**Node.js**

**Backend or server**

* A server is a computer or system that provides resources, data, services, or programs to other computers, known as clients, over a network. It handles requests from clients, processes them, and sends back the appropriate responses.

**Client or frontend**

* A client is a computer or software that requests services, resources, or data from a server over a network. It interacts with the server by sending requests and receiving responses, typically through a web browser, application, or other interface.

**CLI vs GUI**

CLI and GUI are two different ways of interacting with a computer or a software.

CLI: A text-based interface where users type commands to perform tasks.

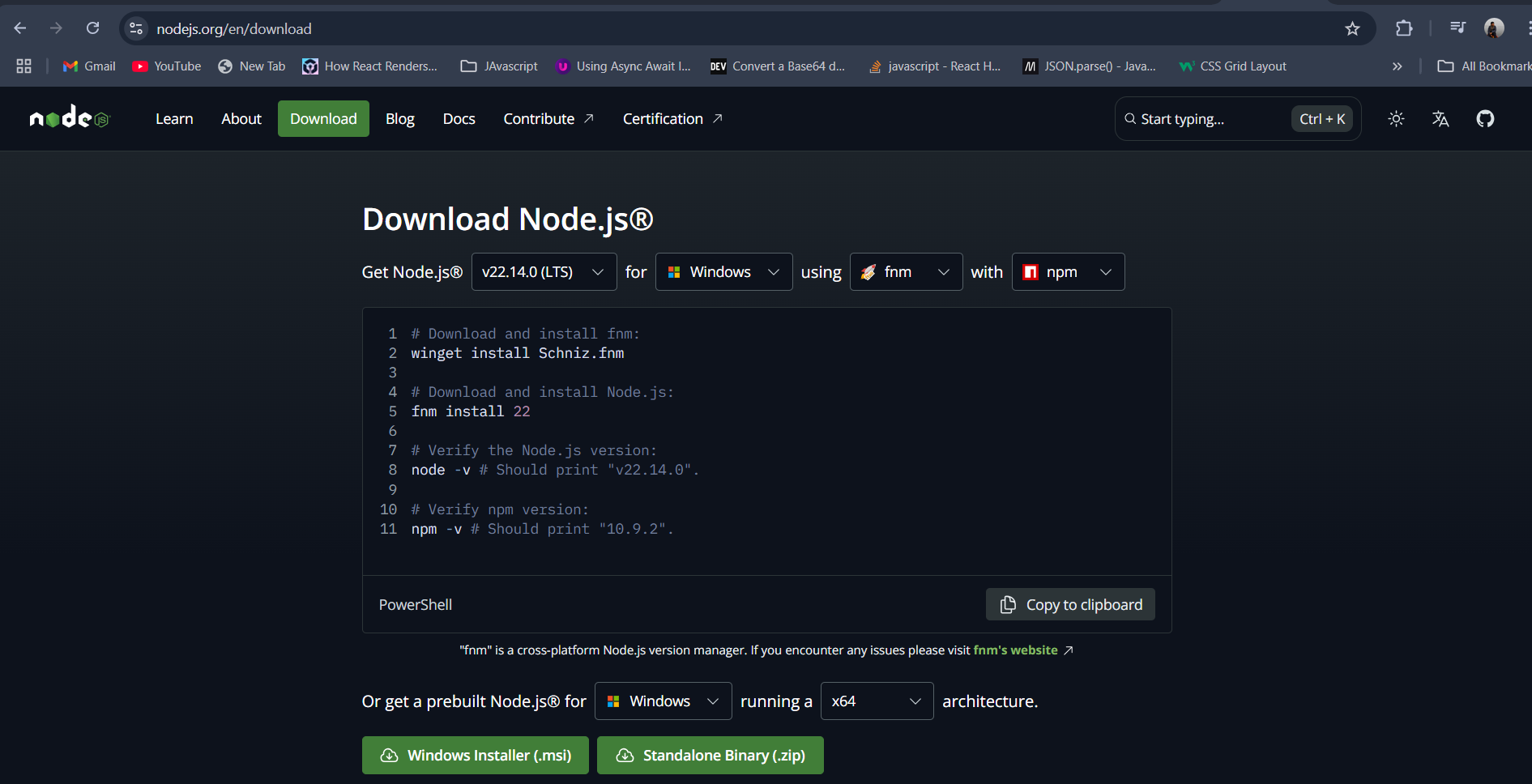
GUI: A visual interface where users interact with the system through graphical elements like windows, icons, buttons, and menus.

**Node.js**

Node.js a C++ application (CLI) which can understand and run JavaScript code outside of the browser. It provides a runtime environment built on Chrome's V8 JavaScript engine, enabling developers to execute JavaScript on the server side.

