React.js and Node.js

Pre-requisites:

HTML

CSS

JavaScript

Software’s Required

1. Editor - Visual Studio Code : <https://code.visualstudio.com/download>
2. Node.js - Runtime environment for Javascript to run at the backend: <https://nodejs.org/en/>
3. Browser - Runtime environment for Javascript to run at the frontend
4. Internet - Open internet to download node-modules

Things to know before going to React.js & Node.js

* HTML
  + div, p, h1,..h6, table, form, img and so on.
* CSS
  + id, class, properties & values
* Javascript
  + Fundamentals
  + Functions & Variables
  + New Features of Javascript (EcmaScript latest features)
    - classes, let, const, super, extends
    - Arrow functions, template string literals
    - Rest & Spread operators
    - Default arguments
    - PadStart & PadEnd
    - Exponential operators
    - Generators

Refresher on JavaScript

There are mainly 3 ways you can print the output in the Javascript

* document
* alert
* console

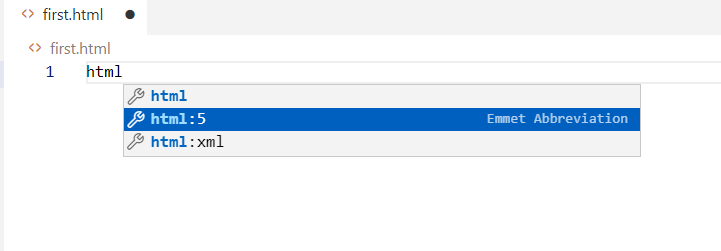
document: It is an inbuilt Javascript object that helps you to access the browser DOM (Document Object Model), it helps to access various functions like write(), getElementById(), getElementsByTagName() and so on.

alert: It is mainly to show the pop-up box, for any kind of warning or some messages

console: It is mainly to debug the javascript if you want to see any output you use log(), or if Javascript has any errors those are logged in the console.

first.html

You can use html in the beginning to see the html default template in the VScode which is auto-created



Then you see as





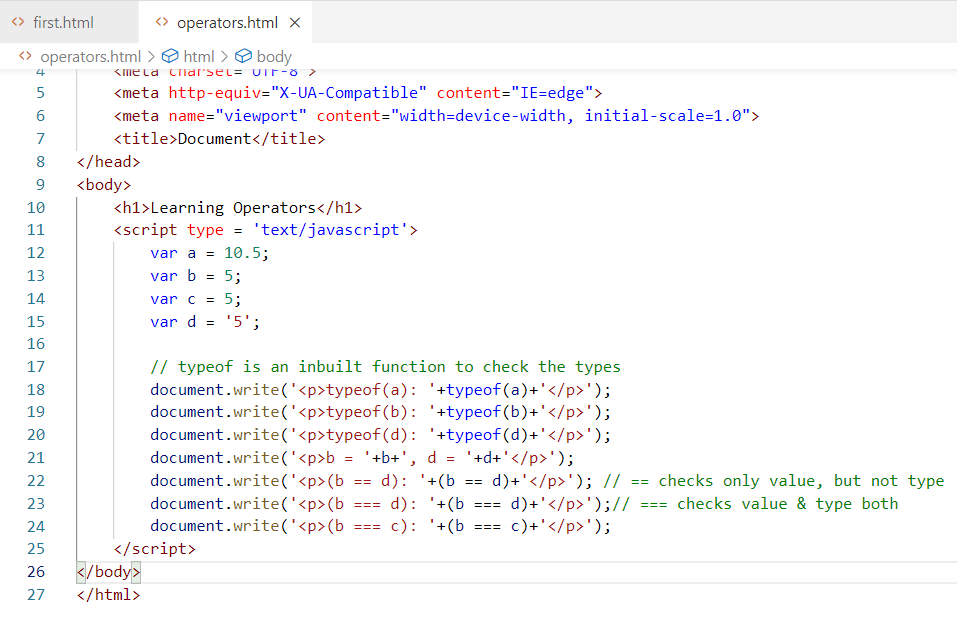
Output:



Fundamentals of JavaScript

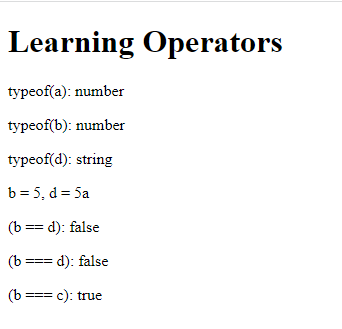
* operators: ++, --, \*, /, +, -, =, <, >, <=, >=, ==, ===, !=
* arrays
* loops: for, while, do-while
* conditions: if, if - else, if else if, else, swtich
* functions
* objects
* events

operators.html



Here typeof checks the data type, in Javascript we have datatypes like number, string, boolean, object, Date,

Output:



Activity:

1. Try out other operators like <=, >=, !=, ++, -- to understand how they work

Arrays: It is a collection of data, in Javascript you can have arrays with various types in a single collection, it used by []

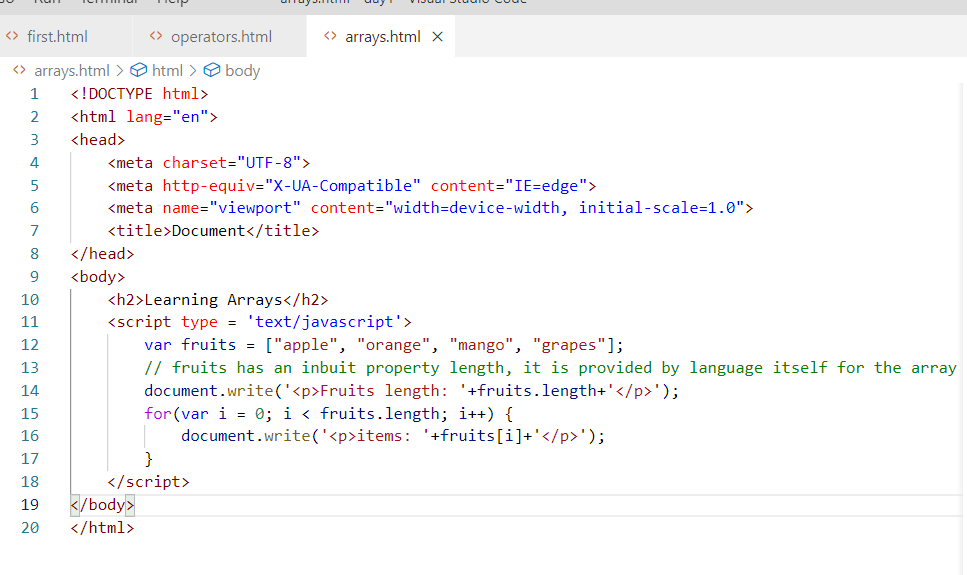
items = [2, 1, 6, 7]; // valid

items = [“hello”, “test”, “demo”]; // valid

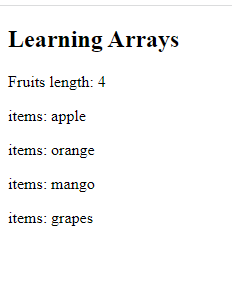
items = [2, “test”, “demo”, true, 5.15]; // valid

