connect(“www.foo.com/xyz”, (result) => console.log(result));  
console.log(“DONE”);

Note: The program prints the DONE first and the result once the asynchronous operation is completed

ex: setTimeout(), db related functions()

Node.js has V8 & Libuv, the V8 is the one which takes care of running your javascript code, it is single threaded

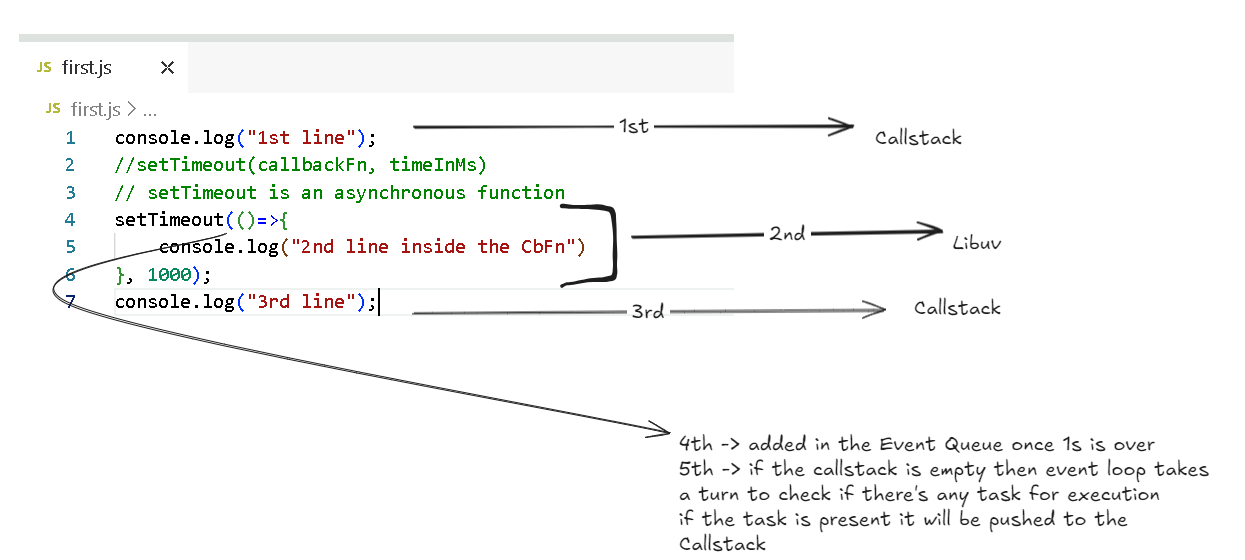
Libuv has many programs like Event Queue, Event Loop, TCP/IP, File IO, N/W IO and etc., these are handled by system threads, they don’t execute the Javascript code user has written, these programs are asynchronous which are implemented using C/C++ & exposed as a Javascript functions, some the functionalities are setTimeout, setInterval(), fs.readFile(), fs.writeFile(), app.get(), app.post() and etc these are all asynchronous methods which are written in C/C++ & provided as Javascript functions

console.log(“first line”);  
setTimeout(() => { some code }, 1000);  
consoe.log(“second line”);

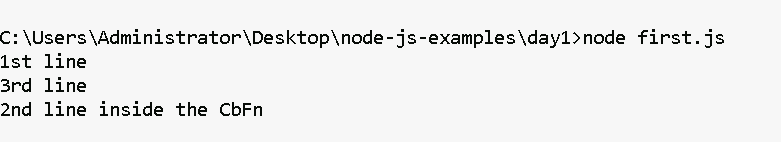
Output:

first line  
second line  
some code

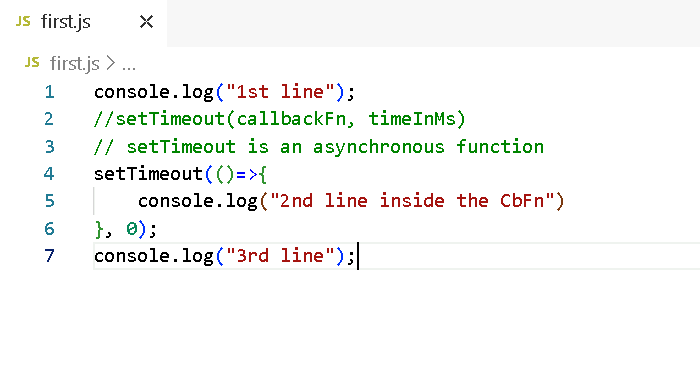
Note: Even if we give 0 in the setTimeout the callback of the setTimeout is executed once the call-stack of V8 is empty or once its pushed to the call-stack after the event loop makes a trip



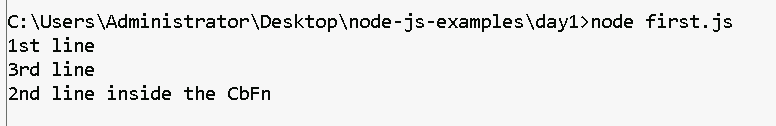
Output:



Even if you give 0 in the setTimeout, it wouldn’t be executed immediately, because the setTimeout() function is handled by libuv & the task is added to the event queue after the timer completes whether its 0 or 1000ms, then the event loop makes a trip to check if the event queue has any task for V8 engine to execute.



Output:



Global Objects

Node.js has 1000’s of objects in many node.js modules like

process  
console  
Buffer  
setTimeout  
setInterval  
setImmediate  
and many more

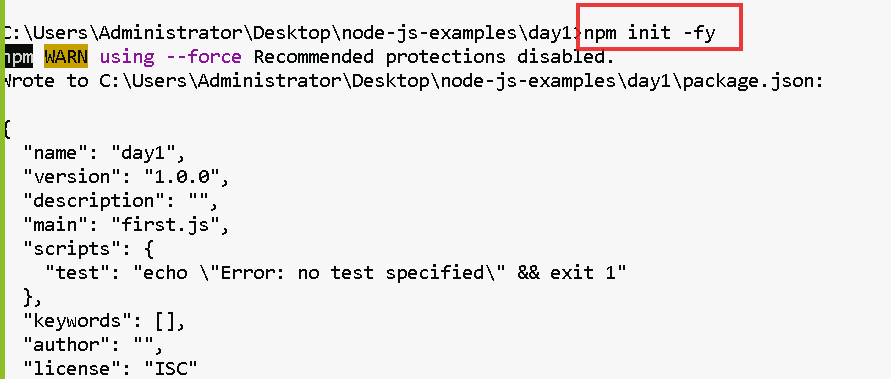
Node.js uses a configuration file called package.json which an heart of every node.js project, if you don’t have this you can create using npm init option

How to create package.json

npm init

or

npm init -fy (force yes)



Note: By default node.js uses ES5 module to import

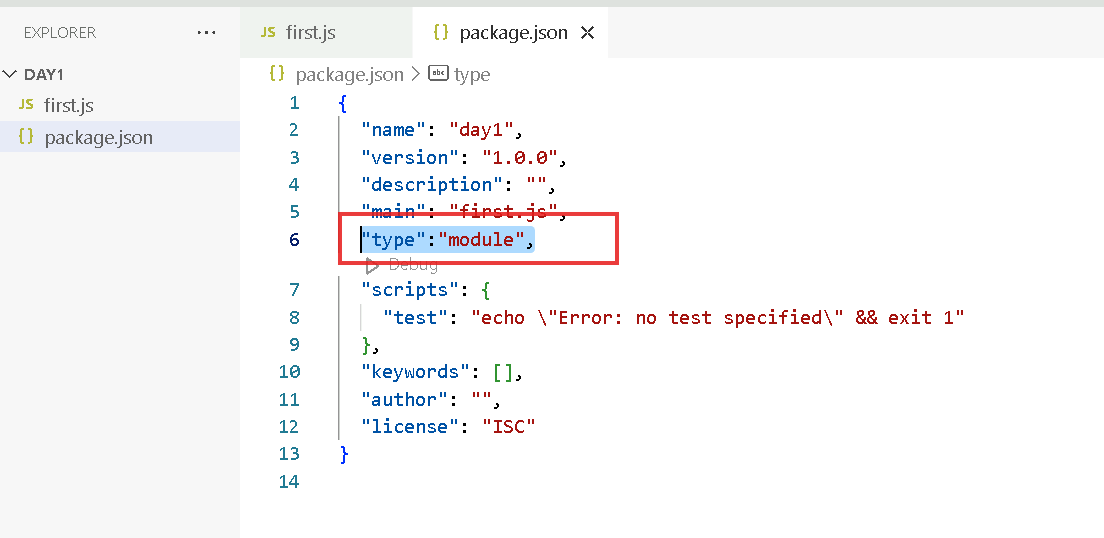
i.e.,

let fs = require(“fs”);

