

What is a Module in JS?

Ans:

- A module is a piece of reusable javascript code.
- It could be a .js file or a directory containing .js files.
- You can export the content of these files and use them in other files.
- It help to break down complex logic to small, simple , and manageable chunks.

What are the different types of modules?

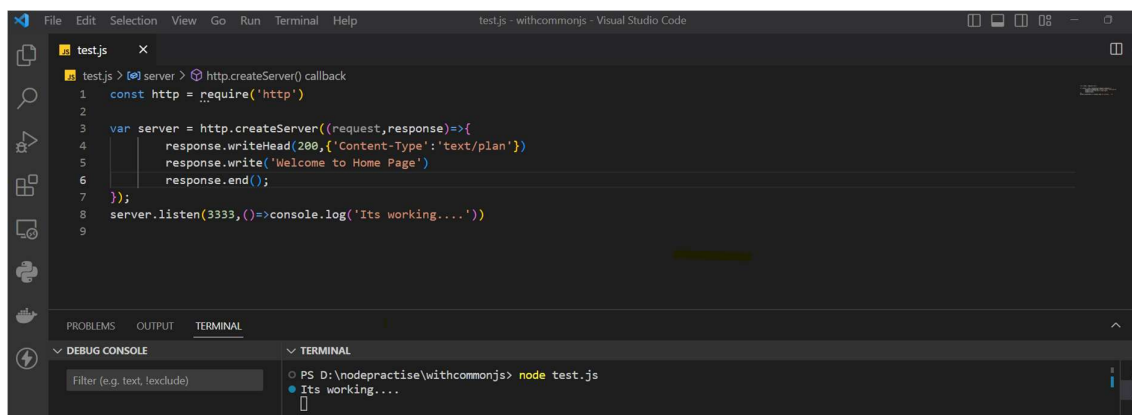
Ans:

- They are the following types of moudles :
 - Build-in Modules
 - Local Modules
 - Third-party Modules

1. Build-in Modules :

- Node.js comes with some modules. These modules are available for use when you install Node.js.
- The following are the modules:
 - http
 - path
 - fs
 - url
 - os
 - Etc...
- If you want to use then we need to include the following way:
 - `var someVariable = require('nameOfModule');`
 - Here you have loaded the module with the require function.
 - The name of the module must be include in single quotation marks. Also, using const to declare to the variable ensures that you do not override the value when calling it.

Example:



The screenshot shows the Visual Studio Code editor with a file named `test.js` open. The code in the file is as follows:

```
1 const http = require('http')
2
3 var server = http.createServer((request,response)=>{
4   response.writeHead(200,{ 'Content-Type': 'text/plain'})
5   response.write('Welcome to Home Page')
6   response.end();
7 });
8 server.listen(3333,()=>console.log('Its working....'))
9
```

Below the editor, the TERMINAL panel is visible, showing the command `node test.js` being executed in a PowerShell prompt, with the output `Its working....`.

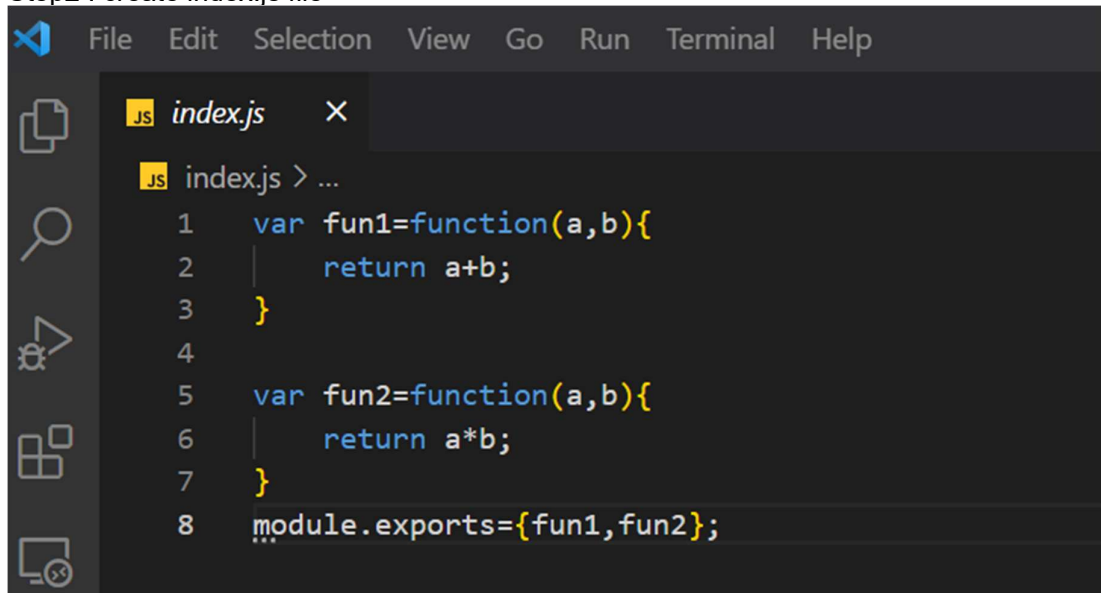
- Here with http object we call a property `createServer()` to create a server.