You can always use loops to iterate the array

arrays.html



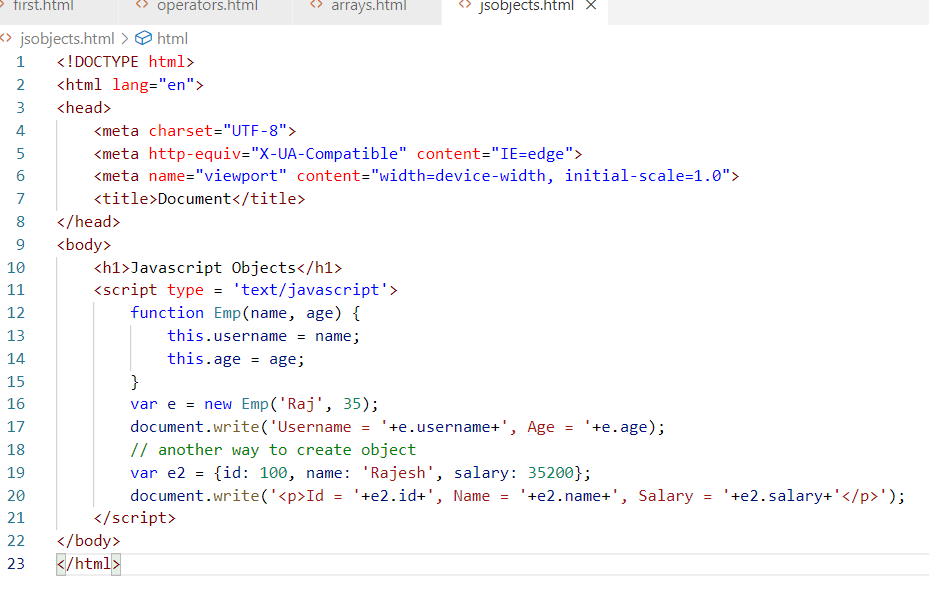
Output:



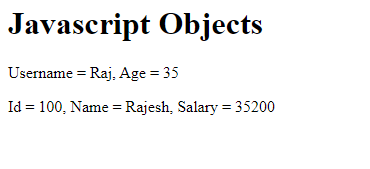
Creating Objects in Javascript

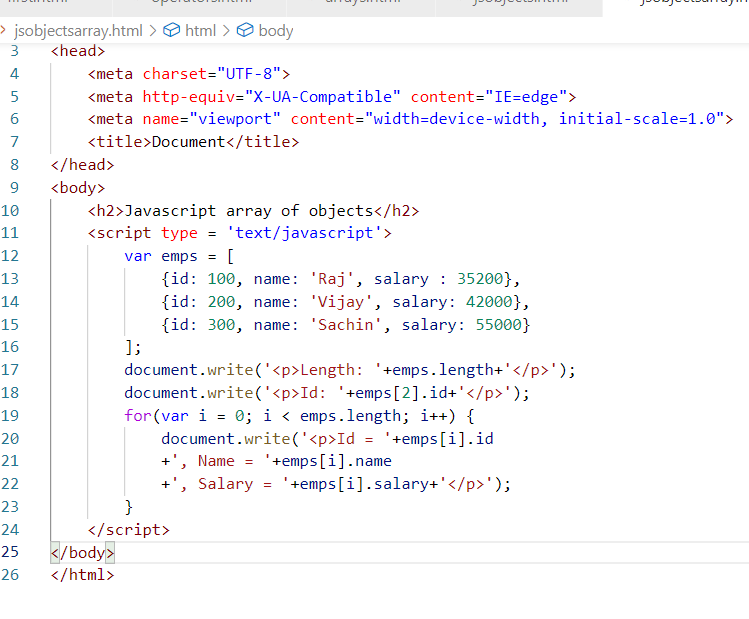
There are two ways you can create object in Javascript

1. Creating an object with {} having properties & values
2. Creating a constructor function that initializes the object

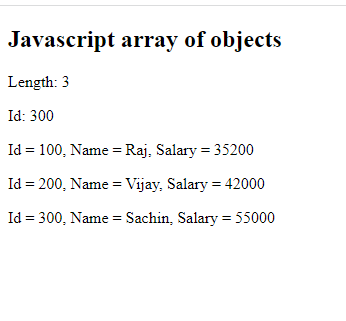


Output:





Output:



How to handle events:

Events are the things which are generated when user does some action on the HTML elements like click, mouseover, submit, change and so on.

In Javascript we can handle this event using the event attributes and their corresponding handler called function to handle.



Output:



There are many ways to get the element

1. event.target
2. document.getElementById()

Activities:

1. Try out all the above examples
2. With your own examples try different operators like <, >, <=, >=, !=
3. Try while & do while loops in Javascript and check out how it works
4. Display the array of employees in a table having rows & columns, use <table> tag to create tables and use the array having employee objects as below

 var emps = [

            {id: 100, name: 'Raj', salary : 35200},

            {id: 200, name: 'Vijay', salary: 42000},

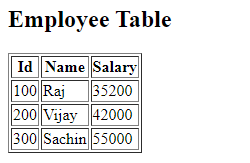
            {id: 300, name: 'Sachin', salary: 55000}

        ];

Solution:



Output:



ECMAScript (European Computer Manufacturing Association):

It is a standard that specifies set of features so that their specifications can implement, ECMAScript specifications are

* Javascript
* Typescript
* JQuery
* Node.js
* Browser
* JSX

ECMA Script is also called as ES it has released lot of new features from V6 onwards till now, V6 was released in 2015, all these new features are understood by browser, node.js and even Javascript.

New Features of ECMAScript

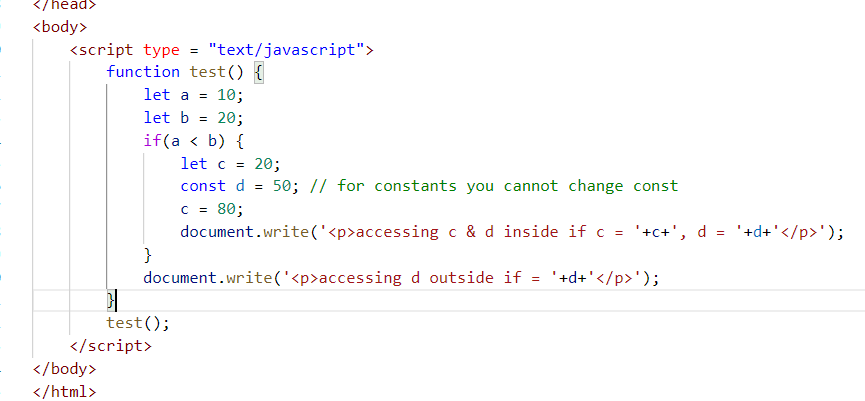
These simplifies writing Javascript code

1. let, const, class, extends, super, constructor
2. Template Strings
3. Rest, Spread & Default Parameters
4. Arrow Functions
5. Destructuring
6. padStart & padEnd
7. Exponential Operator
8. Optional Chain
9. Object entries & values
10. static keyword

let & const

These are the two keywords you can use to declare variables in Javascript apart from var, let & const creates scopes for the variables.

var is always global, means you create variables with var it will be considered as global variable it is not part of any block of code, sometime you want to protect your variables within a block of code then through var its not possible, but possible through let & const



const variables can’t be modified

Output:



Callbacks: These are functions that are called after some time but not immediately, this is mainly used to perform asynchronous operation

function abc() {   
  
}

abc();

x = 10;

y = 20;

abc();

z = 30;

The above statements are synchronous, every statement waits for their previous statements to complete first

Asynchronous calls are the calls that doesn’t block the next statement to execute, the asynchronous calls may happen at any time but other statements doesn’t wait for asynchronous call to complete, it can be achieved only by callbacks

x = 10;

asynch call

y = 20;

z = 30;

asynch call

a = 50;

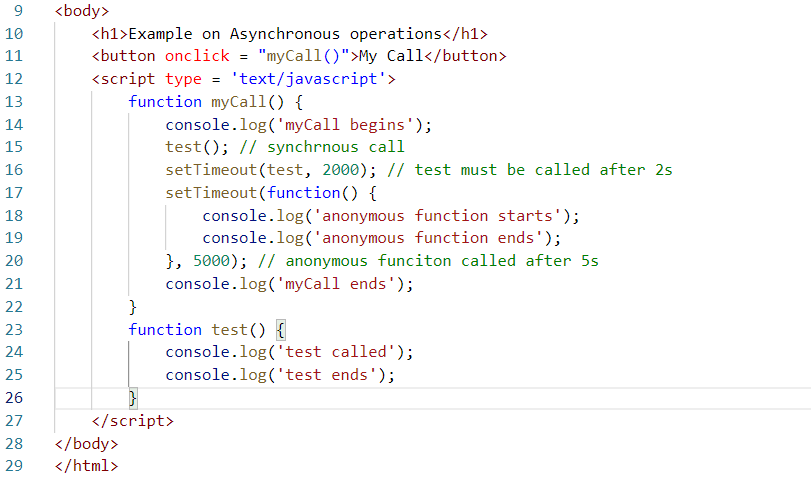
Asynchronous calls don’t block you to do the next operation, these calls may give response late but user doesn’t need to wait for the response to perform the next operations.

ex: Google search let user enter input and it sends each input to the server to fetch the response, but user can continuously enter the input without waiting for the response their previous request

Callbacks are the special kind of functions they are called separately to make operations asynchronous & we wouldn’t be having idea when they are going to complete.