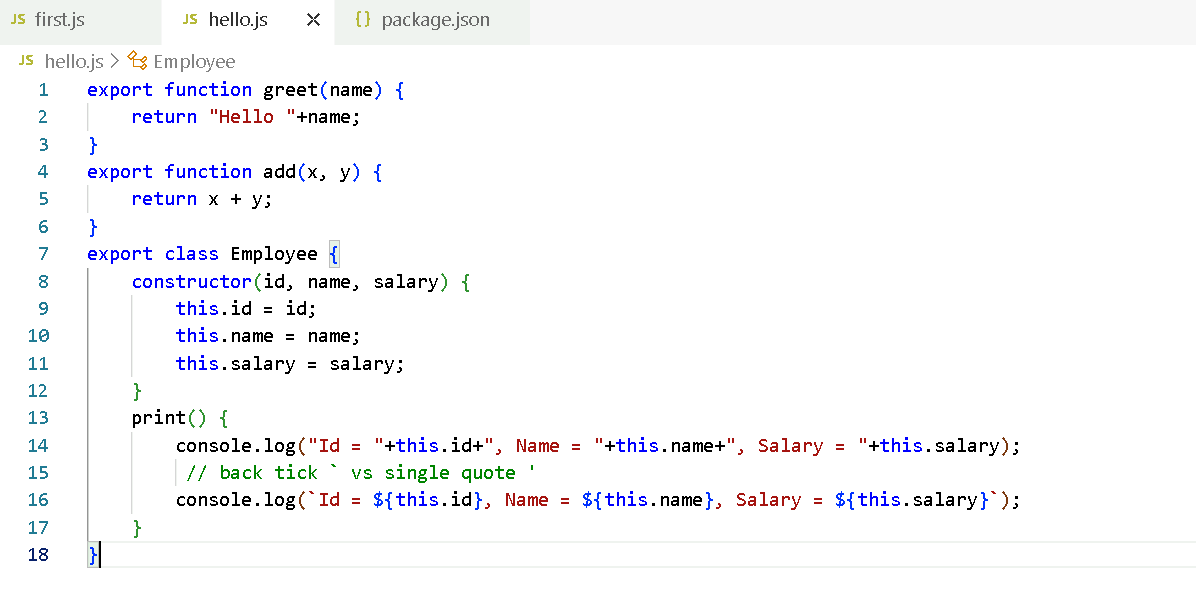
However you can configure package.json to use ES6 modules to import or export

i.e.,

import fs from ‘fs’;



How to export & import the modules in Node.js

hello.js  


first.js



The above modules are imported in {} because they are named modules, in Javascript we have default module that can be imported with any name but we must not use { }

Note: In a single javascript you can have maximum one default export & any number of named exports

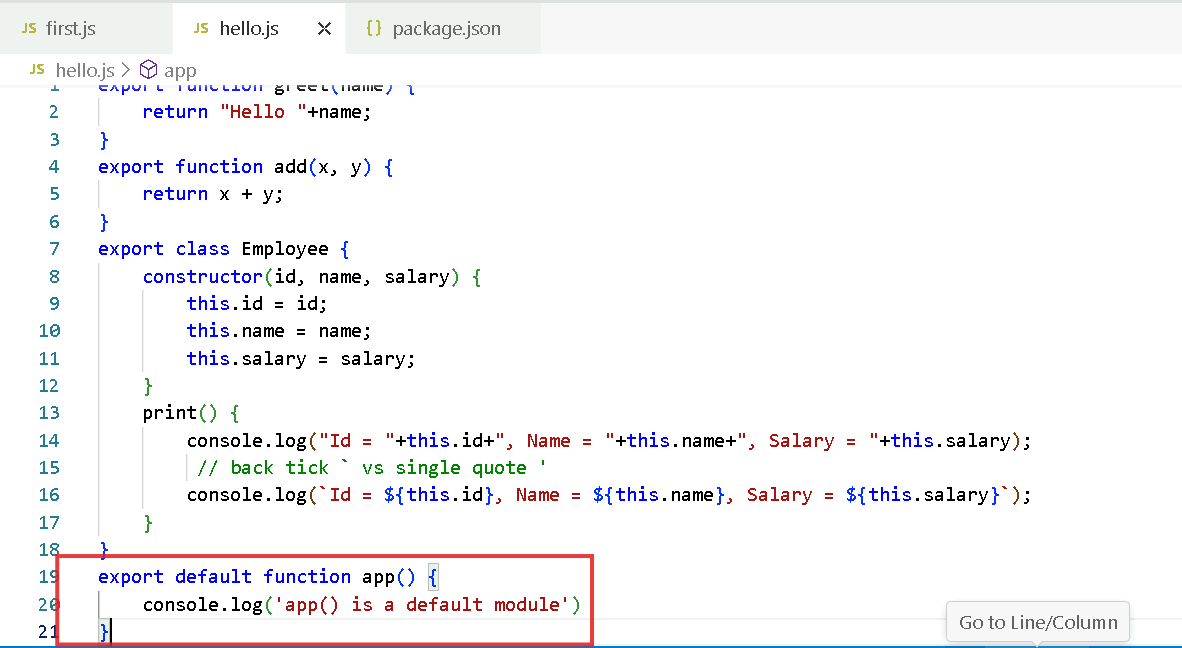
export default function app() { … }

import app from ‘./file.name.js’

or

import x from ‘./file.name.js’;

hello.js



first.js



Note: In Javascript you create default modules when a main functionalities needs to be added in the module, ex:

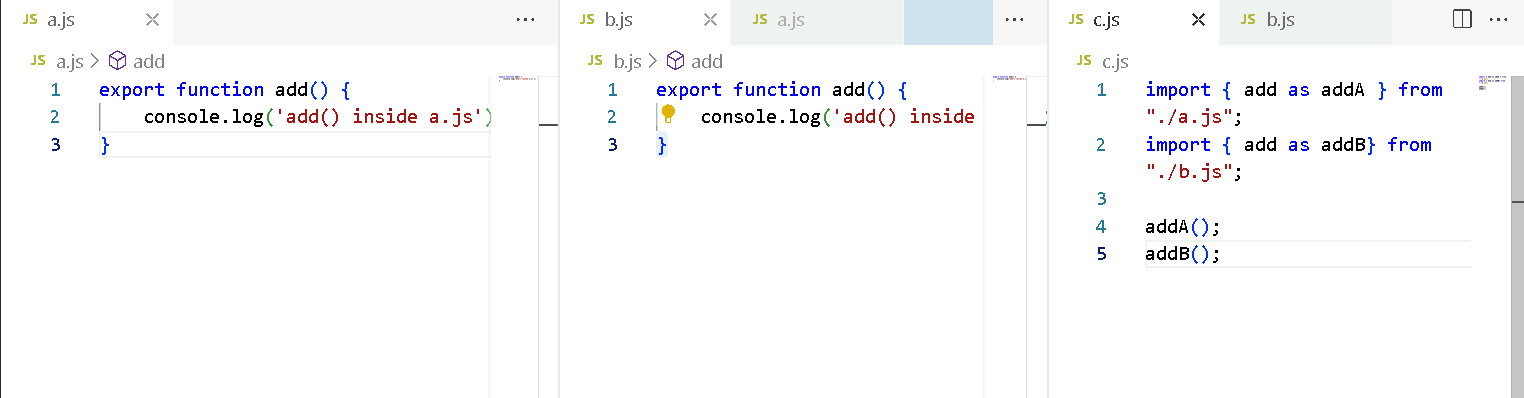
axios library of React.js has a default module that provides methods to access backend

React.js library has a default module that provides functionalities to create UI’s

Express.js library has a default module that provides function to create an object & access HTTP methods

What if there are 2 modules with the same name in different Javascript file, how does the node.js resolves these modules

Activity: Create a.js & b.js and in both the files create same named export function & in c.js try to import the functions present in the a.js & b.js and call them in c.js



Old style of exporting & importing

exports.add = function(x, y) { … } // or module.exports.add = function (x, y) { … }

let add = require(“./file.js”).add;

New style of exporting & importing

export function add(x, y) { }   
import { add } from ‘./file.js’;

process.nextTick(callbackFn):

It is executed automatically when event loop takes a turn

setImmediate(callbackFn): This similar to setTimeout() which is executed immediately without any timer

but if you use setTimeout, you will use setTimeout(callbackFn, timer), however you can give 0 in the timer and make it work like setImmediate()

Summary:

* Node.js
* Node.js architecture (v8 & libuv)
* default & named exports
* package.json:
* process.nextTick, setImmediate, setTimeout

C++ Add ons

In Node.js many libraries which do asynchronous operations are written in C++, C++ add-ons provide an interface which makes Javascript functions & C++ to interact i.e., which are exposed as a Javascript functions.