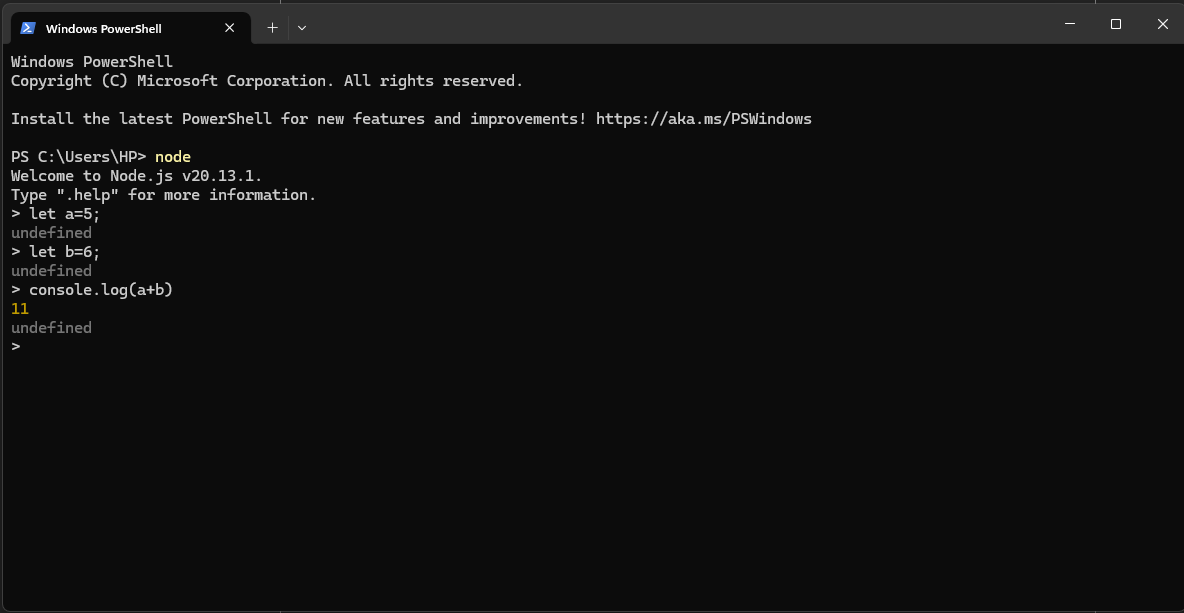
Installing Node.js

Go to nodejs website and download the file



Now install the file that you downloaded.

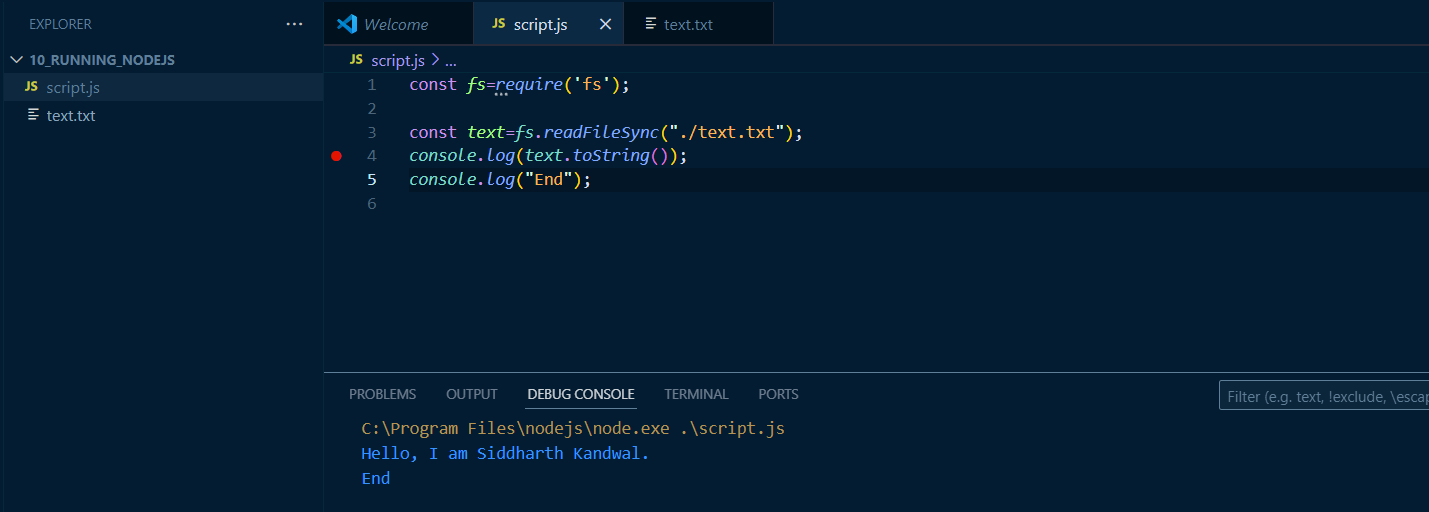
Now open the terminal and write node and you will get the node REPL(like console) where you can write the whole Javascript code.



If you don’t want to write the code like this and you have written the code In a file and you want to run that, then just write node {file\_name}.

**Reading a file from JS**

If you want to read a file from javascript, then to do that:



Important Note: If you want to add a path of a file in JS then add the \ mark after each \ mark.

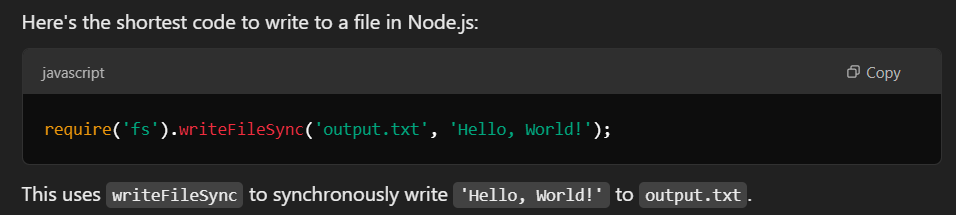
File path: "C:\Users\HP\Desktop\text.txt"

Path that you will give: "C:\\Users\\HP\\Desktop\\text.txt"

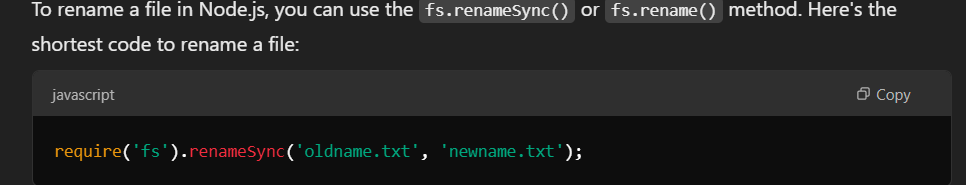
As there is window object in Javascript, similarly there is global object in nodejs as well. You can use globalthis object in javascript as well as nodejs to access window and global object respectively.

**Why Node.js???**

* By Node.js you have the power to interact with file system, allowing you to read, write, delete and manipulate files and directories which cannot be done using normal javascript.