Best example is we can use setTimeout(callback, time);

setTimeout: It is a function that takes callback as an argument, i.e., function as an argument which is called after a specific duration mentioned, but the other statements after setTimeout() is not blocked for callbacks to complete.



Here the 15th line is executed and after it completes then the 16th line is executed once it completes then 17th line is executed, but the callback passed as an argument at 16th is not going to block 17th line to begin the execution, they are callbacks which are executed once the specific duration mentioned is completed, here the callbacks passed to setTimeout at 16th line & 17th line doesn’t block 21st line at all, because these callbacks are executed later without blocking 21st line.

Note: setTimeout is not an asynchronous function, it takes the callback to execute after some time, that callback is an asynchronous.

Note: Not all callbacks are asynchronous

When to use asynchronous functions or callbacks

Suppose you have a script that has statements which needs to fetch the data from the server then the script doesn’t need to wait for the data to be available, because we don’t know when the data will be available, till that time you can’t block the script from execution, hence the fetching data must be Asynchronous, similarly lot of File IO operations can be asynchronous.

There are some callbacks which are not asynchronous, we can still write them without naming the function i.e., anonymous functions.

ex: forEach(), map() and so on.

Two ways of iterating the arrays using callbacks

1. forEach(): it iterates the array & invokes the callback for each iteration and callback will have 2 arguments 1st one is item that is iterated & 2nd one is the index of the item.

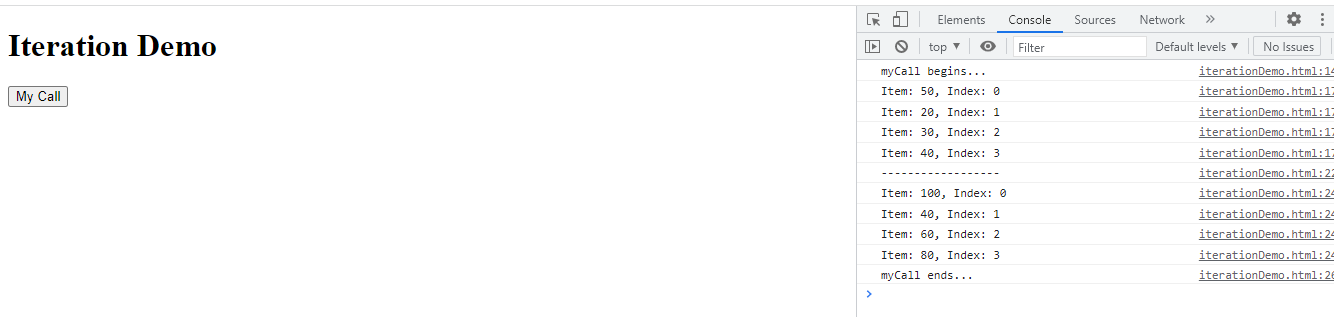
Note: callback argument names can be anything, but we can use item, index to understand

1. map(): it iterates the arrays & invokes the callback for each iteration but returns the new array, it is used to iterate and perform some operations on the iterated item, it is used when you want to convert data to another data

Note: the callback of map also takes 2 arguments similar to forEach



Output:



Arrow function:

It is a new feature of ECMAScript that simplifies writing anonymous functions or callbacks, it doesn’t need any function keyword and doesn’t need body of the function to have {} if its one line statement, doesn’t need return if its one line statement

They are written with () => statement

Some of the way of writing arrow functions

() => 20; similar to function() { return 20; }

(x, y) => x + y; similar to function(x, y) { return (x + y) }

() => console.log(‘hi’); similar to function() { console.log(‘hi’) }

If arrow functions need to be written with morethan one line they you need { }

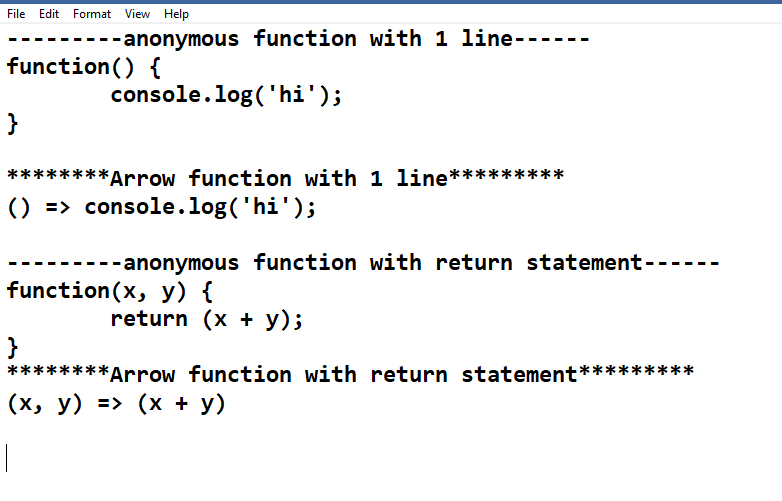
() => { console.log(‘hi’); return 20; }

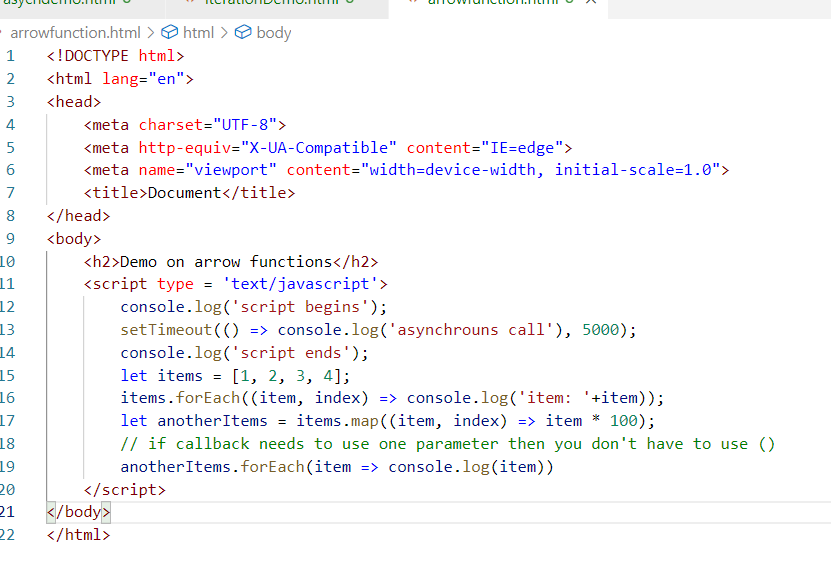
The above code is similar to

function() {  
 console.log(‘hi’);

