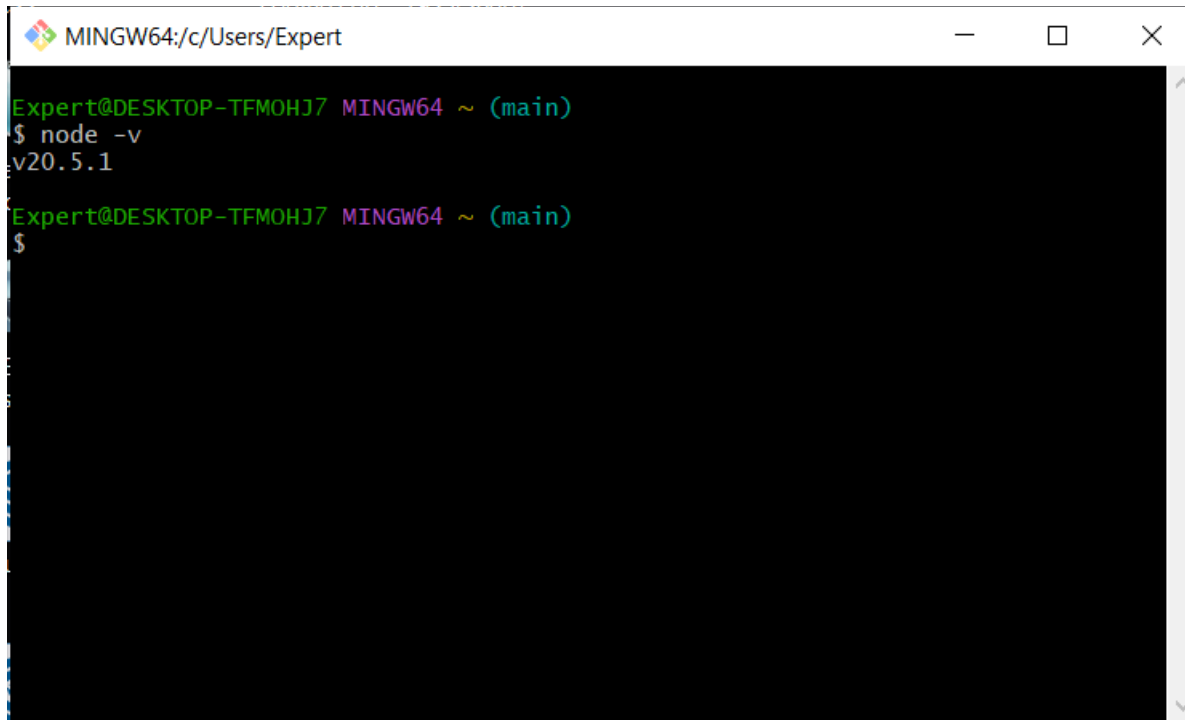


# NODE.JS

- Not a language, library or framework.
- JavaScript Run-time Environment.
- It is used for server side programming.
- VSCode is made using node.js.

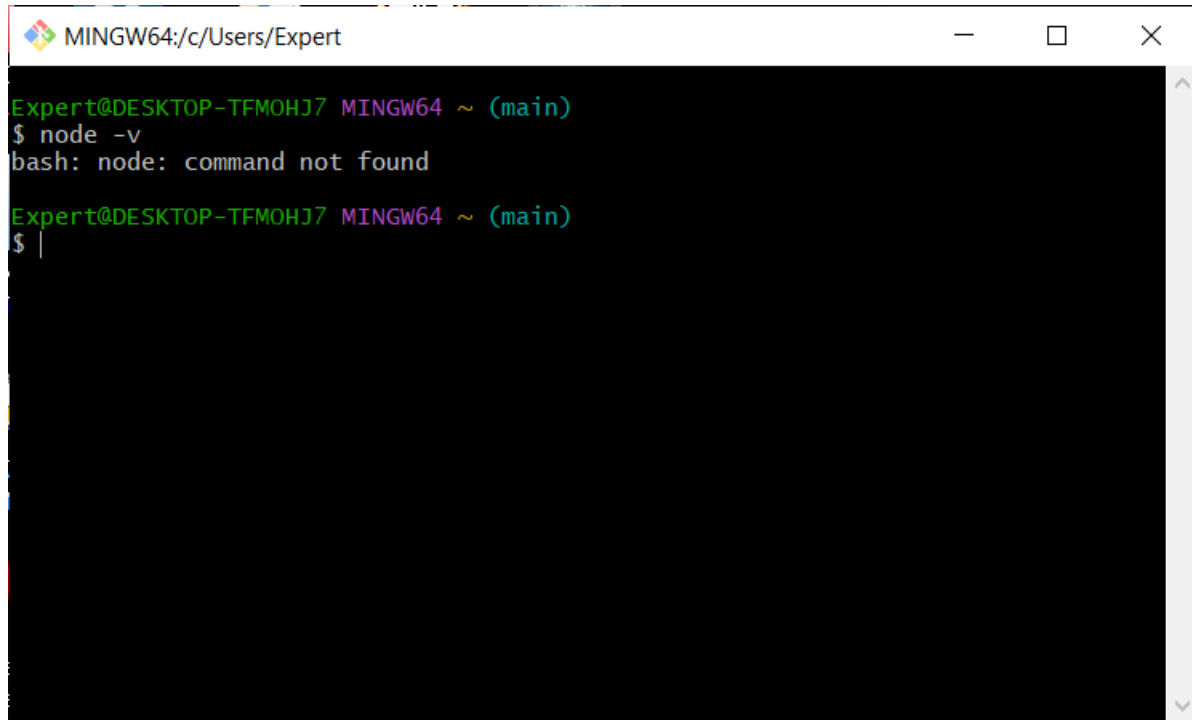
## HOW TO CHECK NODE IN OUR SYSTEM?