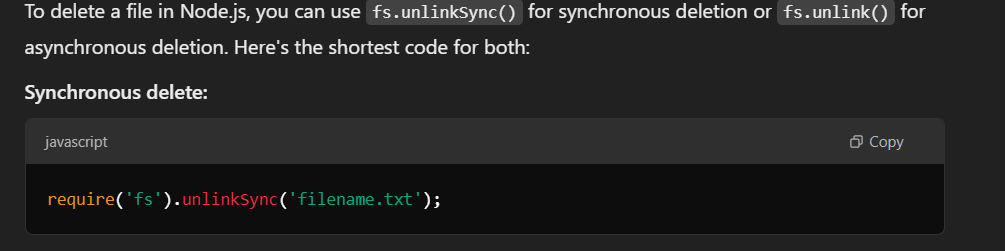
**Write in a file:** 

**Rename file:**



Give full path in both the places otherwise it will search only in current folder.

**Delete file:**



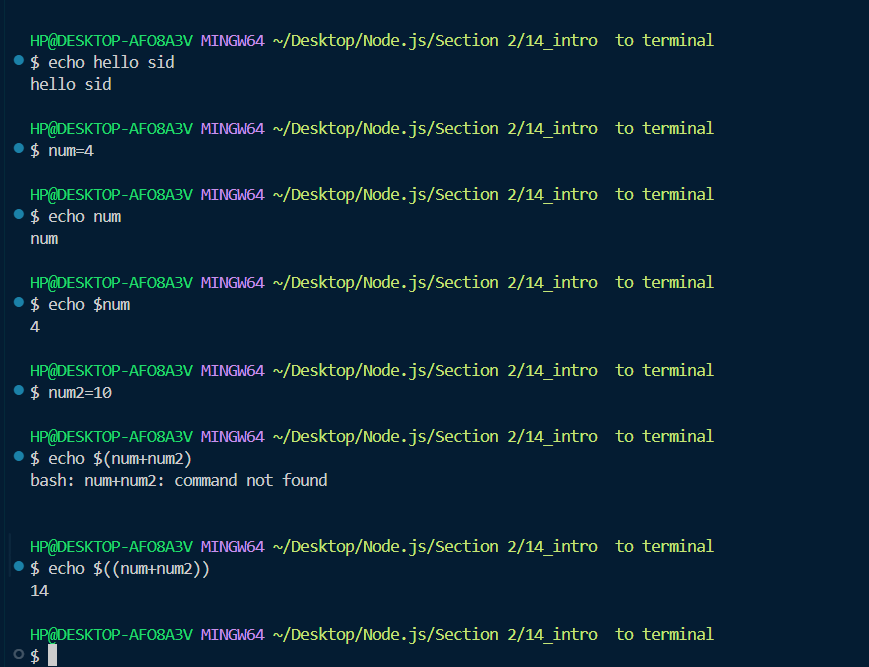
It will permanently delete the file.

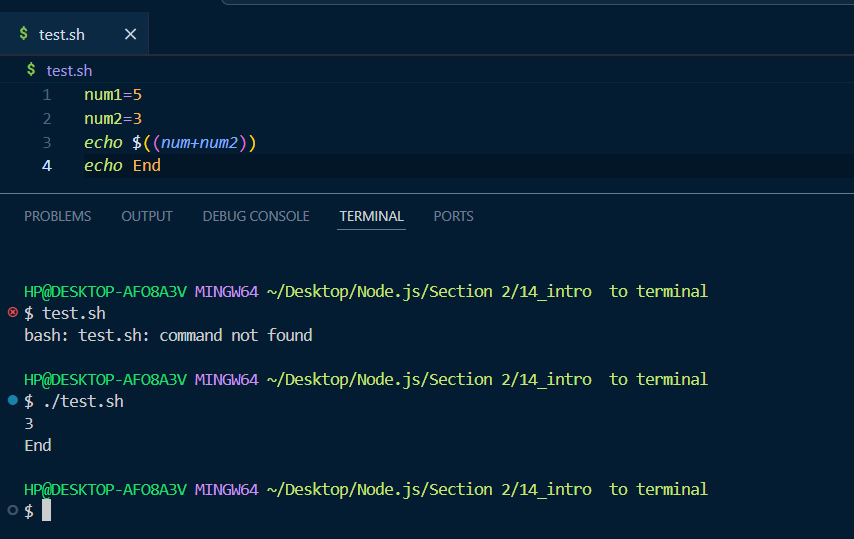
* Nodejs has network capabilities, it can start a server which cannot be done by javascript. Through Javascript, we can send req to server and read response from it.
* Process management can be done using Nodejs while not with Javascript.
* Nodejs can interact with OS to perform tasks like reading env variables, interacting with system processes, scheduling tasks etc. which cannot be done using JS.

**Terminal and commands:**

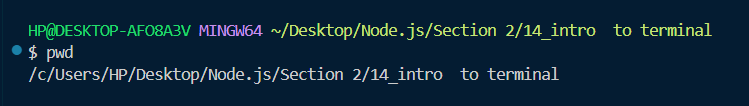
Bash is a scripting language that have loops variables and all things in it. Extension of bash file is .sh

* **Echo**: echo command is like console.log. You can print anything: any text or variable.

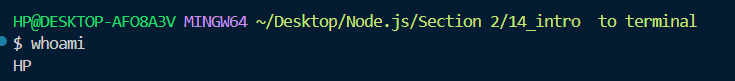




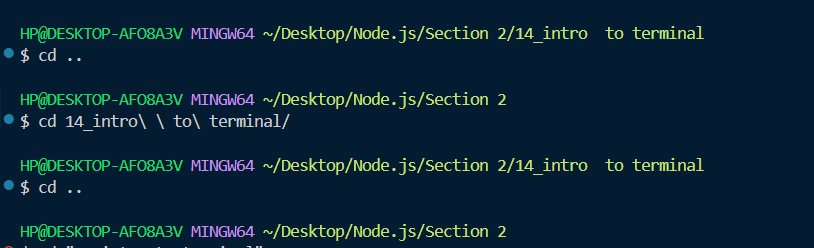
* **Pwd**: It will print the path of the current directory.