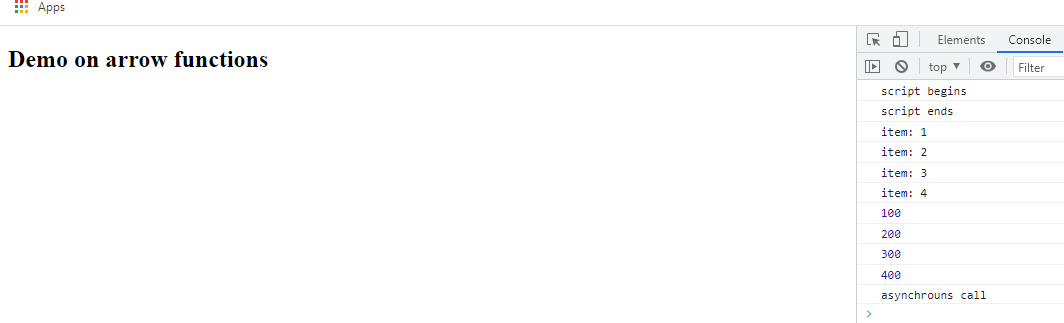
return 20;  
}

The arrow function needs to use return statement mandatorily once {} are used





Output:



ECMAScript new features like Rest, Spread operators

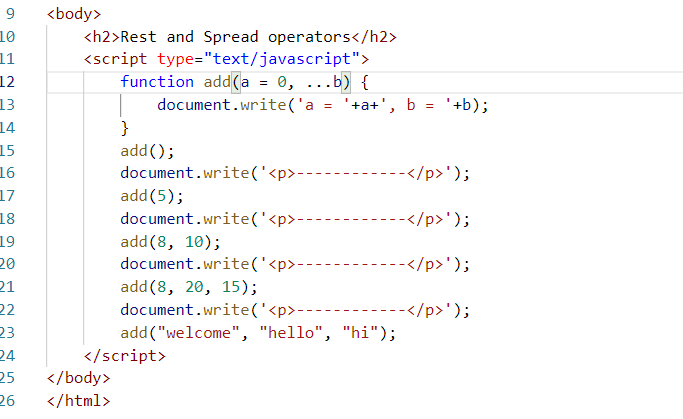
These two solves certain problems while passing arguments to the functions while calling.

Rest: It is a parameter that can accept 0 or more number of arguments, it is going to take rest of the data while accepting the data

Spread: It is to share the data to multiple parameters of a function, it is going to spread data to multiple parameter of a function

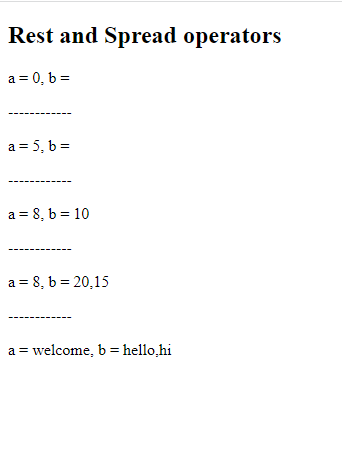
Rest operator:

It is a parameter that accepts 0 or more values and it needs to be at the end of the function parameter, if in case a function has more than one parameter



add(a = 0, …b) means a accepts 0 if in case add() is called without any value to a, b anyways by default will be an empty array, you can always iterate rest parameter variable using for loop.

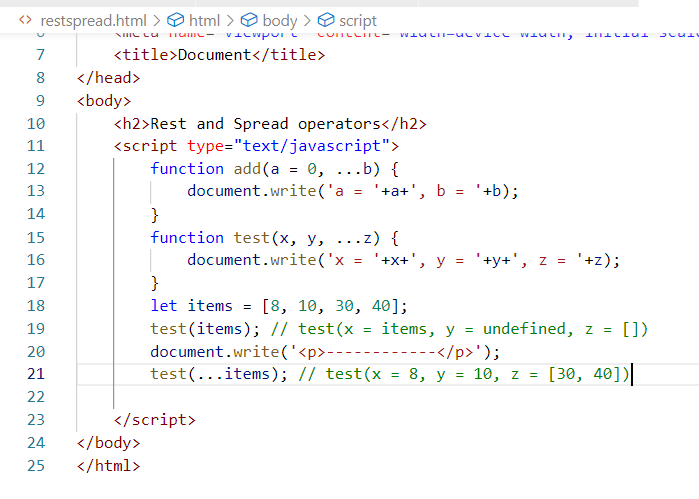
Output:

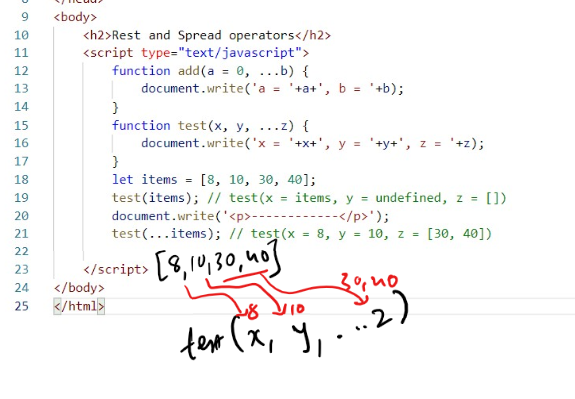


Note: In Javascript a variable can take any kind of value, there’s no specific that it accepts only a specific type of value

Spread operator

When you have multiple values in an array you can spread them to a function matching to multiple parameters

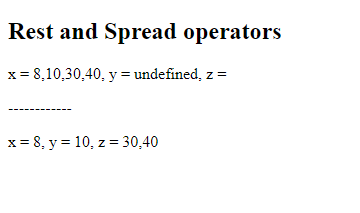




Here the 19th line is accepting the array hence the test(x, y, …z) will accept the array for x variable, but y & z doesn’t get any value

But 21st line is using spread(…), hence the values of array is spread to the parameters of the function that matches to the number of elements in the array, since items has 4 elements & test() accepts 3 arguments and the last one is rest operator, then the 1st two elements are assigned to 1st two parameters i.e., x = 8, y = 10, z = [30, 40], whereas z is rest operator hence it accepts rest of the elements.

Output:



The above code uses 3 new features of Javascript

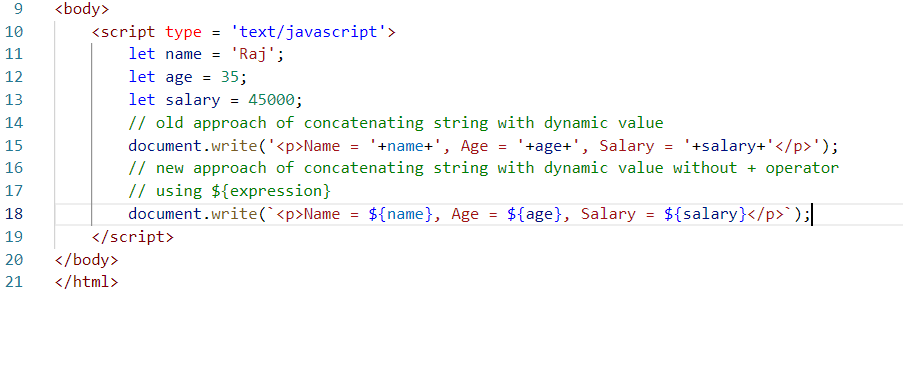
1. Rest Operator
2. Spread Operator
3. Default argument

Default arguments: It is an argument to a function if the value is not supplied then the default is taken

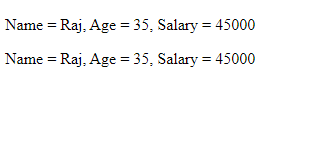
function test(a = 0, y = 0){ } if test is called without values then a & y will be 0, if test is called by passing values like test(20), then a will be 20, y will be 0, if its called test(20, 30) then a = 20, y = 30.

Template String literals

It is used to create strings with dynamic value without breaking the strings with + operator, because when you concatenate the strings with dynamic value you need to break the string with + operator and concatenate the dynamic value, to avoid that we have a template string literals.

