Node.js: The Beginning

Introduction

We are now a partial web developer, but here comes the important part of development, that is, to learn how to render a website using a server, wherein data is permanently stored somewhere.

We will work on some exciting projects like building a social media website, such as Facebook. wherein you can store and render the data according to user preferences. We will implement features like - logging in and logging out, initializing MongoDB, adding friends, posting comments, etc. We will also implement a chat engine.

Getting Our Tools

Try Downloading the following tools to get started on your journey of you becoming a developer.

- Node.js {Latest Version (LTS - Long Term Support) } (Download it and install it).
- Google Chrome.
- Editor - {VSCode is highly recommended}

Setting Up the Tools

- Node.js
    - https://nodejs.org/en/ {Download the LTS and current version according to your operating system}.
    - LTS is suggested as they provide support if there are any of any sort of bugs present.
    - After getting downloaded just open the package and click on continue, agree with the license and simply install.
    - To check whether Node.js is available or not simply open the terminal according to your operating system and type {node -v} & if it gives the version it means you have installed your Node.js successfully.

- Google Chrome
    - Try downloading it using the given link. {https://www.google.com/chrome/?brand=YTUH&gclid=CjwKCAjwr_uCBhAFEiwAX8YJgXvbudxHdhmdtIRZB35qk_ydDJ7Yxi_iRbFuGO2YvC7GHwcNJkhbPRoCp0QQAvD_BwE&gclsrc=aw.ds }.

- Visual Studio Code
    - Using the given link, download Visual Code Studio and install it.{https://code.visualstudio.com/download}

The Terminal

As Node.js is a runtime environment over javascript, so all the files in Node.js will be with an extension of javascript, which is .js.
Let's set up our folder to proceed further into the course.

Open the terminal and follow the commands.

- {cd..} - It goes one step above in the parent directory or parent folder {cd - change directory}.

- {LS} - It shows the list of folders and files inside my current folder.
- {pwd} - It shows the current path.

- {cls or ctrl + L or cmd K or clear cmd [According to your OS]} - To clear the terminal.

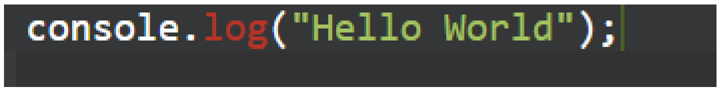
- {mkdir} - To create a folder.
- {cd folder_name} - To go into a specific folder.

Open this folder in VS code & Create a new file there in the created folder with an extension of js

Hello Node.js

Let's run our first JS file on our system using Node.

Go to the terminal on VS code editor and run the first JS file there, which is stored in file **index.js** that is shown below.



```
console.log("Hello World");
```

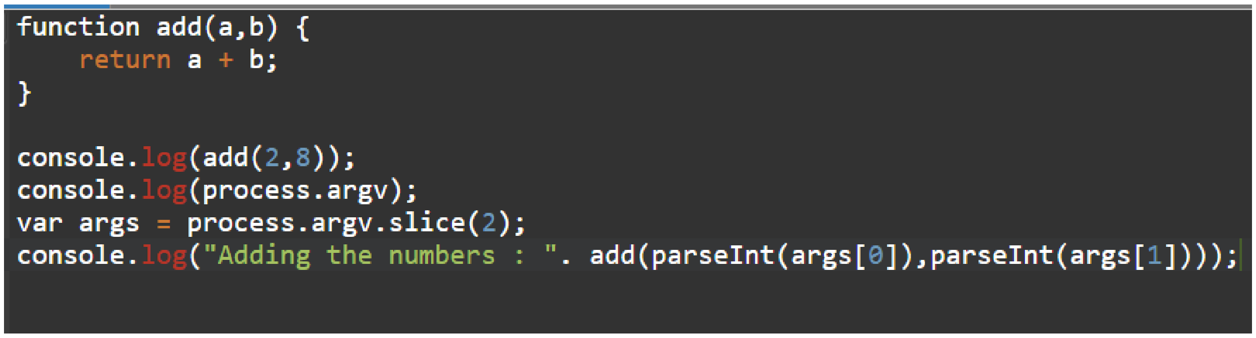
You can run all your terminal commands on the VSCode terminal. To execute your file, simply write "node index.js".

If you want to pass any argument using the command line, you can access it using a global object using "**process"** [It looks over whatever is running on Node.js with all the arguments that are passed to that file. It also looks upon what is going on in Node.js].

If you have passed two arguments in the function , you will get four arguments on display which are -

- First Part - Path from Node.js is being run
- Second Part - Path of the running file which is run by Node.js.
- Two arguments are by default converted into a string.