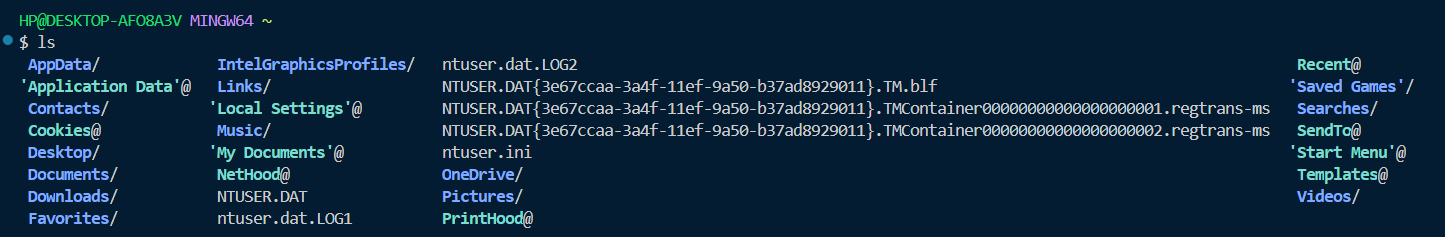
* **Whoami**: It will print the name of current user.



* **Cd**: This command is used to change the directory.



* **Ls**: This command is used to list all the content in a directory.

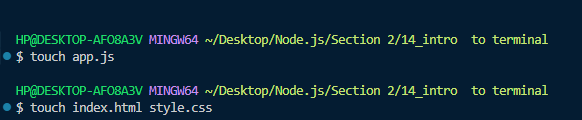


All the names with @ symbol are the symbolic links and they are not visible in the folder, they can be visible by ls command in terminal.

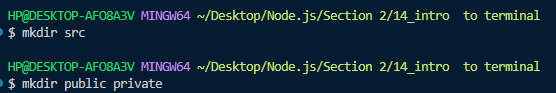
**Ls -a**: this command will display all the files in current directory.

**Ls -la**: this command will display all the file permissions.

* Touch: touch command is used to create a new file.

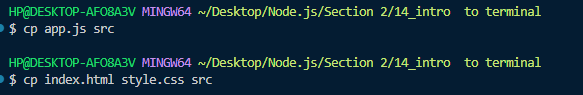


* Mkdir: mkdir command Is used to create a directory.



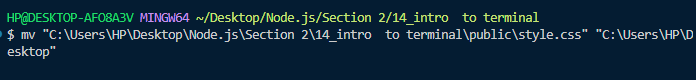
Mkdir public private: it will create two folders i.e. public and private.

* Cp: cp command is used to copy a file to a different location. (only file)



Cp {file\_path} {destination\_path}

* Mv: mv command is used to move a file to a different location.





You can give the full path as well or if your file and destination folder is in same directory as you are then you can just name the file and folder as well.

* Mv: mv command is also used to rename a file. You just have to give the original file name and then the name that you want to give.





This will move the file(first.html) to the path and rename it to indexx.html

* Rm: rm command is used to delete the file(permanently delete).

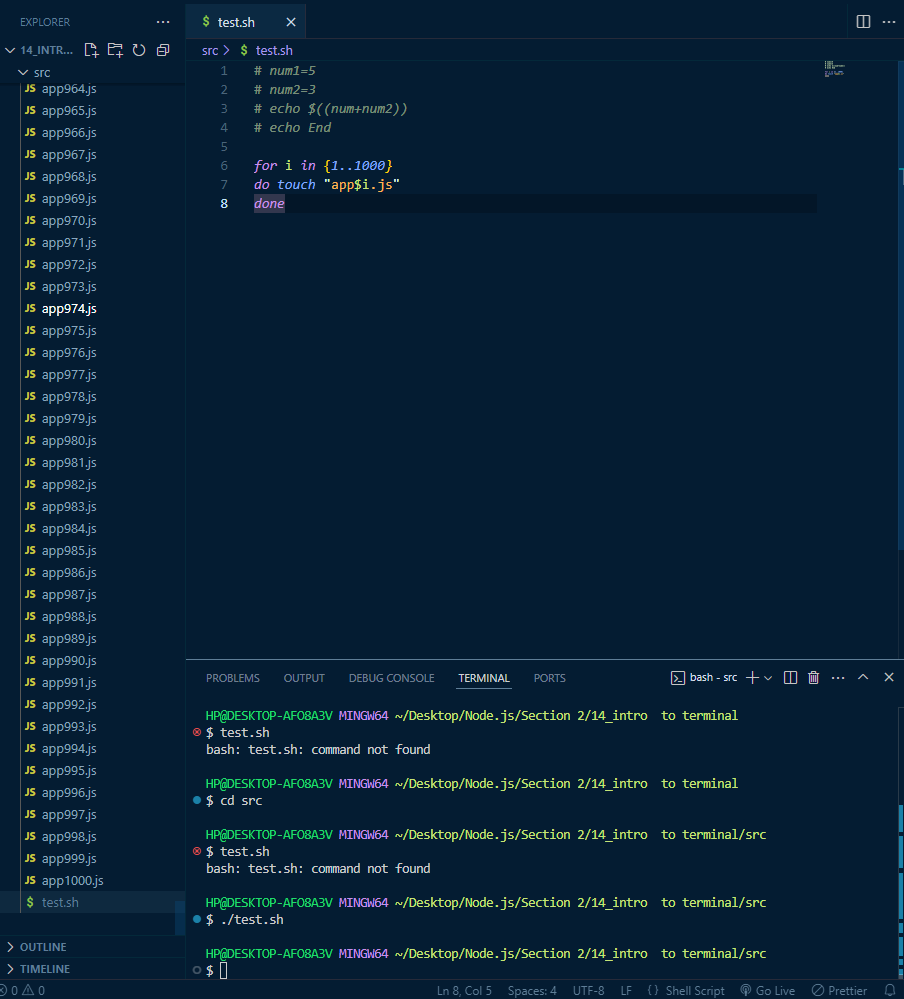


* Rmdir: rmdir command Is used to delete an empty directory.
* Rm -r: rm -r command is used to delete any directory whether it have nested directory or files in it.