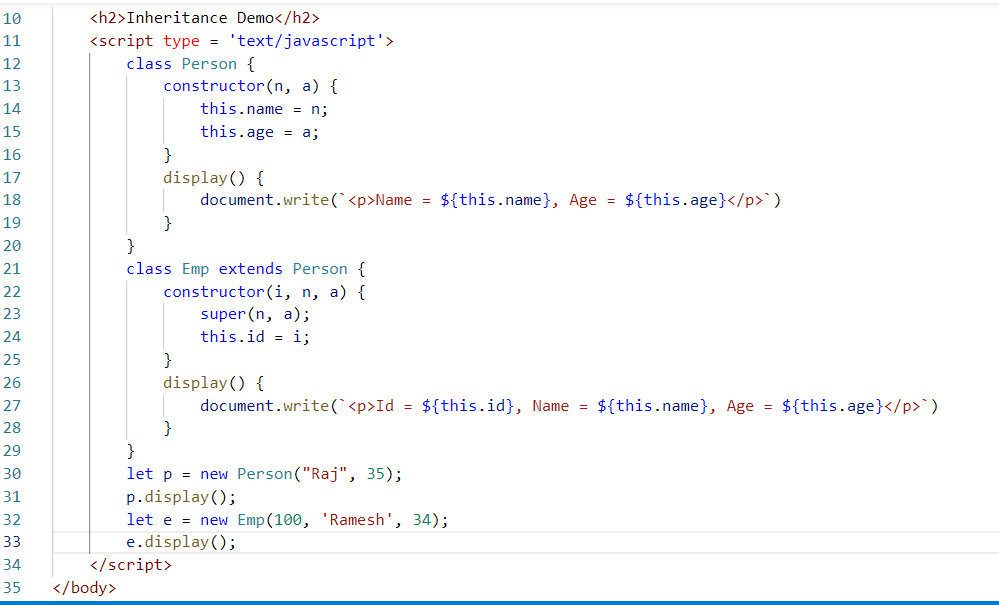


Output:

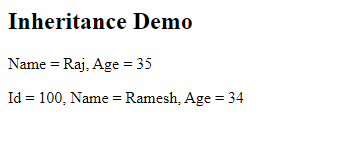


class, constructor, extends, super keyword

These are the new keywords added in Javascript to easily create the functions part of the object & also inherit the object from another object



Output:



ECMAScript new features like exponential operator, trailing commas, optional chain

Exponential Operator: You can easily find the power of 2 numbers using \*\* now, suppose you write 2 \*\* 3 then it returns 2 power 3 i.e., 8, if you use 3 \*\* 3 then it returns 3 power 3, i.e., 27

Trailing commas: Javascript ignores the extra commas if in case you did a mistake while creating arrays, earlier it was an error, but now Javascript ignores it as extra commas are trailed ie., removed

Suppose you write [10, 20, 30, ] here after 30 an extra comma is automatically removed.

Optional Chain: It allows you to access a nested property with a condition like if exists access else don’t access, earlier developers need to write many conditional statements to check the existence of nested properties

i.e., users = [{id: 100, name: ‘Raj’, address: {state: ‘ka’, city: ‘blr’}, {id: 122, name: ‘Ajay’}]

Here users is an array, where first user object has address.state & address.city, however the second user object doesn’t have address property so we can use optional chain

Here when you iterate you usually iterate the item by accessing each property & their property, but chances are there you could get error, as address is not there in the 2nd object.

you can use optional chain in each iteration so that it access address if exists else doesn’t access.

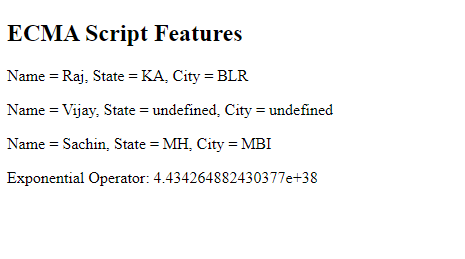
item.address.state : chance of error, if address is undefined

item?.address?.state: will access address if exists else not



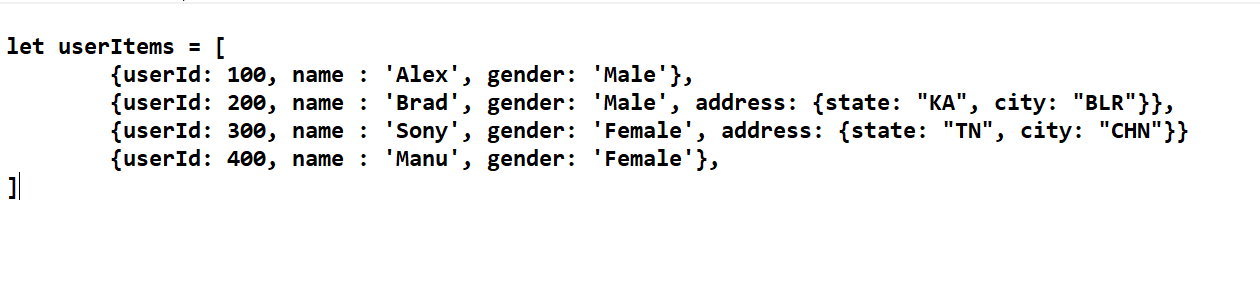
The 15th line has extra comma, which is ignored, the 19th & 20th line will access name and address only if they exists, so that you get no errors, 22nd line is using exponential operator.

Output:



Activity:

1. Using template string literal create table rows dynamically to show user information’s, use the below array as it and show the data in the table with border



Display the above user details in the table, the table must have headings and the data must be printed through Javascript not by hardcoding

Note: Use <div id = “root”></div> and below that create a button once you click on the button the table must be added inside the div#root

Hint: use map() function.

Expected Output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **GENDER** | **STATE** | **CITY** |
| 100 | Alex | Male |  |  |
| 200 | Brad | Male | KA | BLR |
| 300 | Sony | Female | TN | CHN |
| 400 | Manu | Female |  |  |

Summary:

* Javascript fundamentals
* Event Handling
* Object
* ECMAScript features
* let, const, class, constructor, extends, super
* Default, Rest & Spread parameters
* Template String literals
* Arrow functions
* Optional Chain
* Exponential operator

Node.js

It is a runtime environment to execute Javascript code without browser, you can write backend programs that can perform various backend operations like IO operations, DB operations, Request Handling and so on.

Node.js provides lot of Javascript libraries which are called as node\_modules using which you can perform any kind of operations.

Now you can use Javascript for both UI development as well as Backend Services i.e., Full Stack Development can be done with a single language

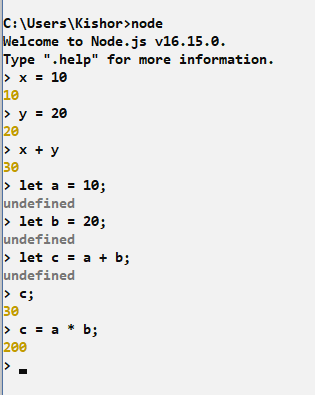
Node.js can directly run Javascript files and since it doesn’t use browser you can’t use some of the inbuilt objects & functions that are used for the Frontend UI’s like document, window, alert(), however Node.js can solve a different requirement like interacting with DB, File systems, OS resources and so on.

