

Node

Node.js

JavaScript Runtime Environment

It is used for **server** side programming.

***Node.js is not a language, library or framework.**

If you want to run a js file using node make sure you are in the current directory in which the file exists and then write node filename.

```
shradhakhapra@Shradhas-MacBook-Air ~ % cd WebClassroom/Backend
shradhakhapra@Shradhas-MacBook-Air Backend % ls
shradhakhapra@Shradhas-MacBook-Air Backend % touch script.js
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
hello, 0
hello, 1
hello, 2
hello, 3
hello, 4
shradhakhapra@Shradhas-MacBook-Air Backend %
```

```
JS script.js
1 let n = 5;
2
3 for (let i = 0; i < n; i++) {
4   console.log("hello, ", i);
5 }
6
```

Process

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

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

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
[
  '/usr/local/bin/node',
  '/Users/shradhakhapra/WebClassroom/Backend/script.js'
]
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js hello bye
[
  '/usr/local/bin/node',
  '/Users/shradhakhapra/WebClassroom/Backend/script.js',
  'hello',
  'bye'
]
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js hello bye
shraddha apnacollege 52
[
  '/usr/local/bin/node',
  '/Users/shradhakhapra/WebClassroom/Backend/script.js',
  'hello',
  'bye',
  'shraddha',
  'apnacollege',
  '52'
]
shradhakhapra@Shradhas-MacBook-Air Backend %
```

```
JS script.js > ...
1 // let n = 5;
2
3 // for (let i = 0; i < n; i++) {
4 //   console.log("hello, ", i);
5 // }
6
7 // console.log("bye!");
8
9 let args = process.argv;
10
11 for (let i = 2; i < args.length; i++) {
12   console.log("hello to ", args[i]);
13 }
14
```

Whatever we write after writing the name of the file becomes the argument for argv argument.

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js shradha rajat rahul neha manisha
hello to shradha
hello to rajat
hello to rahul
hello to neha
hello to manisha
shradhakhapra@Shradhas-MacBook-Air Backend %
```

```
JS script.js > ...
1 // let n = 5;
2
3 // for (let i = 0; i < n; i++) {
4 //   console.log("hello, ", i);
5 // }
6
7 // console.log("bye!");
8
9 let args = process.argv;
10
11 for (let i = 2; i < args.length; i++) {
12   console.log("hello to ", args[i]);
13 }
14
```

APNA COLLEGE

module.exports

requiring files

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

module.exports a special object

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
123
shradhakhapra@Shradhas-MacBook-Air Backend %
```

```
JS math.js > [unknown]
1 const sum = (a, b) => a + b;
2 const mul = (a, b) => a * b;
3 const g = 9.8;
4 const PI = 3.14;
5
6 module.exports = 123;
7
```

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
123
shradhakhapra@Shradhas-MacBook-Air Backend %
```

```
JS script.js x JS math.js
JS script.js > ...
1 const someValue = require("./math");
2
3 console.log(someValue);
4
```

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
123
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
hello!
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
{}
shradhakhapra@Shradhas-MacBook-Air Backend %
```

```
JS script.js JS math.js x
JS math.js > ...
1 const sum = (a, b) => a + b;
2 const mul = (a, b) => a * b;
3 const g = 9.8;
4 const PI = 3.14;
5
6 let obj = {
7   sum: sum,
8   mul: mul,
9   g: g,
10  PI: PI,
11 };
12
13 module.exports = obj;
14
```

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
123
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
hello!
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
{}
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
{ sum: [Function: sum], mul: [Function: mul], g: 9.8, PI: 3.14 }
shradhakhapra@Shradhas-MacBook-Air Backend %
```

```
JS script.js x JS math.js
JS script.js > ...
1 const math = require("./math");
2
3 console.log(math);
4
```

Another ways to export:

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
123
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
hello!
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
{}
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
{ sum: [Function: sum], mul: [Function: mul], g: 9.8, PI: 3.14 }
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
4
3.14
shradhakhapra@Shradhas-MacBook-Air Backend %
```