- Go in Terminal (MacOS) or GitBash (Windows OS).
- Type command `node -v`.
- If it exists, version will be output:



```
MINGW64:/c/Users/Expert
Expert@DESKTOP-TFMOHJ7 MINGW64 ~ (main)
$ node -v
v20.5.1
Expert@DESKTOP-TFMOHJ7 MINGW64 ~ (main)
$
```

*If node doesn't exist, we need to install it.*

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/Expert'. The prompt is 'Expert@DESKTOP-TFM0HJ7 MINGW64 ~ (main)'. The user enters '\$ node -v' and the terminal outputs 'bash: node: command not found'. The prompt returns to 'Expert@DESKTOP-TFM0HJ7 MINGW64 ~ (main)' with a cursor on a new line.

```
Expert@DESKTOP-TFM0HJ7 MINGW64 ~ (main)
$ node -v
bash: node: command not found
Expert@DESKTOP-TFM0HJ7 MINGW64 ~ (main)
$ |
```

- Search [nodejs.org](https://nodejs.org/en) or follow the link to install <https://nodejs.org/en>

## USING NODE

1. Type “node” in terminal.

```
Expert@DESKTOP-TFMOHJ7 MINGW64 ~/Desktop/Aafreen/JS (main)
$ node
Welcome to Node.js v20.5.1.
Type ".help" for more information.
>
```

We are entered in node where we can't use any terminal commands.

2. Exit node using “.exit” or “press ctrl + C two times.”

## RUNNING JS FILES IN NODE

1. To run JS files in Node, be in the directory where our files are already present.

We have a file hello.js

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

2. Type “node filename.js”.

```
Expert@DESKTOP-TFMOHJ7 MINGW64 ~/Desktop/Aafreen/JS (main)  
$ node hello.js  
Hello
```

3. The output of file will be printed.

## PROCESS IN NODE

**process** : This object provides information about, and control over, the current Node.js process.

**process.argv** : returns an array containing the command line argument passed when the Node.js process was launched.

Node.js	Process.js
<pre>Expert@DESKTOP-TFM0HJ7 MINGW64 ~/Desktop/Aafreen/JS (main) \$ node process.js Aafreen Saniya Najma Vaishnavi [   'C:\\Program Files\\nodejs\\node.exe',   'C:\\Users\\Expert\\Desktop\\Aafreen\\JS\\process.js',   'Aafreen',   'Saniya',   'Najma',   'Vaishnavi' ] Hello Aafreen Hello Saniya Hello Najma Hello Vaishnavi</pre>	<pre>Welcome JS process.js U X JS process.js &gt; ... 1 console.log(process.argv); 2 let args= process.argv; 3 v for (let i=2;i&lt;args.length;i++) 4 { 5     console.log("Hello ",args[i]); 6 }</pre>

## module.exports(Export in files)

`require()` – a built-in function to include external modules that exist in separate files.

`module.exports` – a special object.