Benefits of Node.js

* Developers don’t need to switch to different language to write backend programs
* Enable developers to write entire application in one language i.e., Javascript
* JSON is the common format used to exchange the data between front end & backend applications & JSON is the native language to Javascript
* Many of the NoSQL database(MongoDB, CouchDB) store their data in JSON format, if node.js interacts with it then it will be easier for Node.js to understand their data
* Node.js gives you an environment which allows you to run the scripts without Javascript file i.e., node terminal

To open node terminal

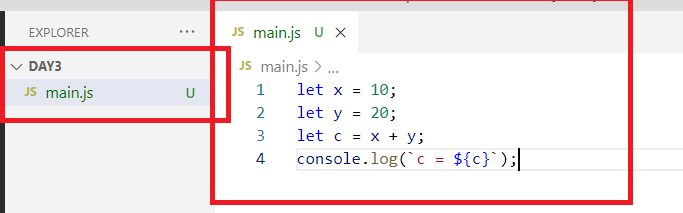
* In CMD prompt enter node
* To exit use Control + C



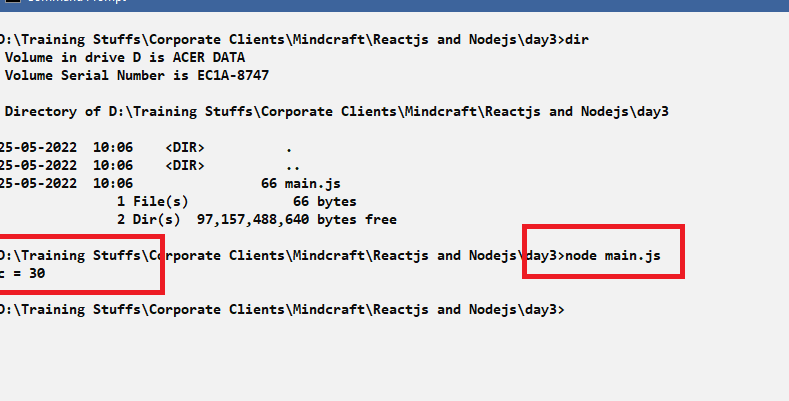
But these programs are not saved in any file, once you come out of the terminal these codes are gone, hence you need to create a JS file and run that using node command

You can create one file named app.js or index.js or main.js or anyname.js and you can write Javascript code & run using node filename.js

day3/main.js



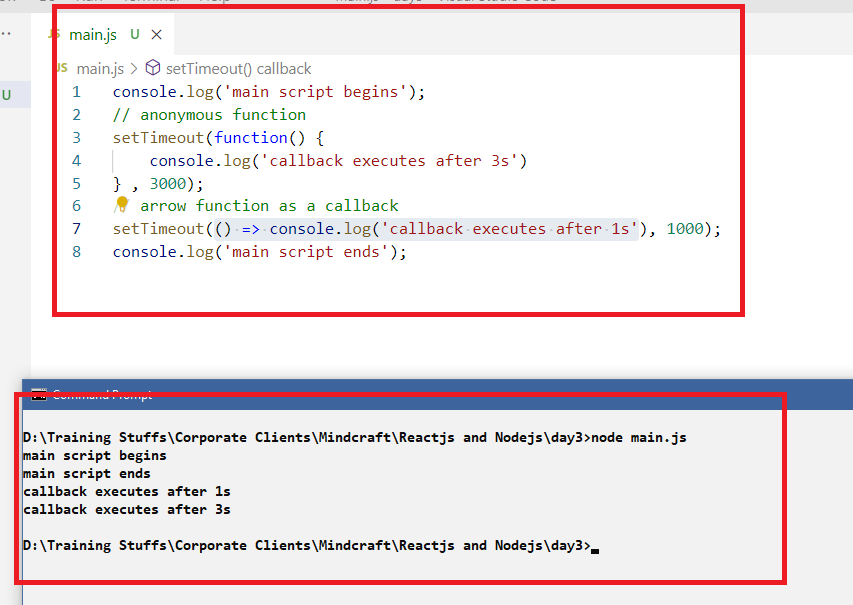
Output:



Note: While practicing create different filenames for each program like ex1.js, ex2.js, ex3.js and so on.

You can use asynchronous functionalities also in node.js programs that is callbacks that can be supplied to the setTimeout, setInterval kind of functions.

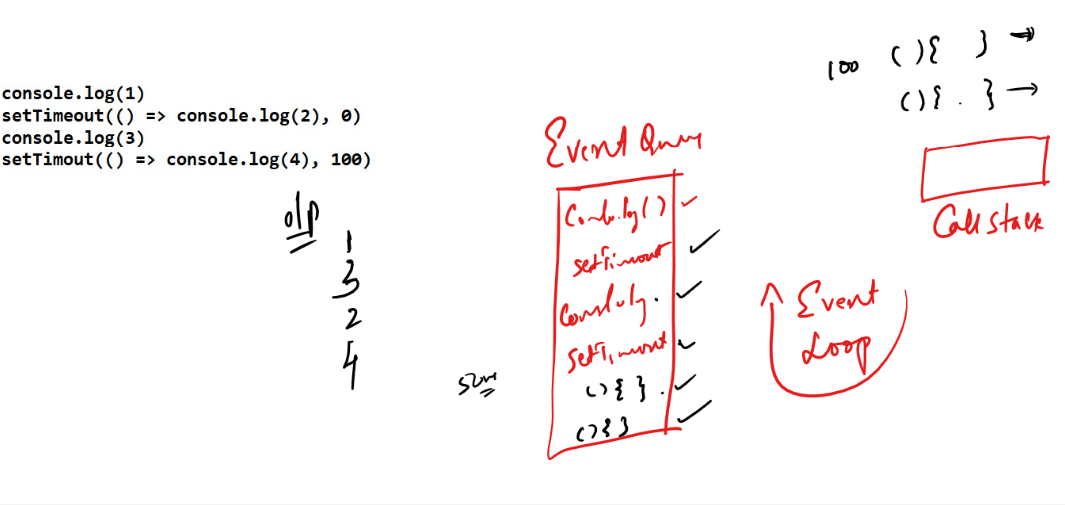
day3/main.js

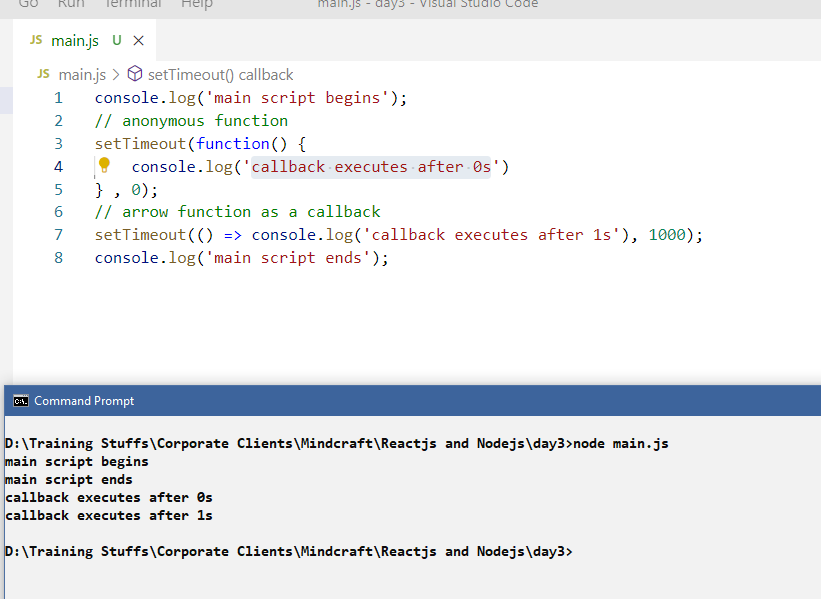


Node.js also doesn’t block other scripts when an asynchronous call is made, it uses a mechanism called Event Looping behind the scene

Event Loop in Node.js

Node.js uses a single thread to run the script, it is not multi-threaded, with single thread & event looping mechanism it can perform all the IO operations without blocking other scripts.





Modules: These are reusable code that can be used in any Javascript files to access their functionalities

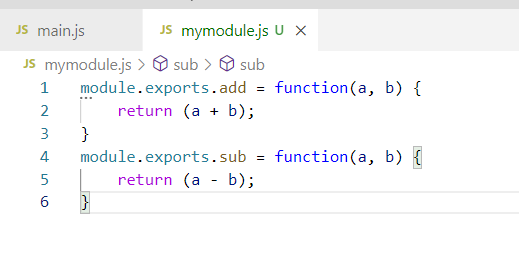
There are mainly 3 types of modules

1. Core Modules: These are functionalities available in node.js itself
2. Local Modules: These are functionalities which you can create and use it in any Javascript
3. Third party Modules; These are functionalities which are available from the internet which you can download and use in your script.