```
JS script.js JS math.js 1
JS math.js > ...
1 const sum = (a, b) => a + b;
2 const mul = (a, b) => a * b;
3 const g = 9.8;
4 const PI = 3.14;
5
6 module.exports = {
7   sum: sum,
8   mul: mul,
9   g: g,
10  PI: PI,
11 };
12
```

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
123
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
hello!
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
{}
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
{ sum: [Function: sum], mul: [Function: mul], g: 9.8, PI: 3.14 }
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
4
3.14
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
4
3.14
shradhakhapra@Shradhas-MacBook-Air Backend %
```

```
JS script.js JS math.js x
JS math.js > PI
1 module.exports.sum = (a, b) => a + b;
2 module.exports.mul = (a, b) => a * b;
3 module.exports.g = 9.8;
4 module.exports.PI = 3.14;
5
6 // module.exports = {
7 //   sum: sum,
8 //   mul: mul,
9 //   g: g,
10 //   PI: PI,
11 // };
12
```

The image shows a development environment with a terminal and a code editor. The terminal window on the left displays the following commands and output:

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
123
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
hello!
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
{}
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
{ sum: [Function: sum], mul: [Function: mul], g: 9.8, PI: 3.14 }
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
4
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
4
3.14
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
4
3.14
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
4
3.14
shradhakhapra@Shradhas-MacBook-Air Backend %
```

The code editor on the right shows the file 'math.js' with the following code:

```
JS math.js > [?] PI
1  exports.sum = (a, b) => a + b;
2  exports.mul = (a, b) => a * b;
3  exports.g = 9.8;
4  exports.PI = 3.14;
5
6  // module.exports = {
7  //   sum: sum,
8  //   mul: mul,
9  //   g: g,
10 //   PI: PI,
11 // };
12
```

module.exports

requiring directories

main()

1) require → index.js
(entry point)

APNA COLLEGE

45


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

module.exports a special object

When we want to export and require file from different directories then we create an index file in that directory in which we take data from all the files in that directory and then return the array of that and then require it somewhere else.

Hence nesting of export and require.

```
shradhakhapra@Shradhas-MacBook-Air Fruits % touch apple.js
shradhakhapra@Shradhas-MacBook-Air Fruits % touch banana.js
shradhakhapra@Shradhas-MacBook-Air Fruits % touch orange.js
shradhakhapra@Shradhas-MacBook-Air Fruits %
```




```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js
[
  { name: 'apple', color: 'red' },
  { name: 'banana', color: 'yellow' },
  { name: 'orange', color: 'orange' }
]
shradhakhapra@Shradhas-MacBook-Air Backend %
```

```
JS script.js x
JS script.js > info
1 const info = require("../Fruits");
2
3 console.log(info);
4
```

NPM (Node Package Manager)

npm is the standard package manager for Node.js.

- ① library of packages
- ② command line tool

code

Installing Packages

node_modules The node_modules folder contains every installed dependency for your project.

package-lock.json It records the exact version of every installed dependency, including its sub-dependencies and their versions.

Whenever we require packages then we do not have to write ./

```
shradhakhapra@Shradhas-MacBook-Air FigletDir % ls
shradhakhapra@Shradhas-MacBook-Air FigletDir % npm install figlet
added 1 package, and audited 2 packages in 783ms
found 0 vulnerabilities
shradhakhapra@Shradhas-MacBook-Air FigletDir % ls
node_modules      package.json
package-lock.json
shradhakhapra@Shradhas-MacBook-Air FigletDir % node index.js
Hello World
shradhakhapra@Shradhas-MacBook-Air FigletDir % node index.js
APNA College
shradhakhapra@Shradhas-MacBook-Air FigletDir % node index.js
Shradha
shradhakhapra@Shradhas-MacBook-Air FigletDir %
```