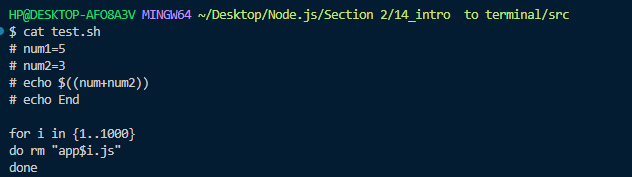


Why bash is useful??

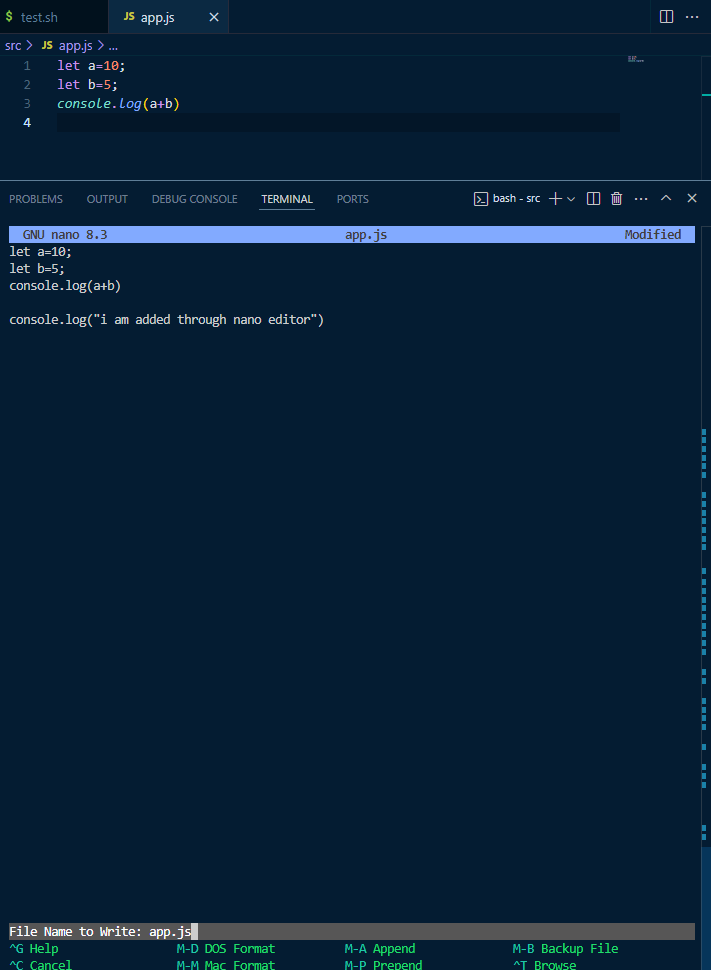
Suppose you want to create 1000 files, then if you manually try to do that it will take a lot of time. So you can use bash commands to generate it.



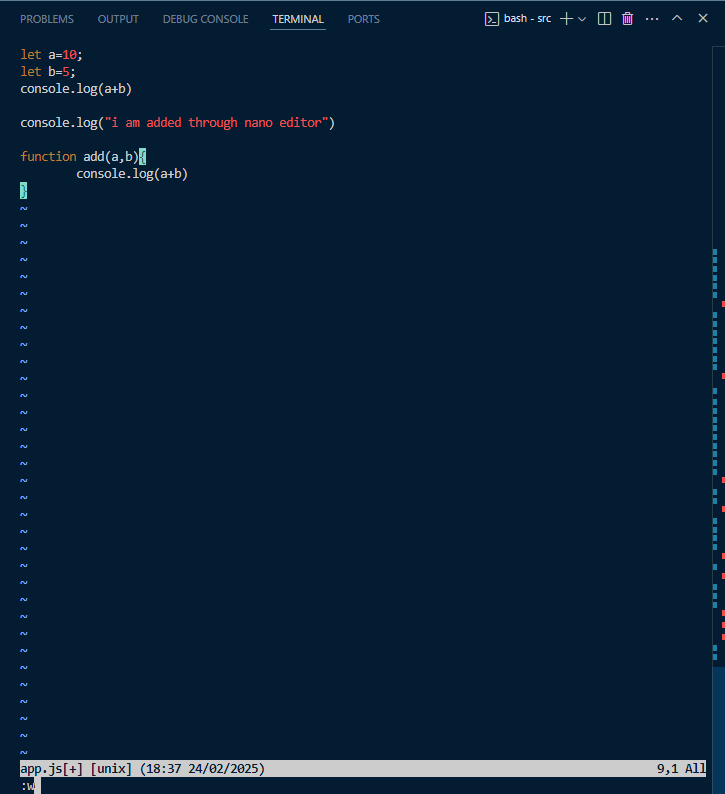
* Cat: cat command is used to display the content of a file.



* Nano: nano command is used to edit a file. It opens the file in nano editor.



* Vim/vi: This command is also used to edit a file.



When we are typing then it will be in insert mode then press ESC and then:

:w is used to save the changes to the file.

:q is used to exit the vim editor.

:q! is used to exit the vim editor without saving the changes.

:wq is used to save and quit simultaneously at the same time.