Local Modules

These are the modules we can create in our local machine and use them in any scripts

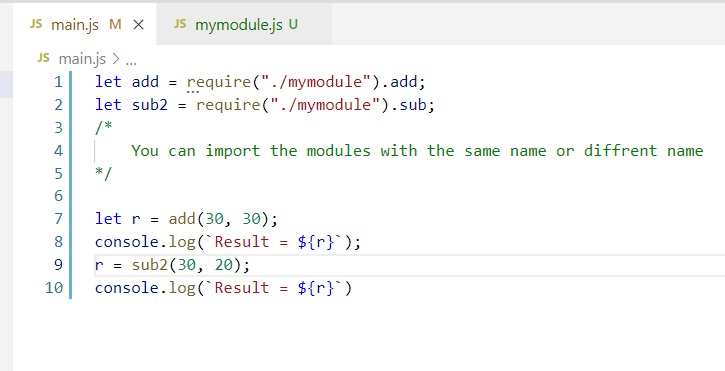
mymodules.js



Here module.exports helps your script to import these functions like add and sub

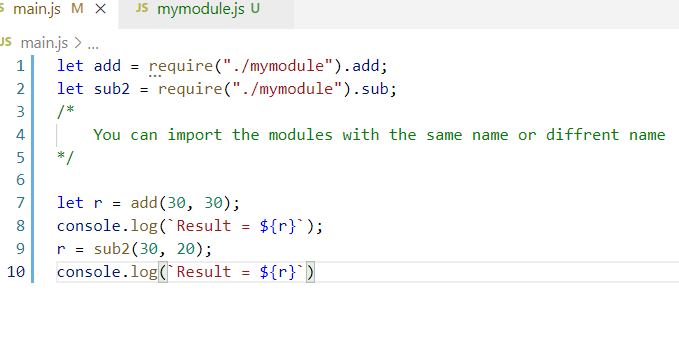
We need to use require(“”) to import these modules

main.js



Here require(“path”) need to mention only the path & the filename and file must not have any extension like .js, we can import the module with any names, but its better to keep same names to easily understand which function we are calling

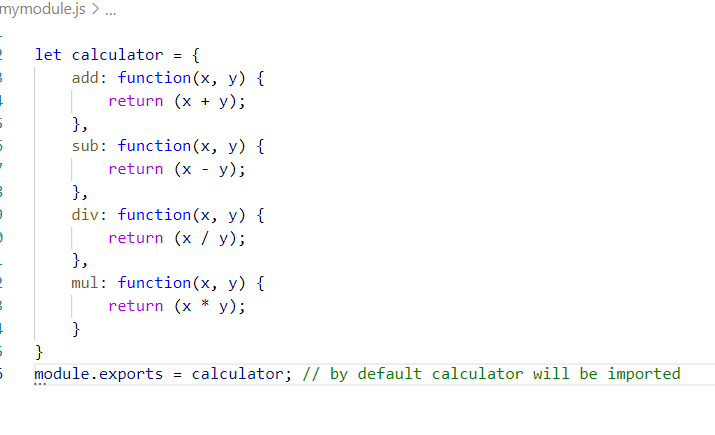
Output:



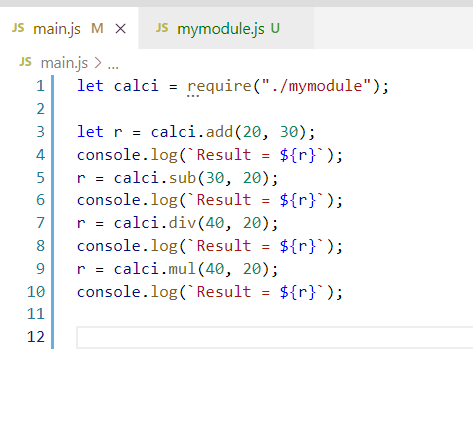
You can also import multiple functions from an object

You can use this to avoid multiple exports statement for each function name, this is good when you have all the functions in an object and directly export all the functions using the object name, you need to write the exports at the end of the file

mymodule.js



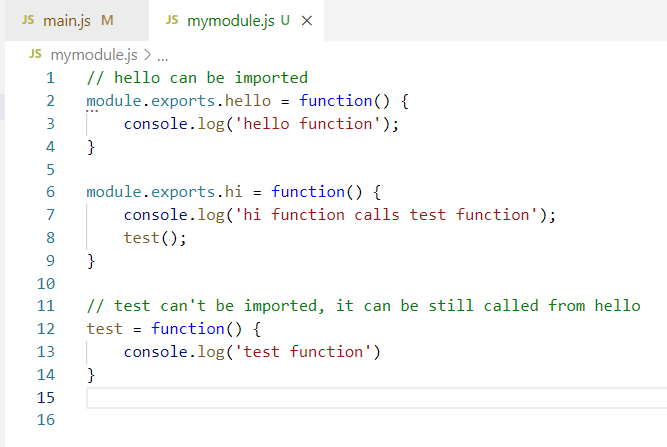
main.js



Note: the let calci = require(‘./mymodule’); imports the default module, hence you need to have only one default exports in a module, but you can have any number of named exports

Some functions can be made private to the module so that they can’t be exported

mymodule.js



Here you can import hello() & hi() but not test

main.js



Basically you have two types of exports

1. Named exports which will have export for each modules
2. Default exports which is used when there is only one module to export

Named Exports

module.exports.add = function() { }

module.exports.sub = function() { }

Default Exports

let obj = {…..}

module.exports = obj;

Note: You can have only one default exports in single Javascript file

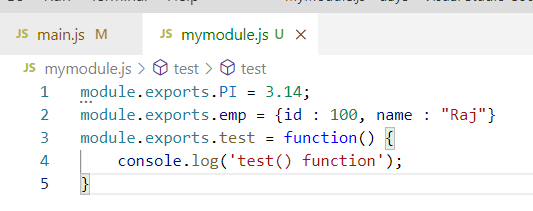
Use of default exports is you can together import all the features of the module

Named Exports vs Default Exports

|  |  |
| --- | --- |
| Named Exports | Default Exports |
| You can have any number of named exports in a module | You can have maximum one default exports in a module |
| You need to use the name and import the named modules  i.e., let a = require(“mymodule”).add  Here a is having add module | You don’t need to use the name at all while importing default module, it automatically imports default module only  i.e., let a = require(“mymodule”)  here a is having the default module |

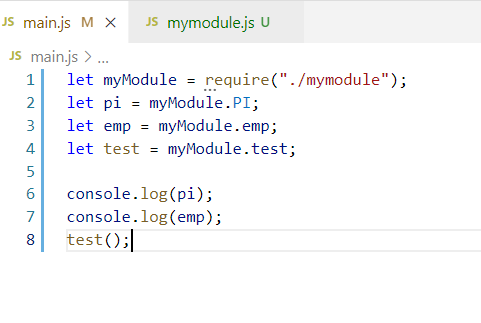
You can also export other modules like functions, objects, classes, variables

mymodule.js

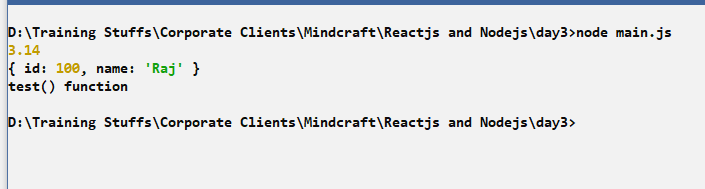


Here there are 3 modules which needs to be imported using the names PI, emp, test

main.js



Output:



Core Modules

These are by default available from the node.js which you can import and use them in the script