```
{ package-lock.json JS index.js x
FigletDir > JS index.js > ...
1 const figlet = require("figlet");
2
3 figlet("Shradha", function (err, data) {
4   if (err) {
5     console.log("Something went wrong...");
6     console.dir(err);
7     return;
8   }
9   console.log(data);
10 });
11
```

package.json

The package.json file contains descriptive and functional **metadata** about a project, such as a name, version, and dependencies

npm init

Package.json contains the data that is contained in the node modules and what all to install if it gets deleted. So we send the package.json file and do not send the node modules folder.

Npm init

If want to leave the name of the project as it is.

Set version to whatever you want to. Leave it for 1.0.0

Set the description for your project.

Set the entry point. Leave it to index.js

Enter the rest

In author write your own name.

Licence : leave empty and enter

Local v/s Global

`npm install -g <- package name ->`

`npm link <- package name ->`

require v/s import

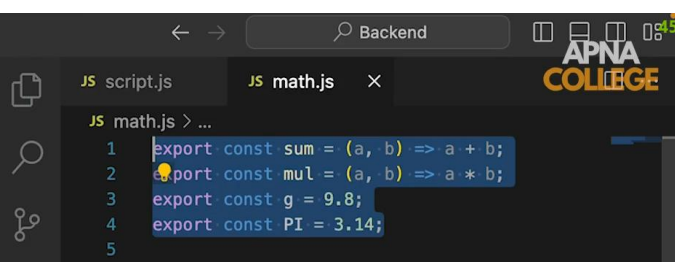
`import { sum } from "./math.js"`

export
type → module

We can't selectively load only the pieces we need with require but with import, we can selectively load only the pieces we need, which can save memory.

Loading is synchronous for 'require' but can be asynchronous for 'import'.

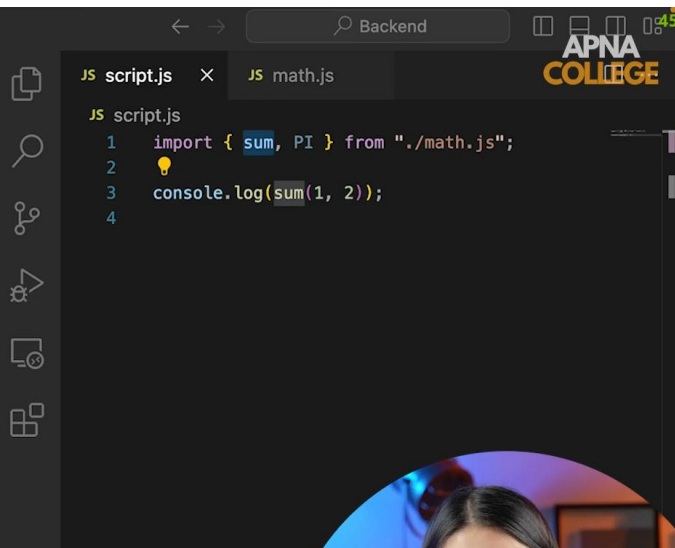
shradhakhapra@Shradhas-MacBook-Air Backend %

A screenshot of a code editor with two tabs: 'JS script.js' and 'JS math.js'. The 'JS math.js' tab is active, showing the following code:

```
1 export const sum = (a, b) => a + b;  
2 export const mul = (a, b) => a * b;  
3 export const g = 9.8;  
4 export const PI = 3.14;  
5
```

```
shradhakhapra@Shradhas-MacBook-Air Backend % node script.js  
(node:75659) Warning: To load an ES module, set "type": "module"  
in the package.json or use the .mjs extension.  
(Use `node --trace-warnings ...` to show where the warning was created)  
/Users/shradhakhapra/WebClassroom/Backend/script.js:1  
import { sum, PI } from "./math.js";  
^^^^^^
```

```
SyntaxError: Cannot use import statement outside a module  
    at Object.compileFunction (node:vm:352:18)  
    at wrapSafe (node:internal/modules/cjs/loader:1031:15)  
    at Module._compile (node:internal/modules/cjs/loader:1065:27)  
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1153:10)  
    at Module.load (node:internal/modules/cjs/loader:981:32)  
    at Function.Module._load (node:internal/modules/cjs/loader:822:12)  
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)  
    at node:internal/main/run_main_module:17:47  
shradhakhapra@Shradhas-MacBook-Air Backend %
```

A screenshot of a code editor with two tabs: 'JS script.js' and 'JS math.js'. The 'JS script.js' tab is active, showing the following code:

```
1 import { sum, PI } from "./math.js";  
2  
3 console.log(sum(1, 2));  
4
```

We cant directly use this we need to run npm init and then in the package.json we need to add "type": "module".

JS script.js {} package.json X JS math.js ...45

APNA COLLEGE

package.json > type

```
1  {
2    "name": "backend",
3    "version": "1.0.0",
4    "description": "",
5    "main": "math.js",
6    "dependencies": {
7      "figlet": "^1.6.0"
8    },
9    "devDependencies": {},
10   "scripts": {
11     "test": "echo \"Error: no test specified\"
12   },
13   "author": "ShradhaKhapra",
14   "license": "ISC",
15   "type": "module"
16 }
17
```

/pack