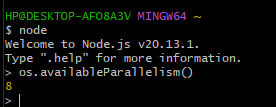
**Prompt in Terminal**

**CPU/ Processor**

A CPU (Central Processing Unit) core is the processor within your computer that executes instructions and processes data.

**Core**

Core is a processing unit that can process any task independently. Each core can be thought of as a mini-processor within the CPU.

To get the number of logical cors of the system through nodejs: 

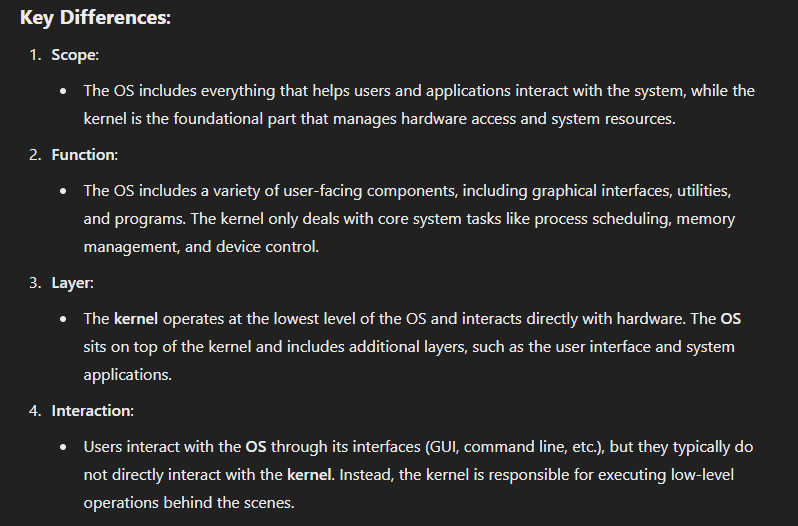
**Operating system**

The **Operating System** is the software that acts as an intermediary between computer hardware and the user. OS is a software that manage hardware resources for the application. It decides which application uses how much hardware, how much RAM, how much Disk, how much CPU time will that app use.

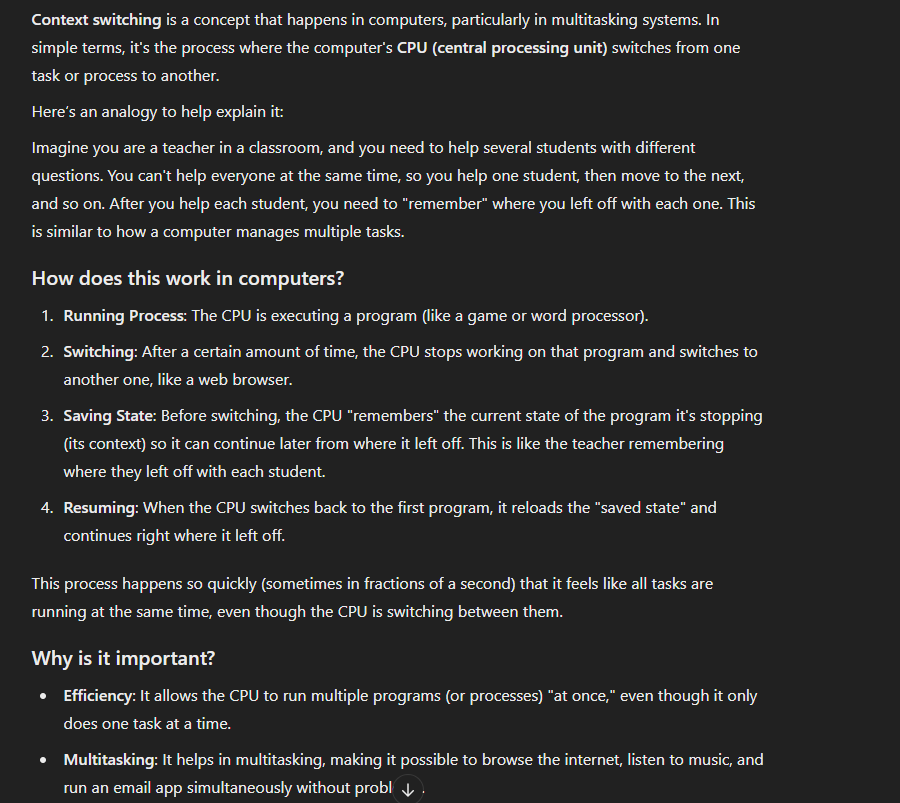
**Kernel**

Kernel is a part of OS that helps managing hardware resources.

**Key Difference between OS and Kernel**

****

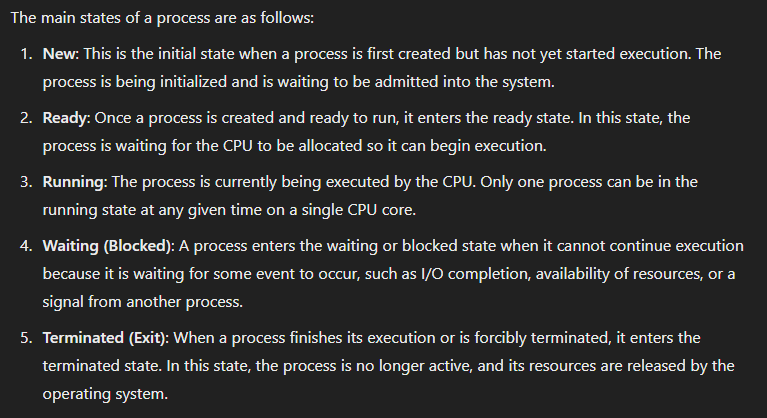
**Context Switching**

****

**Process:**

Process is a unit of execution. A program that processor executes or can do computation on.

Process exist in different states. It’s not necessary that process has any CLI or user interface, it may just be a program that is executing.