To access the arguments that we have passed we need to slice that into two parts. We then convert those strings into an integer for further usage, For Example - {Here in add function}.



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

console.log(add(2,8));
console.log(process.argv);
var args = process.argv.slice(2);
console.log("Adding the numbers : ". add(parseInt(args[0]),parseInt(args[1])));
```

***<u>EXTRA</u>*: You can try a javascript game or calculator as your mini assignment.**

Modules

Modules is a library or set of reusable functions which make our code easy to read and less cluttered.

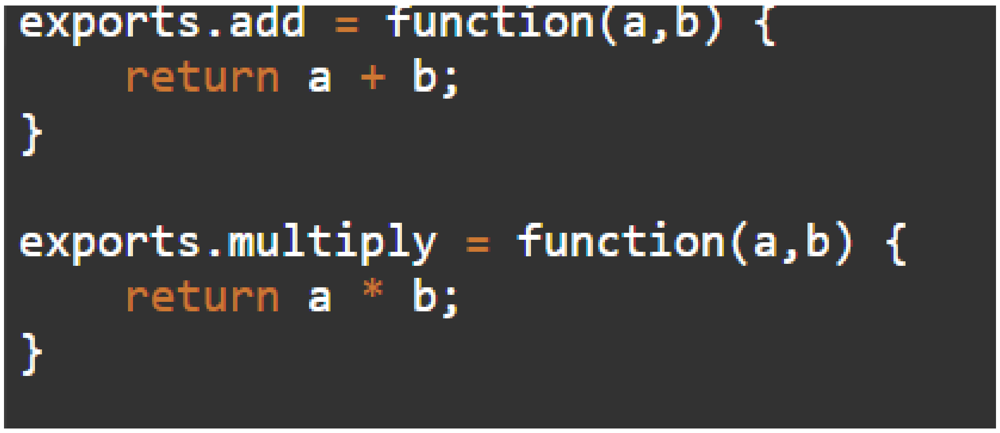
- Make a new folder {modules_node_js} .
- Inside that create two new files {index.js} & {operations.js}.
- Create {add} function .
- To make {operations.js} available outside the file we use {module.exports}.

The whole file {operations.js} is a module and {module. exports} is an object. In that object, we are giving function_name {add} as a key. This key corresponds to the function created. We can add multiple functions as keys that have values as functions.

- Create one more function {multiplication} and export that as well {To make it available in other files too}.
- To import those functions- in {index.js}, we need to store that module into some variable.

★ ***NOTE - [./ means it goes inside the file at the same level below that file]***

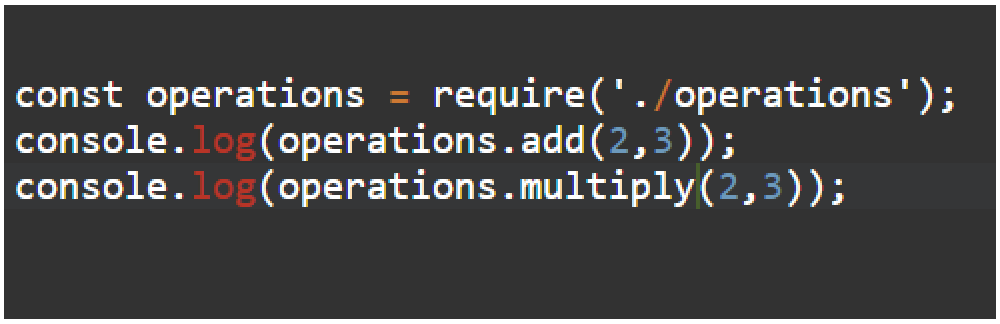
{operations.js}



```
exports.add = function(a,b) {
    return a + b;
}

exports.multiply = function(a,b) {
    return a * b;
}
```

{index.js}



```
const operations = require('./operations');
console.log(operations.add(2,3));
console.log(operations.multiply(2,3));
```

- ***NOTE - If you don't export any function you won't be able to access it and if you try to do so you will get an error that {function_name} is not a function.***
- ***NOTE - You can also remove modules from {modules.export {function_name}} as it is by default available to all files in the modules folder.***

Summarizing It -

Let's summarize what we have done in this module.

- We set up all the tools including Node. js the chrome, Vs code.
- Learned a little about Node.js.
- Learned about modules and it's usage along with it's implementation.

NPM [Node package Manager] always gets installed automatically when we install Node. Whenever we want to use a library or package, we need Node to install it. To check if Node is installed in the system, run the command {npm -v} on the terminal. This will also display the version number.

***<u>EXTRA</u>*: https://www.npmjs.com/**
This website holds lists and statistics of all the packages which are available