Day 2 Agenda

* Fundamentals of Node programming (controls, loops, operators)
* FS Module
* HTTP Module

Fundamentals of Node

* Operators
* Control Statements
* Looping Constructors

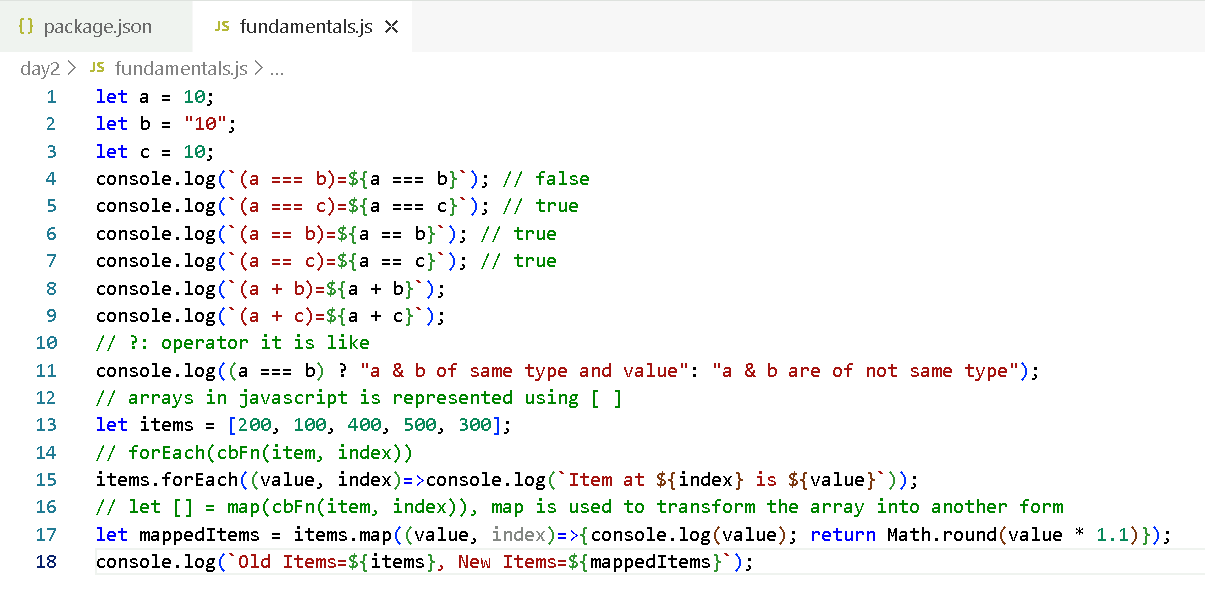
Operators in Javascript/Node.js:

++, --, =, ==, ===, <, >, <=, >=, !=, &&, ||, ?:

== vs ===

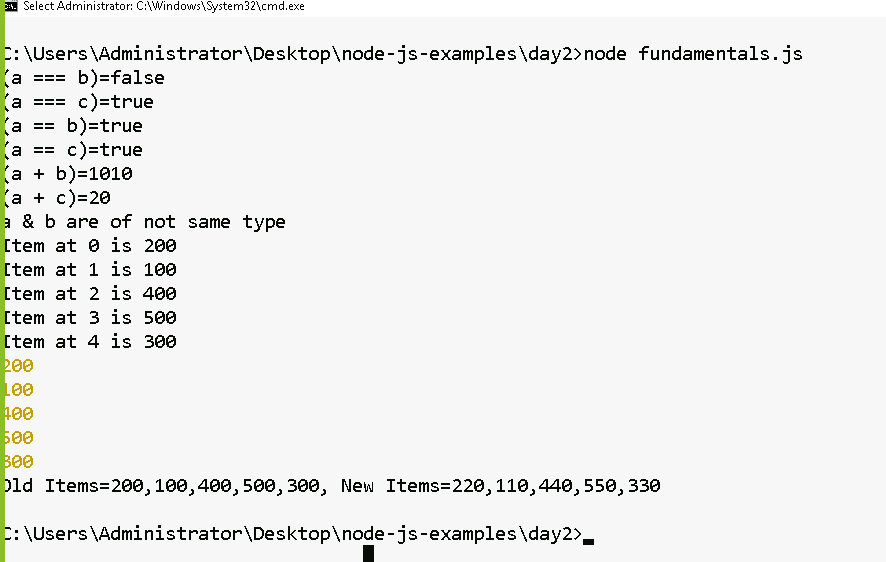
== will only check value is same or not

=== will check both value & type of value are same or not

Using forEach & map

forEach is only for iterating, however map can iterate and generate a new array from the old array

Output:



Different ways of writing arrow functions

(arg) => statement;

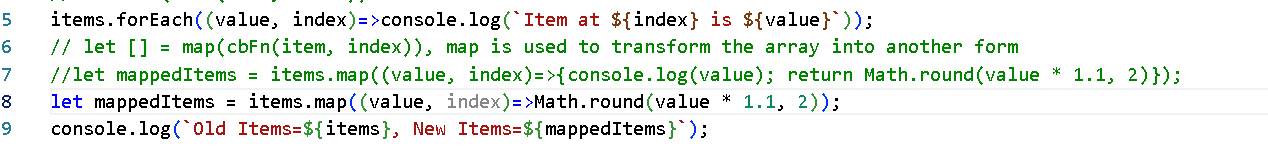
arg => statement;

(arg1, arg2) => statement;

Arrow functions are the alternate form of callback functions

Callback functions vs Arrow functions are written as below

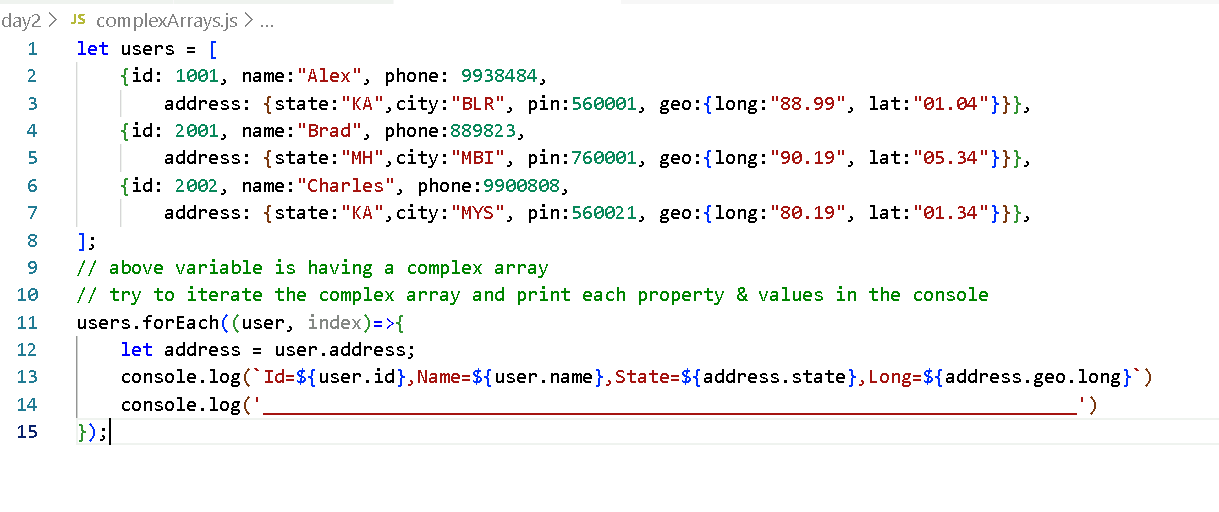
|  |  |
| --- | --- |
| Callback Fn | Arrow Fn |
| function(arg) {   statement;  return exp; } | arg => { statement; return exp; } |
| function (arg) {  return exp; } | arg => exp;  [or]  arg => { return exp; } |
| map( function(item, index) {   return exp; }); | map( (item, index) => exp ) |



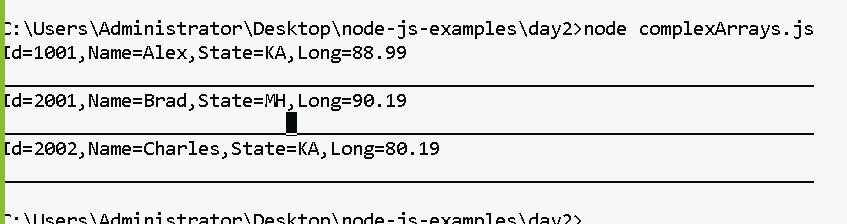
Working with complex arrays which will have objects & nested objects

[  
 {id: 100, name : “Alex”, phone: 99993, address : {state:”..”, city:”…”, pin: “…”, geo: {long:”…”, lat: “…” }} },  
 {id: 200, name : “Brad”, phone: 9876, address : {state:”..”, city:”…”, pin: “…”, geo: {long:”…”, lat: “…” }} },  
  
]