There is a global variable in nodejs that you can access by which you can get the info about the nodejs process. You can get the process id(pid), parent process id(ppid).

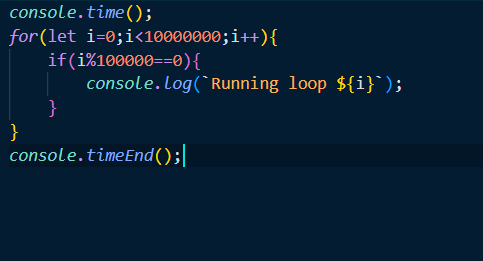
Spawning a process means when a parent process starts or triggers a child process.

Spawning a process takes more time while spawning a thread takes less time. This is because process has its own memory space it takes some memory from RAM and then starts while thread shares the memory of process.

If you have 2 cores and you have two threads of a process, then they will run parallelly one process will run on one core and another process on another core.

But if you have only 1 core and two threads of a process, so is it worth it to create two threads of a process then? Can’t we directly have two processes as there is only one core?? Yes it is worth it to create two threads, two threads will run concurrently (or in context switching mode) in a core and we can get the output of both the threads simultaneously. Suppose one thread is of watching a video and another is of downloading a video, so it will download video a bit and buffer video a bit so that we can watch it. So we can say multithreading is worth it.

**How to find the time taken by a particular code**

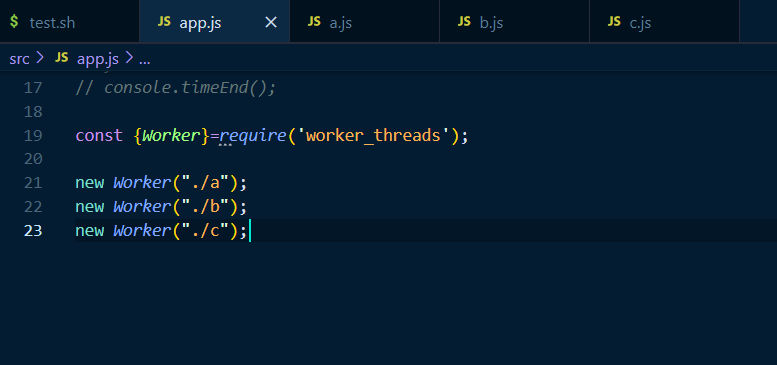
****

Console.timeEnd() will print the time taken for the code run from console.time() to console.timeEnd();

**All work in a single file:**



**Dividing the work into different threads:**

****

These three lines created 3 new threads and all 3 threads will run parallely.

**Is node js single threaded or multi threaded?**

Nodejs is multithreaded. Previously nodejs was single threaded, it uses multithreading behind the scenes but as a developer we can’t use multithreading previously. But in 2019 worker\_thread module was introduced in nodejs so from that time nodejs is truly a multithreading language.

**Can a process exist without a thread?**

It’s not possible that a process exist without a thread. Because whenever a process starts, it already have one thread by default called as main thread. So main thread is always there with the process. So, when we say that a process is getting into the core that means main thread is getting into the core to execute necessary functionality.

**Environmental variables**

Env variables are the string based key value pairs.

If we want to create a env variable then we have to write ***export num=58*** in terminal. It will add it as an env variable in bash terminal, so whenever a child process of bash terminal starts this value will automatically get passed into it’s child process env variable value.

To access env variable we have to do *echo ${variable\_name}*.

**Files and folders**

Folders is a container that consists of some files or some folders while files is a container that consists of data.

**Symbolic links**

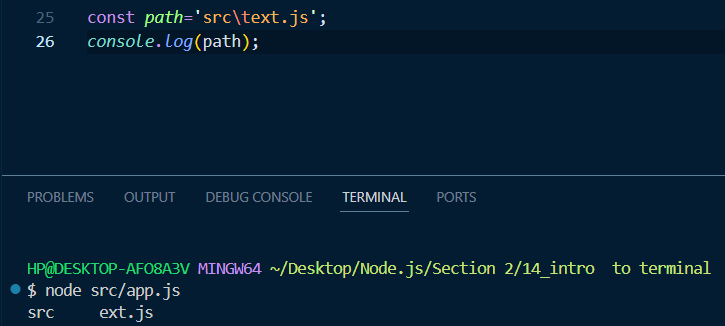
A **symbolic link** (or **symlink**) is a special type of file in a filesystem that serves as a reference or pointer to another file or directory. It essentially acts like a shortcut, allowing you to access a file or directory from a different location in the filesystem without duplicating the actual data.

**Paths are of two types**

Absolute path and relative path. Absolute path is a path from root directory to the current file (in windows root directory is C drive). While relative path specifies a file or directory's location in relation to the current working directory. It doesn’t start from the root, but rather from where you currently are in the filesystem.

For any path in nodejs or JS we generally use \\ because using single \ will create issues as \ is a special character in programming language.

Without using \\:



With using \\:



Cd . -> Current folder

Cd .. -> Parent folder

cd ../ -> Parent folder.

Cd ~/->Home folder.