Consider following example of 2 files – `script.js` and `math.js`

script.js	math.js	node
<pre>const someValue=require("./math"); //someValue can be any value console.log(someValue);</pre>	<pre>module.exports=143;</pre>	<pre>Expert@DESKTOP-TFMOHJ7 MINGW64 ~/Desktop/Aafreen/JS (main) \$ node script.js 143</pre>
<pre>const summ=require("./math");   console.log(summ);</pre>	<pre>module.exports.sum=(a,b)=&gt;a+b;</pre>	<pre>Expert@DESKTOP-TFMOHJ7 MINGW64 ~/Desktop/Aafreen/JS (main) \$ node script.js { sum: [Function (anonymous)] }</pre>
<pre>const summ=require("./math");   console.log(summ);</pre>	<pre>const sum=(a,b)=&gt;a+b; const mul=(a,b)=&gt;a*b; const g=9.8; const PI=3.14;  module.exports={   sum: sum,   mul: mul,   g: g,   PI: PI, };</pre>	<pre>Expert@DESKTOP-TFMOHJ7 MINGW64 ~/Desktop/Aafreen/JS (main) \$ node script.js { sum: [Function: sum], mul: [Function: mul], g: 9.8, PI: 3.14 }</pre>

```
const summ=require("./math");
|
console.log(summ);
```

```
const sum=(a,b)=>a+b;
const mul=(a,b)=>a*b;
const g=9.8;
const PI=3.14;
```

```
Expert@DESKTOP-TFMOHJ7 MINGW64 ~/Desktop/Aafreen/JS (main)
$ node script.js
{ sum: [Function: sum], mul: [Function: mul], g: 9.8, PI: 3.14 }
```

```
let obj={
  sum: sum,
  mul: mul,
  g: g,
  PI: PI,
};
```

```
module.exports=obj;
```

```
const summ=require("./math");
console.log(summ.sum(2,2));
|
```

```
const sum=(a,b)=>a+b;
const mul=(a,b)=>a*b;
const g=9.8;
const PI=3.14;
```

```
Expert@DESKTOP-TFMOHJ7 MINGW64 ~/Desktop/Aafreen/JS (main)
$ node script.js
4
```

```
let obj={
  sum: sum,
  mul: mul,
  g: g,
  PI: PI,
};
```

```
module.exports=obj;
```

## module.exports(Export in Directories)

e.g.Fruits

1. Create a folder fruits.

```
> fruits
JS hello.js      U
JS math.js      U
NODE.pdf        U
JS process.js   U
JS script.js    U
```



▼ fruits	●
JS apple.js	U
JS index.js	U
JS orange.js	U
JS watermelon.js	U

2. Add files.

3. Add data in files apple.js , orange.js, watermelon.js ; index.js is a special file to export data.

```
module.exports={  
  name: "apple",  
  color: "red",  
};
```

```
module.exports={  
  name: "orange",  
  color: "orange",  
};
```

```
module.exports={  
  name: "watermelon",  
  color: "red-in",  
};
```

4. Acquire all exports in index.js and export to script.js

```
const apple= require("./apple");  
const orange= require("./orange");  
const watermelon= require("./watermelon");  
  
let fruits = [apple,orange,watermelon];  
  
module.exports=fruits;
```

5. Script.js

```
const info=require("./fruits");
```

```
console.log(info);
```

## 6.Node

```
Expert@DESKTOP-TFM0HJ7 MINGW64 ~/Desktop/Aafreen/JS (main)
```

```
$ node script.js
```

```
[
```

```
  { name: 'apple', color: 'red' },
```

```
  { name: 'orange', color: 'orange' },
```

```
  { name: 'watermelon', color: 'red-in' } 
```

```
]
```

## NPM(Node Package Manager)

- npm is the standard package manager for Node.js.
- Library of packages.
- npm is command line tool.
- Packages are the codes of other peoples to use.
- Express.js,react.js are also packages which are used by many users all over the world.
- It is preinstalled with node. Check it in terminal.

```
Expert@DESKTOP-TFM0HJ7 MINGW64 ~/Desktop/Aafreen/JS (main)
$ npm
npm <command>

Usage:

npm install          install all the dependencies in your project
npm install <foo>    add the <foo> dependency to your project
npm test            run this project's tests
npm run <foo>        run the script named <foo>
npm <command> -h     quick help on <command>
npm -l              display usage info for all commands
npm help <term>      search for help on <term> (in a browser)
npm help npm         more involved overview (in a browser)

All commands:

access, adduser, audit, bugs, cache, ci, completion,
config, dedupe, deprecate, diff, dist-tag, docs, doctor,
edit, exec, explain, explore, find-dupes, fund, get, help,
help-search, hook, init, install, install-ci-test,
install-test, link, ll, login, logout, ls, org, outdated,
owner, pack, ping, pkg, prefix, profile, prune, publish,
query, rebuild, repo, restart, root, run-script, search,
set, shrinkwrap, star, stars, start, stop, team, test,
token, uninstall, unpublish, unstar, update, version, view,
whoami

Specify configs in the ini-formatted file:
  C:\Users\Expert\.npmrc
or on the command line via: npm <command> --key=value

More configuration info: npm help config
Configuration fields: npm help 7 config

npm@9.8.0 C:\Program Files\nodejs\node_modules\npm
```

## Installing packages

- Use the following command to install package

`npm install <-Package name->`

```
Expert@DESKTOP-TFMOHJ7 MINGW64 ~/Desktop/Aafreen/npm/figlet (main)  
$ npm install figlet  
added 1 package in 12s
```

*Install the package where we will run the code of other files,*