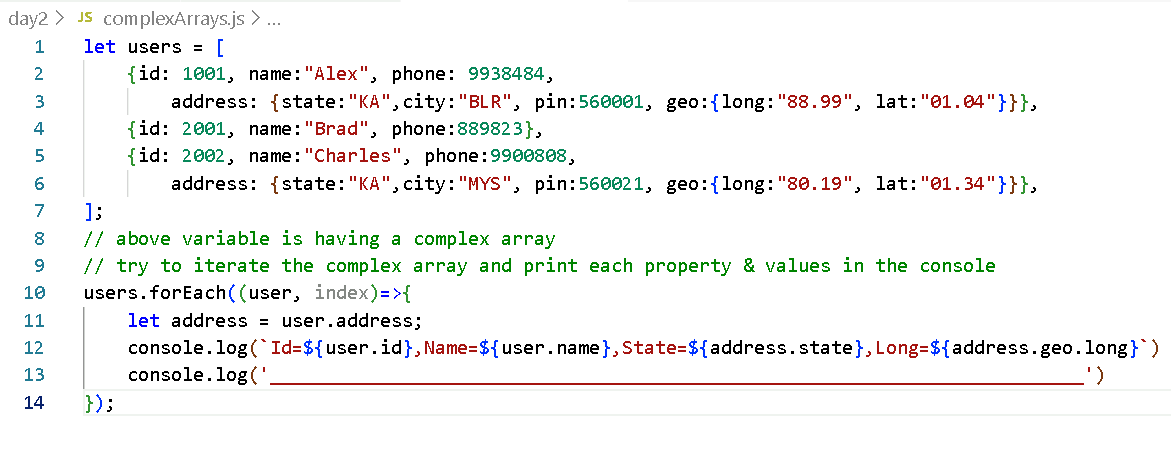
complexArrays.js



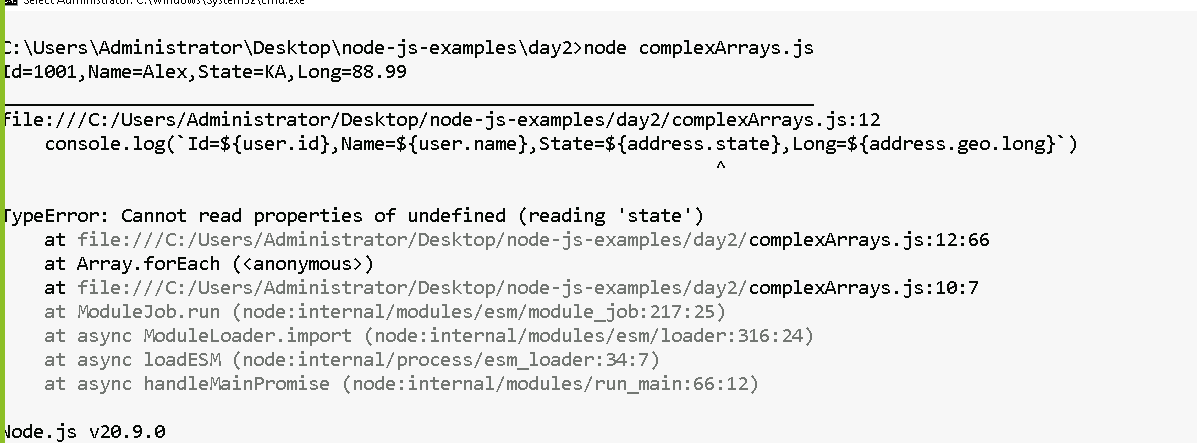
Output:



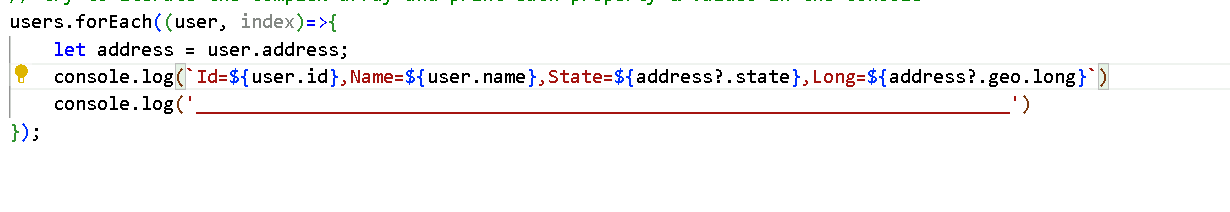
Note: There could be some scenarios where complex properties may not exist, in that case what is the solution



Output:



You can use ?. to access the nested properties to avoid the errors



How to take input from the keyboard in node.js

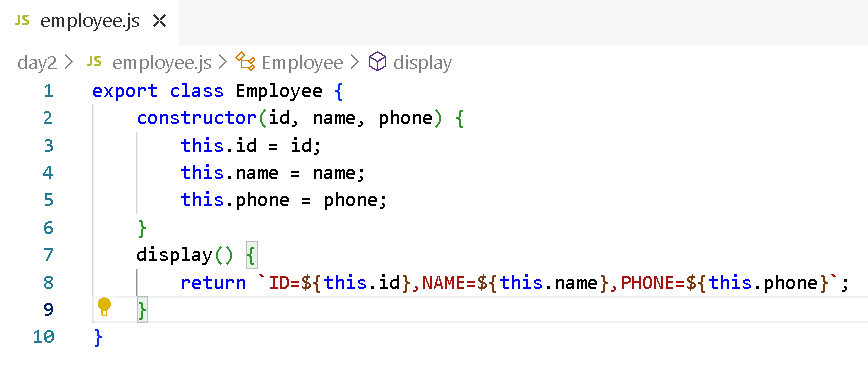
There’s a third party library you can download use which is readline-sync, it provides various methods to take different types of inputs

To install: npm install readline-sync

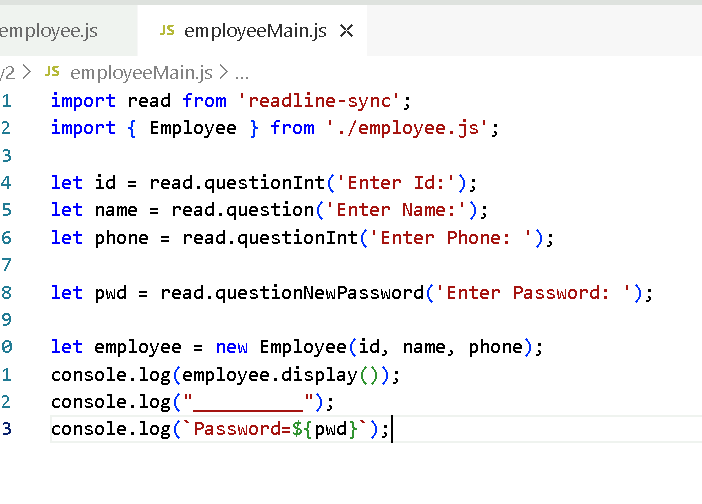
import read from ‘readline-sync’;

let id = read.questionInt(“Enter id: “);  
let name = read.question(“Enter name: “);

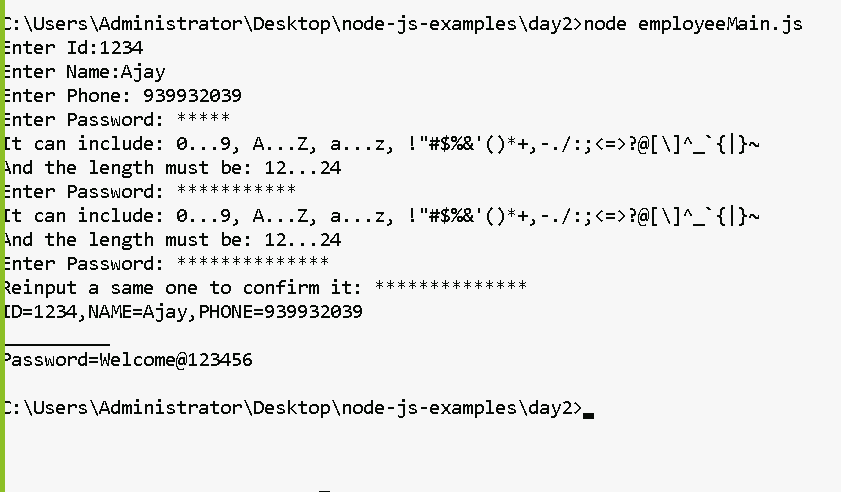
employee.js



employeeMain.js



Output:



How to read/write files in Node.js

Node.js provides fs module to read/write files, it has methods like

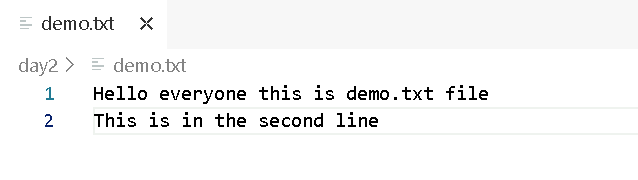
readSync(“filename”)  
writeSync(“filename”, content, options)

fs module is inbuilt in the Node.js, which means you can use it without explicitly downloading

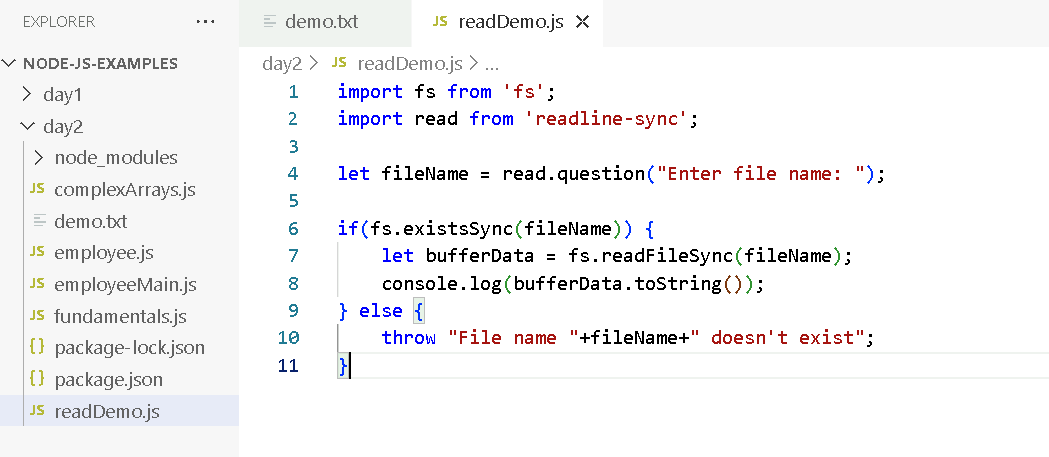
Note: writeSync function can only write text data / Buffer / JSON , but not any javascript object or array of Javascript objects

readSync reads the file and returns the Buffer which must be converted to text format if in case the data is text data