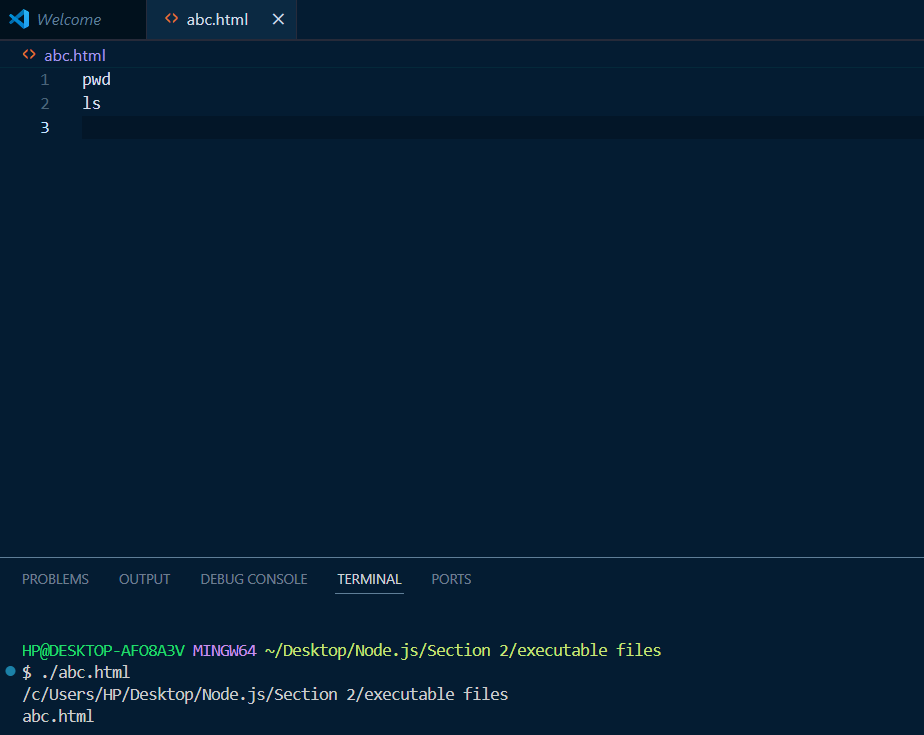
Cd ~->Home folder.

**Executable files**

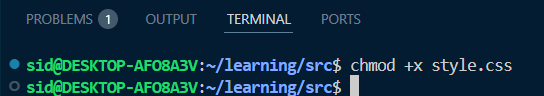
Executable files are of two types: Binary and script exec files.

Script exec files are those text files which we can run by naming it in terminal. Each line is treated as a command and they run when we name it in terminal. Script files are plain text files that contain a series of commands, written in a programming or scripting language, that are interpreted and executed by an interpreter or a scripting engine. These commands are human-readable and typically include instructions for automation, file manipulation, or system control.

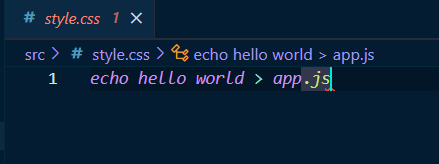
Examples:



It’s necessary to write ./ before file name in terminal otherwise it will considered as a command and will not run and give error.



This command gives executable file permission to a file.



When this file runs, it will create a app.js file with content as “hello world”.

**Binary files**: Binary files contain data in a format that is not human-readable. These files are often used to store compiled code or data that is intended to be processed by a machine, rather than directly interpreted by humans. Some common types of binary files include:

* **Executable files**: .exe (Windows), .bin (Linux), .app (Mac)
* **Object files**: .obj, .o
* **Library files**: .dll (Windows), .so (Linux)
* **Image, audio, and video files**: .jpg, .mp3, .mp4s

**Important methods and properties of Process Object**

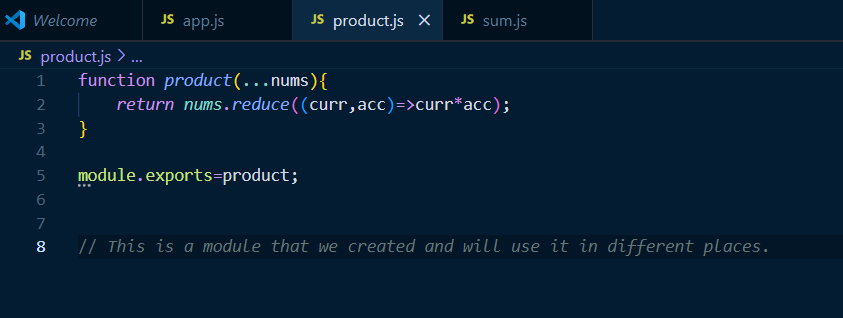
* Process.argv: This print an object, one is the path of nodejs and second is the path of file in which that command is run. But if you directly run process.argv in terminal then it will print the path of nodejs only.
* Process.pid: it will give the process id.
* Process.ppid: it will give the process id of the parent.
* Process.cwd(): this will give the current working directory.
* Process.chdir(“./tmp”): this will help in changing the current working directory.
* Process.memoryUsage(): give memory usage.
* Process.uptime(): it tells how long our process is started.
* Process.exit(0): it stop the process and tells no error is encountered.
* Process.exit(1): it stop the process and tells an error is encountered.
* Process.kill(): it is used to stop the process.

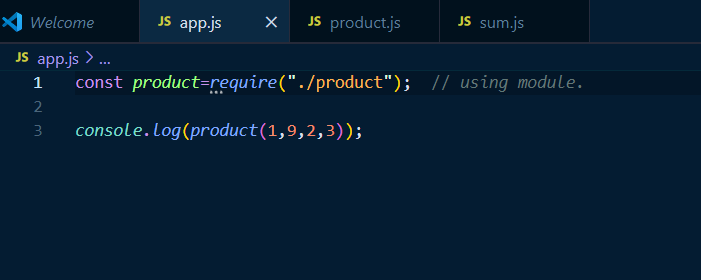
**Module system**

One of the method of creating a global variable in nodejs is to use **global.num=45**. It will make a global variable num.

When the code increases with time, then we split the functionality of the code into different files, and we just use the functionality accordingly. They allow developers to separate the code into reusable chunks, which helps in managing code more efficiently.

How to make any module and how to use them????





This require is a function, as it is a function then it will have some arguments and also it returns some value.

So let’s see its argument: Argument of require function is a string that should be a path of a file. If it does not find the path of that file, then it will give an error stating cannot find module…… But if it finds the file given in the path as argument then it will execute the whole code of that file and will decide the return value from the file. The return value of require function is the **module.exports** present in the file given as path of the require function.