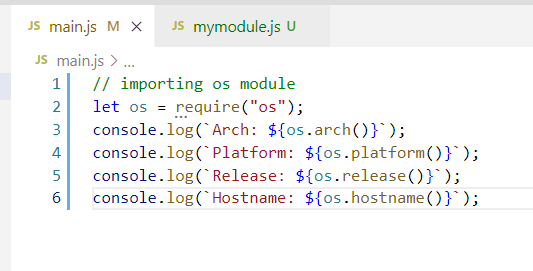
Some of the core modules are:

1. os
2. fs
3. http
4. readline
5. util

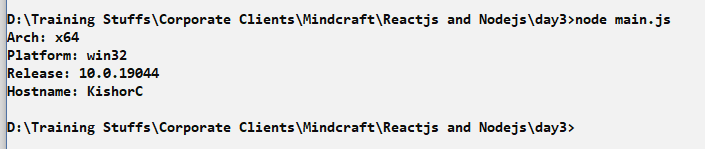
OS Module

You can use this to extract the details of your os like arch(), platform(), release(), type(), freemem()

main.js



Output:



We could see x64 for arch() as it’s the 64-bit architecture, we see platform win32 as it is Windows platform, and release number is 10, since its Windows10, Hostname prints the username.

fs: (File System) File Handling

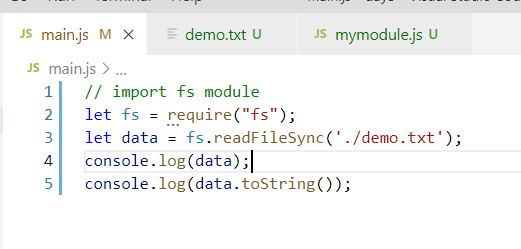
Node.js can perform read/write operations on files using a module called fs

This module provides functions like readFileSync(filename) to read the data and writeFileSync() to write the data to the file.

demo.txt

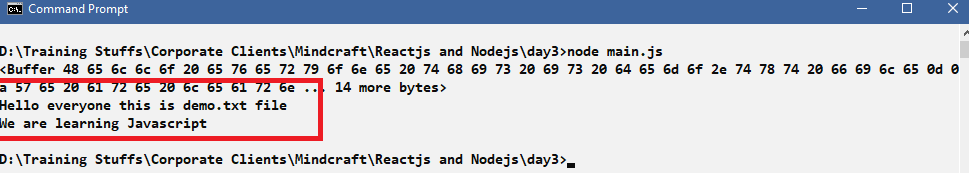
Write some content in demo.txt file

main.js



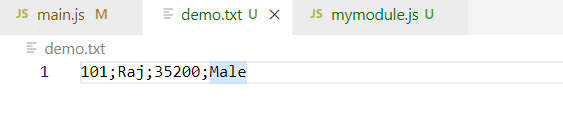
Here the readFileSync reads the file content and returns the data in Buffer format which needs to be converted to string format to understand what is the data.

Output:



You can also read the data having some details separated by some delimiter

demo.txt

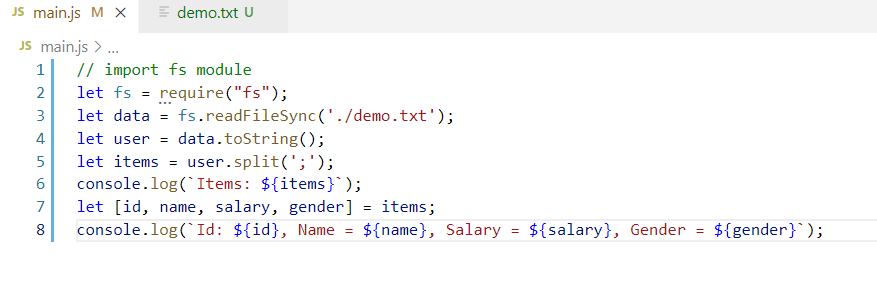


Now you want node.js to read the content as separate data like id, name, salary, gender

Since when you read the content it reads as a string, we can make use of one built function of string called split(delimiter), this splits the string into multiple strings based on delimiter

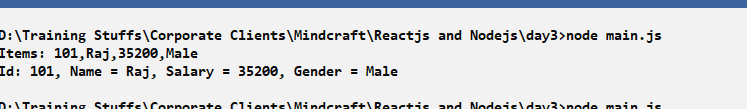
Suppose the data is Raj;Male;3000, then you can use data.split(“;”) then you get an array of string which will have [Raj, Male, 3000] which you can assign to variables either by using destructure or index based assignment

main.js



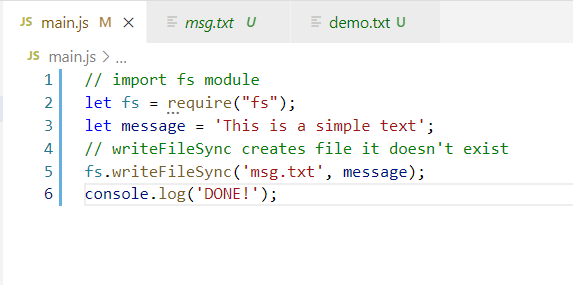
Here user will have the entire content of the txt file & split(‘;’) splits the content to multiple strings and stores in the array, since ; is delimiter we get 4 items in the array as we have 100;Raj;3000;Male it will be stored in an array as [100, Raj, 3000, Male], we have used array destructuring to assign each items into the variables

Output:

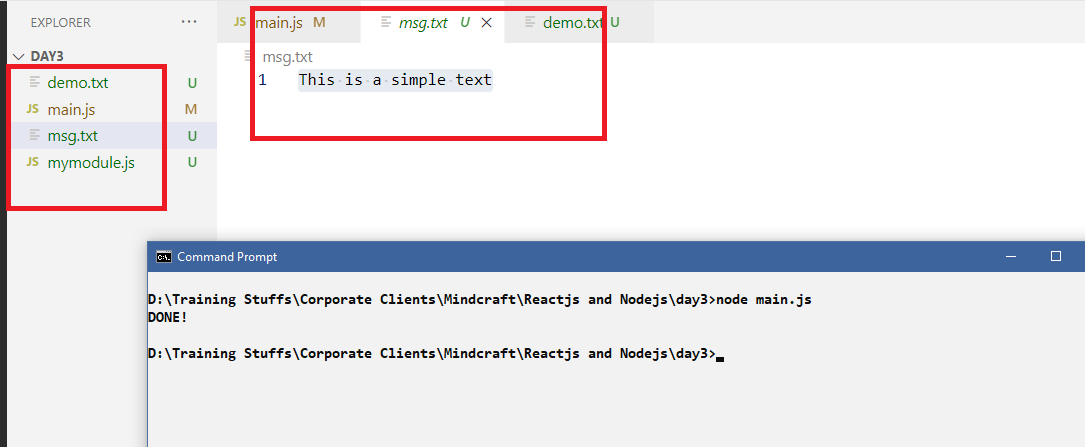


Writing data to the file

Using writeFileSync() we can write the data to the file



Output:

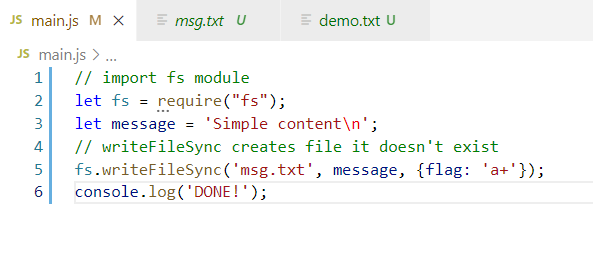


writeFileSync doesn’t automatically append the new content with the previous content, hence you must use a flag : ‘a+’} in the third argument of writeFileSync.

Whats the problem if we don’t use {flag:’a+’}

It erases older data and keeps only the current data you write, to avoid that we need to use {flag: ‘a+’} in the 3rd argument of writeFileSync.

main.js



Output:

You will see the msg.txt retaining old content because {flag:’a+’} makes writing appendable.