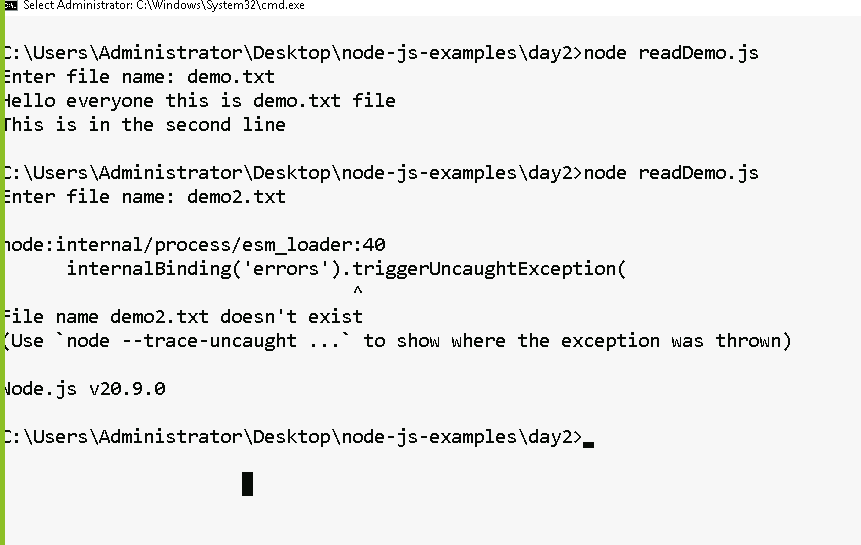
Create a demo.txt



Now we can use fs module to read the demo.txt using readFileSync(“demo.txt”)



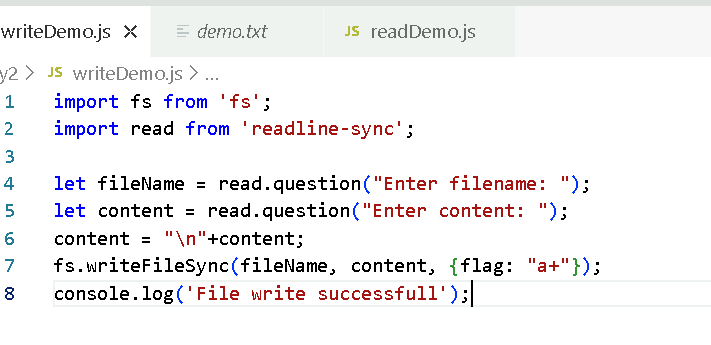
Output:



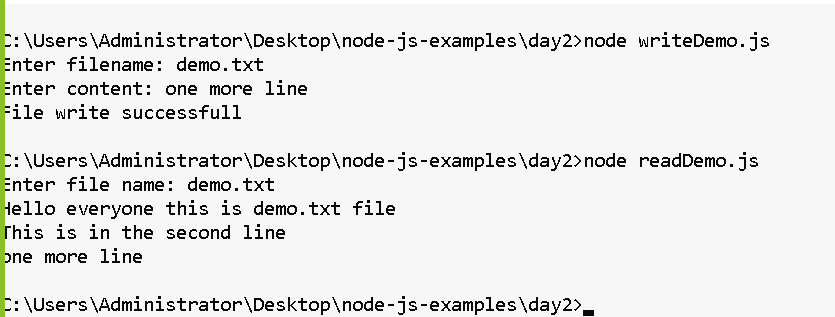
Activity:

Using the fs.writeFileSync(filename, content, [optional]) write the content to the text file by taking input of the content from the keyboard, you need to keep the old content of the file each time you write the new content, ensure you don’t overwrite the existing content with the new content

fs.writeFileSync(filename, content, { flag: “a+” });



Output:



Try to create an employee object by taking id, name and phone number from the keyboard input and write to a JSON file

You cannot write a Javascript object directly to the file, you must convert it into a JSON and then write to the JSON file

There are methods available in a JSON object to convert javascript to JSON & vice versa

1. JSON.stringify(object): Converts Javascript object to JSON string
2. JSON.parse(jsonString): Converts JSON to Javascript object

Javascript object

{id: 1234, name : “Raj”, phone : 9939393}

JSON object

{“id”: 1234, “name” : “Raj”, “phone” : 9939393}

We will create a reusable program that can accept an object & write to a file

export function writeEmployee(obj) {   
 let json = JSON.stringify(obj);  
 fs.writeFileSync(filename, json);  
}

Note: You can’t use {flag:”a+”} while writing JSON to append old JSON with new JSON

It will become invalid JSON because it looks like this

{“id”:1234, “name”:”Alex”, “phone”:9392933}   
{“id”:4556, “name”:”Brad”, “phone”:939339 }

What is a valid JSON if in case it needs to maintain multiple JSON data

[ { … } , { … } ]

Since the JSON file is re-written we must follow following steps

1. Check if the file is present, if not present create a json file & write an empty array

employee-db.json >> []

1. If file is present read the JSON array convert to Javascript, then using push of array push the javascript object
2. Convert the Javascript object to JSON which becomes JSON array

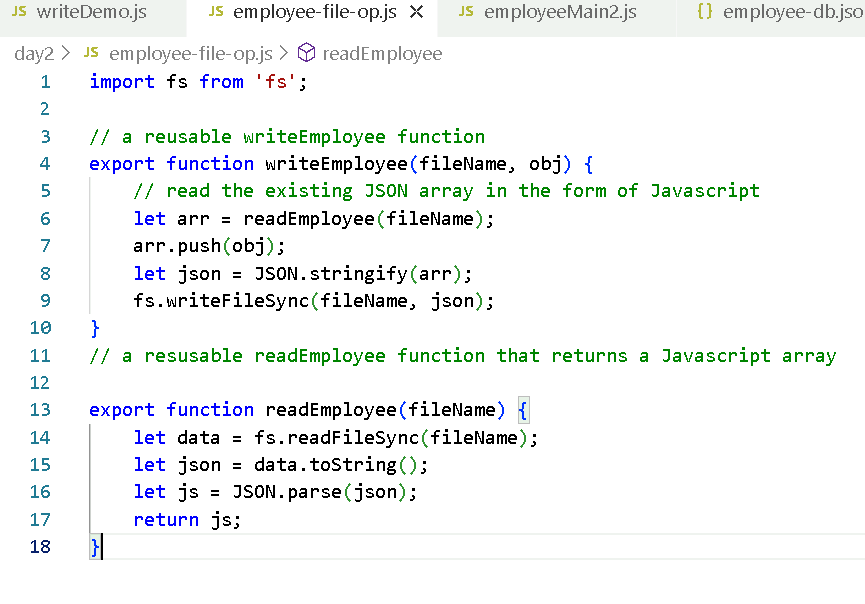
i.e., [ { “key” : value,… }, { “key” : value,… } ]

Step 1:

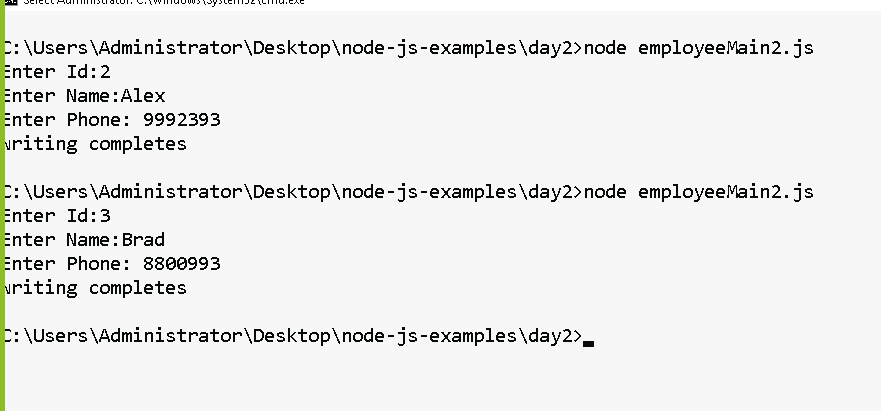


We can either write a program to write an empty array or create an empty array in JSON file

Step2:



Output:



Day 3

1. HTTP
2. EXPRESS

HTTP:

It is an inbuilt module in Node.js that can handle requests & response

It helps you to create a server to handle HTTP requests

// Importing the module

import http from ‘http’;

// Creating a server

let server = http.createServer( callbackFn(req, res) );

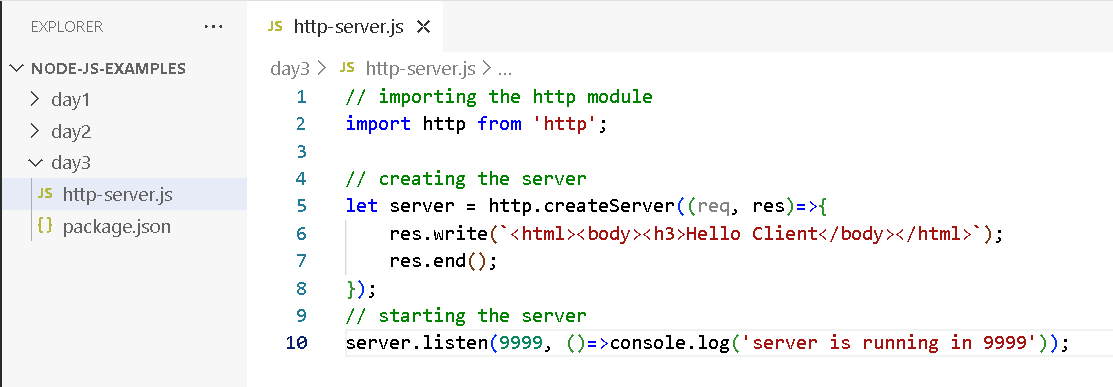
callbackFn is executed whenever a request is made to the server

// starting the server

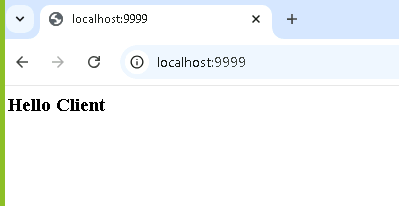
server.listen(PORT, callbackFn());

callbackFn of the listen is executed once the server starts

http-server.js

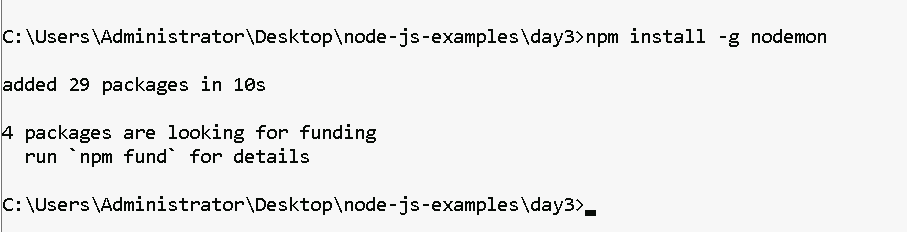


Output:



nodemon: It is a library that can detect the changes in your application and reloads when the application changes

npm install -g nodemon



You can use nodemon from any location

node app.js

nodemon app.js

Your server an also take query strings in the URL and parse the URL to read the data

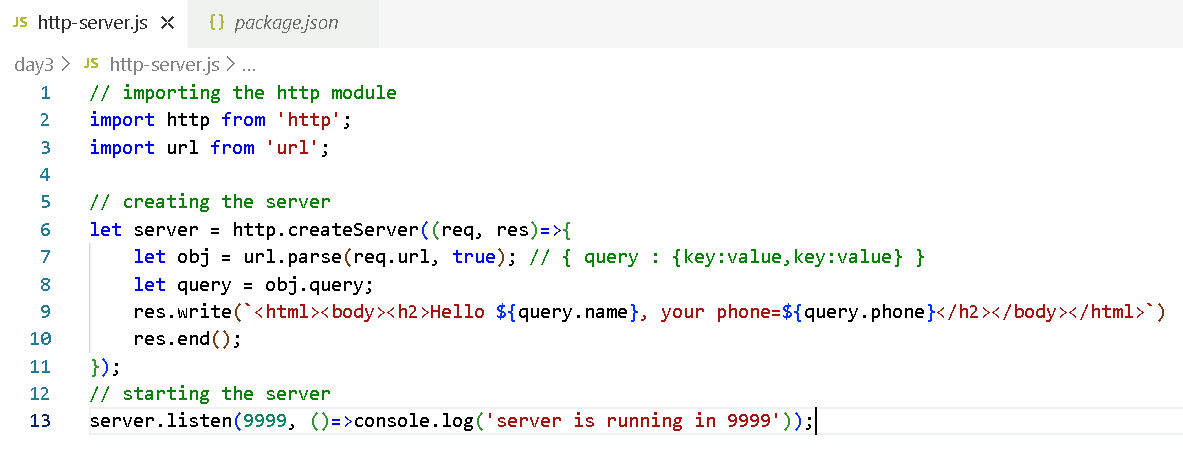
Query String: URL?key=value&key=value

let urlStr = request.url;

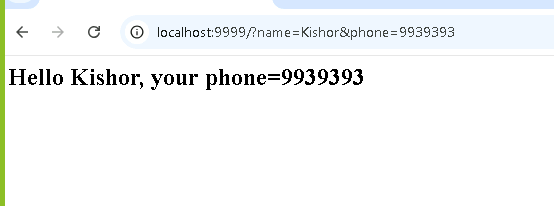
import url from ‘url’; // this module can parse the request url & convert into a form that maintains query string in an object

let obj = url.parse( request.url ); // { query: { key: value, key: value} }

http-server.js



Output:



HTTP module is very limited

It is a core module of node.js that allows you to just create a basic HTTP server, it can only handle requests & responses, however if you want routes to be done like having logics for login, registration, update, delete and etc then you must write everything inside the callback function and separate the URL’s for different operations, which will become tedious

ex:

URL/login  
URL/register  
URL/update

then your code will be like this in a single callback function

let url = request.url;  
if(url = ‘/login’) {   
} else if (url = ‘/register’) {   
}

HTTP need to have lot of manual setup for parsing the request data like parsing JSON or adding middleware’s (like CORS handling)

To avoid these problems, we have a Web framework built on top of HTTP i.e., Express

Express:

It is a web framework built on top of HTTP, it handles everything what HTTP module does and many advanced requirements like

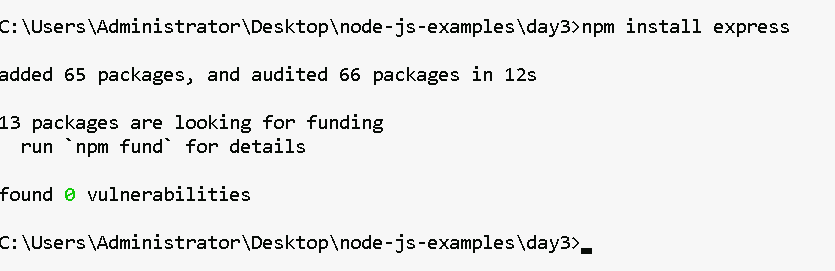
* Routing: handling different URL for different purposes using Javascript methods

ex: /login -> function, /search -> function, /register -> function

* Middleware support: Handling CORS, parsing the request data in a simple statement
* Integration with View engines: This makes your express to handle static files that is served to the client so that client can use that static files to fill out the details like input box, forms and many more, this is possible by using some view engines like PUG, JADE

Note: You can always use Express to create a feature rich web application and APIs

Note: You don’t get express in node.js by default, you must install from npm.

Creating an express server

import express from ‘express’;

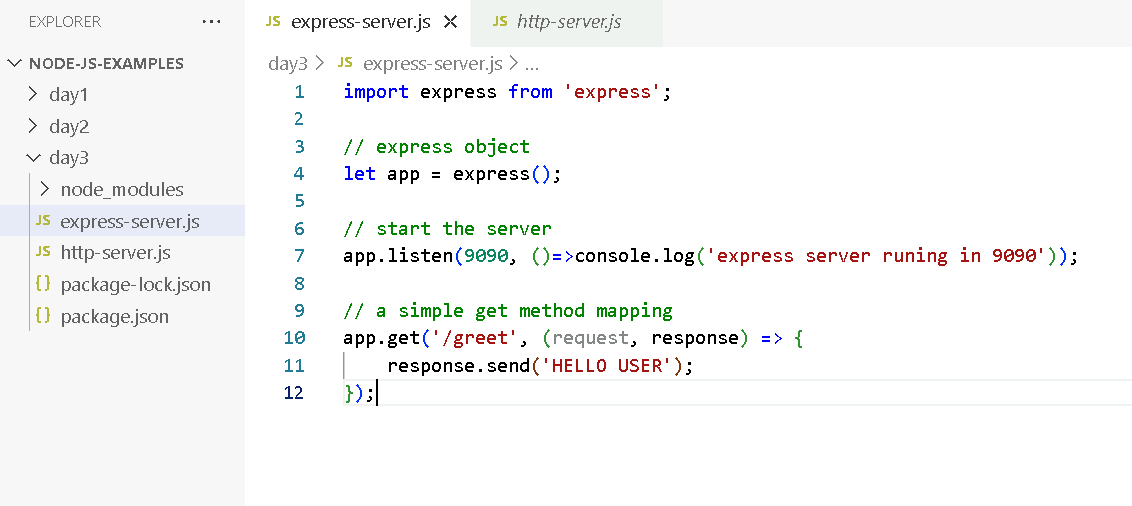
Creating an object of express

let app = express();

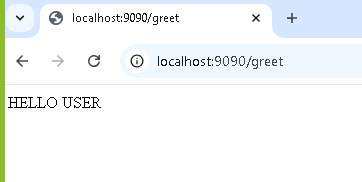
The express object provides functions like

get(), post(), put(), delete(), use(), listen() and so on

express-server.js



Output:



Template engine:

In Express a template engine is used in-order to generate dynamic HTML pages by supplying the data into the templates, this helps in separating the Presentation logic with Business logic

Some of the template engines are:-

1. PUG (formerly known as jade)
2. EJS
3. Handlebars

Installing the template engine

npm install pug

You must set the template engine in the express

let app = express();

app.set(‘view engine’, ‘pug’)

index.pug

html  
 head

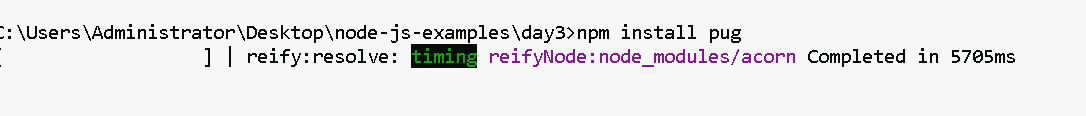
title

body

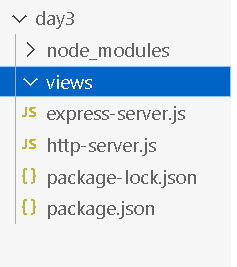
h1 = name

h1 = phone

Installing pug



Create views folder in the project



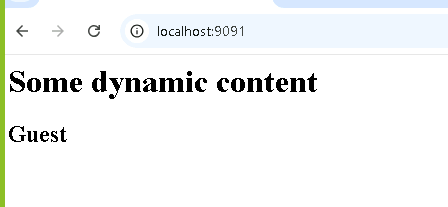
views/index.pug



app-template.js



Output:



Express.js is more widely used to develop API’s or Webservices

What are webservices?

These are online services which can share data over the internet to various applications and also can accept data from various applications.

Webservices are of two forms

1. SOAP based webservice – Simple Object Access Protocol – data exchange in only via XML
2. REST based webservcies – Representational State Transfer – data exchange can be in various formats like xml, json, csv and so on.

Note: JSON is more widely used format now a days

Express.js helps you to create REST based webservcies and REST based webservices have some guidelines in it

1. Resource locator: These are the webservices locator which will be URL’s the client programs must use
2. HTTP protocols: Consumer & Producer must use HTTP protocols & methods to communicate, so that both will be having standard methods to perform operations

Consumer & Producer must adhere to the HTTP protocols & their methods

Below are the standard methods HTTP provides

* post: It must be used when a resource is created at the backend
* get: It must be used when a resource need to be read
* put: It must be used when a resource needs to be updated
* delete: It must be used when a resource needs to be deleted

Examples:

1. OLA/UBER applications can get the data from the GOOGLE Map webservice
2. Using phone pay/ google pay to transfer amount to various banks

Testing tools for Webservices

* Postman
* VS Code – thunder client plugin
* SOAP UI

Our first webservice program

let app = express();

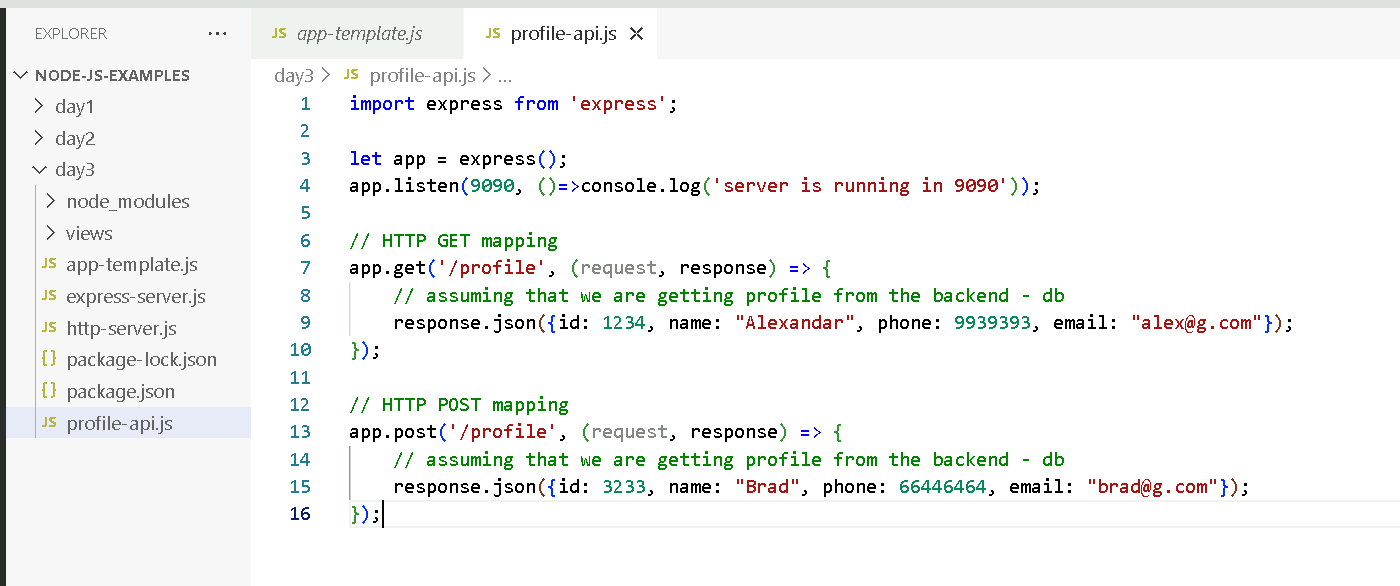
app.get(‘/profile’, (request, response) => {

// some logics to call the business logics & db logics  
 // producing the response in json format

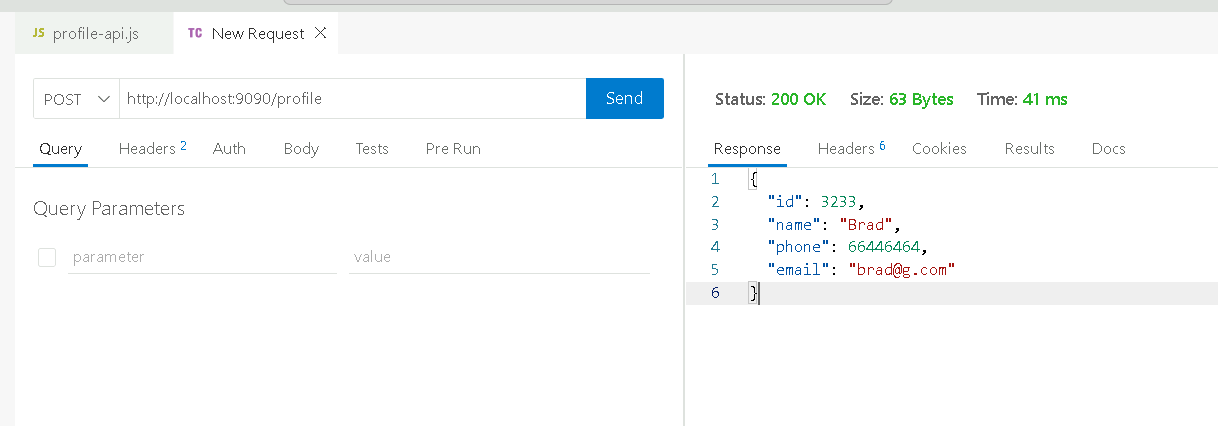
response.json( javascript object );

} );

profile-api.js



Output:



Sending the data to the webservice from the client program

A webservice must able to read the data from the client program in following ways

1. Path Parameter
   1. Request Body

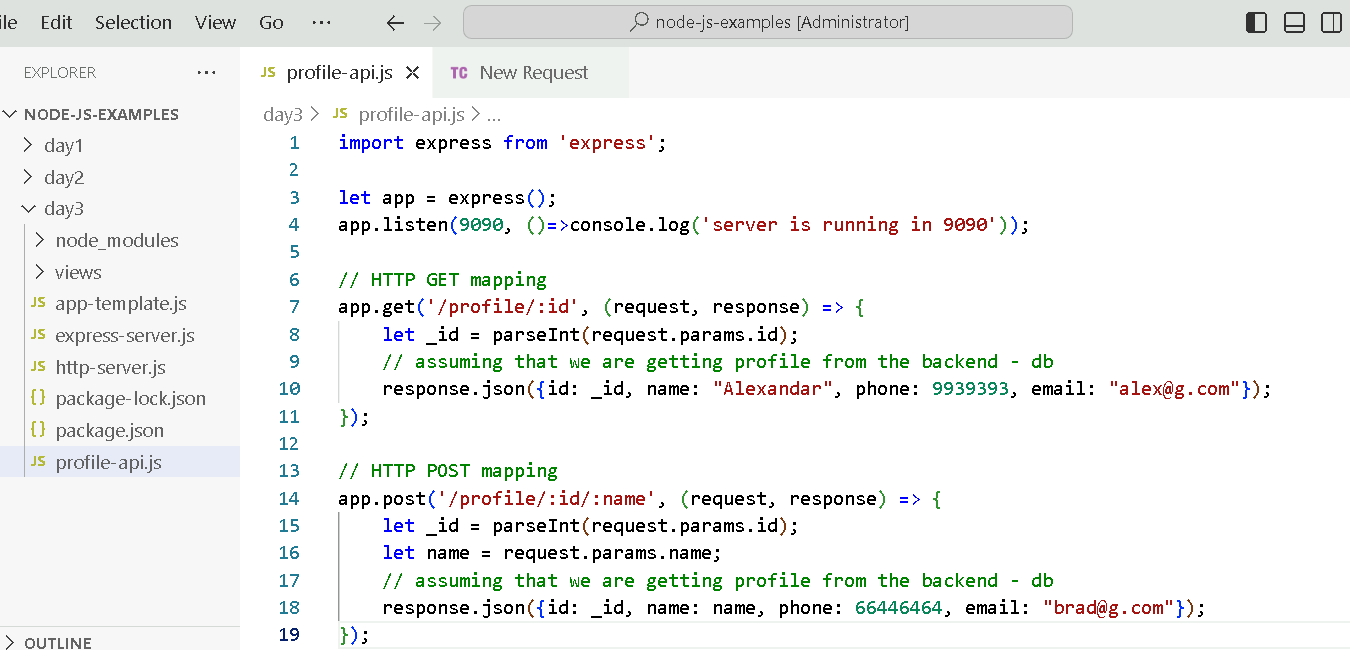
Path Parameter: You will have a webservice whose URL can be parameterized so that it can match to some values

/profile/:id can now handle following paths

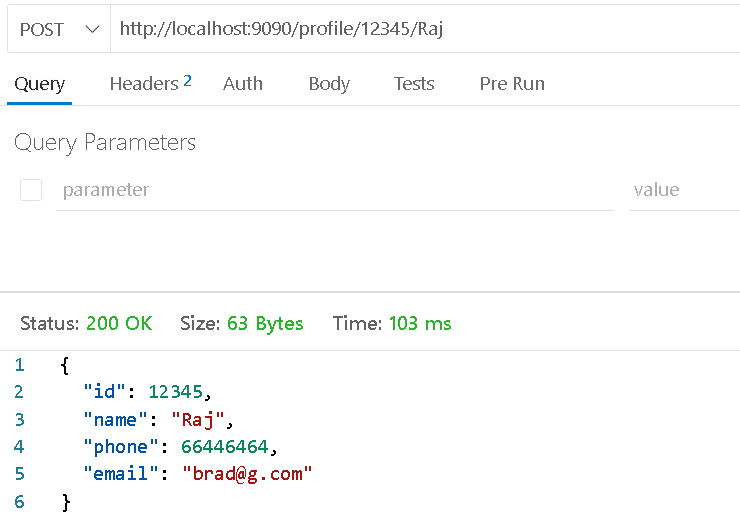
/profile/1

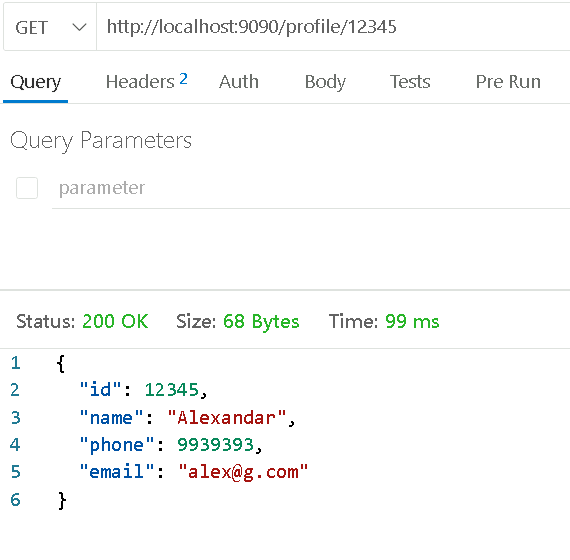
/profile/2

/profile/3 and so on.



Output:





Download in cloud labs

1. Mongodb community server – zip version 7
2. Mongodb shell – latest version – zip format

MongoDB stores the data in JSON format, it understands javascript

There are two command we will be using

1. mongod – starts the database & maintains the data in C:\data\db folder
2. mongosh – shell to interact with the mongodb

Collections: These are like tables that stores JSON documents, each documents will have \_id property to uniquely recognize the document

Inbuilt functions available in mongodb that you can use in the mongo shell

db.createCollection(“some-name”); // in mongodb collection names are case-sensitive

db.collection-name.insertOne( { json } ); // store the document

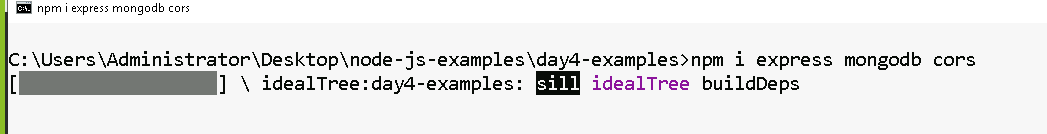
db.demo.updateOne({\_id:1}, {$set:{“name”:”XYZ”}});

db.demo.deleteOne({\_id:1});

Creating REST API’s that interacts with MongoDB

Libraries required

* express
* mongodb
* cors (to enable cross origin resource sharing)



Establishing connection to the Mongodb can be done either using then & catch or async await

import { MongoClient } from ‘mongodb’;

const DB\_URL = “….”;

1st way of working with promise

MongoClient.connect(DB\_URL).then( (con) => {  
 con.then( (result) => result.insertOne({}));  
} ).catch( (err) => {   
 DB ERROR  
} )

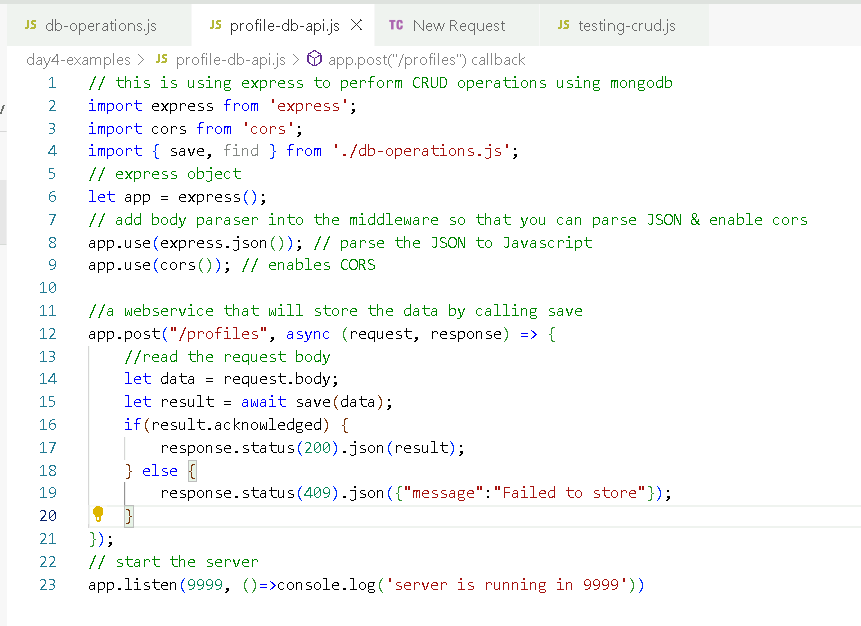
2nd way of working with Promise

let con = await MongoClient(DB\_URL)

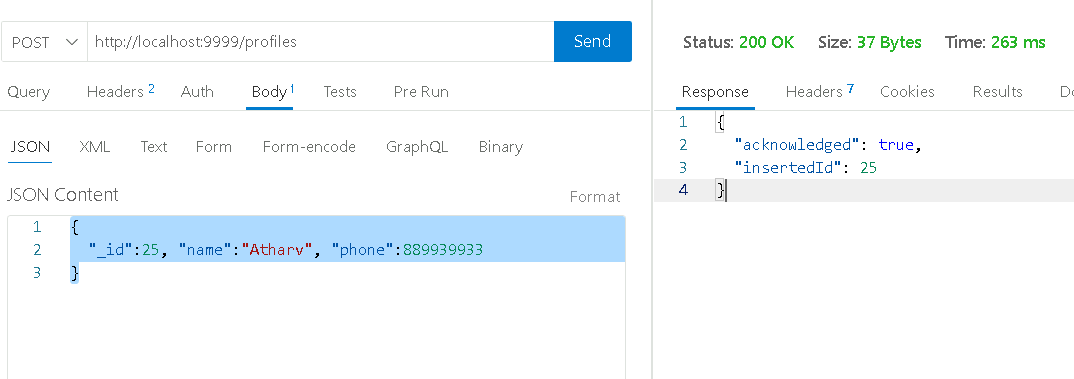
let result = await con;  
result.insertOne( { } )

Now we can create REST APIs that would interact with the mongodb

profile-db-api.js



Output:



Activity:

1. create an api /profiles/:id and produce a json response that will show profile in json if id is found else show the error message in json
2. create an api /profiles/:id/:phone and produce a json response that will update the profile phone in the existing document based on the \_id, , if updated show the updated document in the json format, if \_id is not present show the error message in json format