2. Local Modules:

Approach 1 : By using commonjs

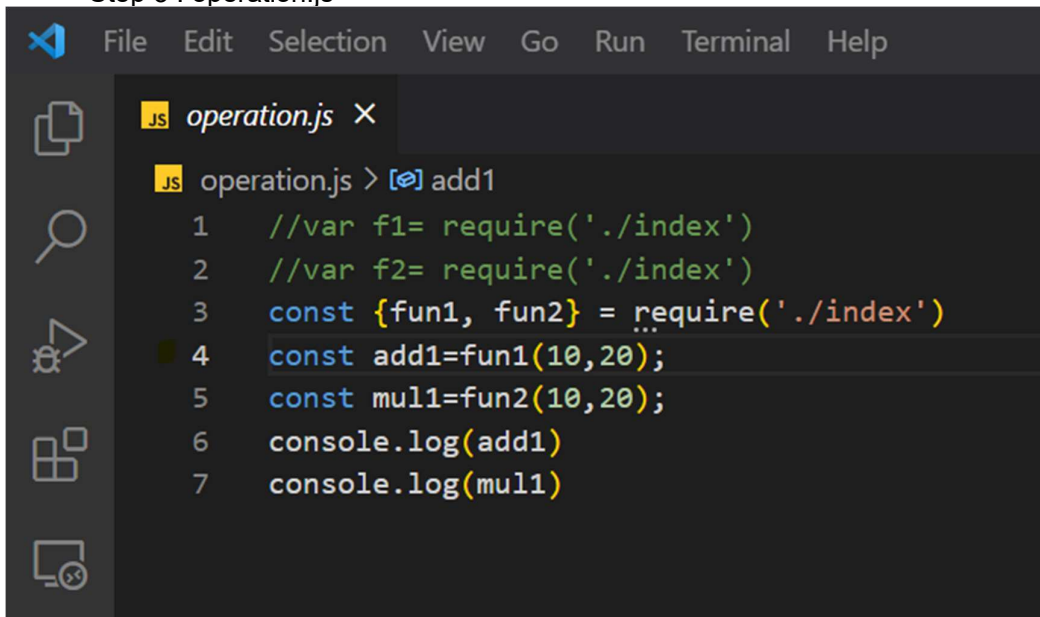
- When you work with Node.js , you create local modules that you load and use in your program.

- Let's create a simple module which contain multiple functions and use by another js file.
- Step1 : npm init -y
- Step2 : create index.js file

A screenshot of the Visual Studio Code editor interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The left sidebar shows icons for Explorer, Search, Run and Debug, Extensions, and Remote Explorer. The main editor area has a tab for 'index.js'. The code in the editor is as follows:

```
1  var fun1=function(a,b){
2      return a+b;
3  }
4
5  var fun2=function(a,b){
6      return a*b;
7  }
8  module.exports={fun1,fun2};
```

- Step 3 : operation.js

A screenshot of the Visual Studio Code editor interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The left sidebar shows icons for Explorer, Search, Run and Debug, Extensions, and Remote Explorer. The main editor area has a tab for 'operation.js'. The code in the editor is as follows:

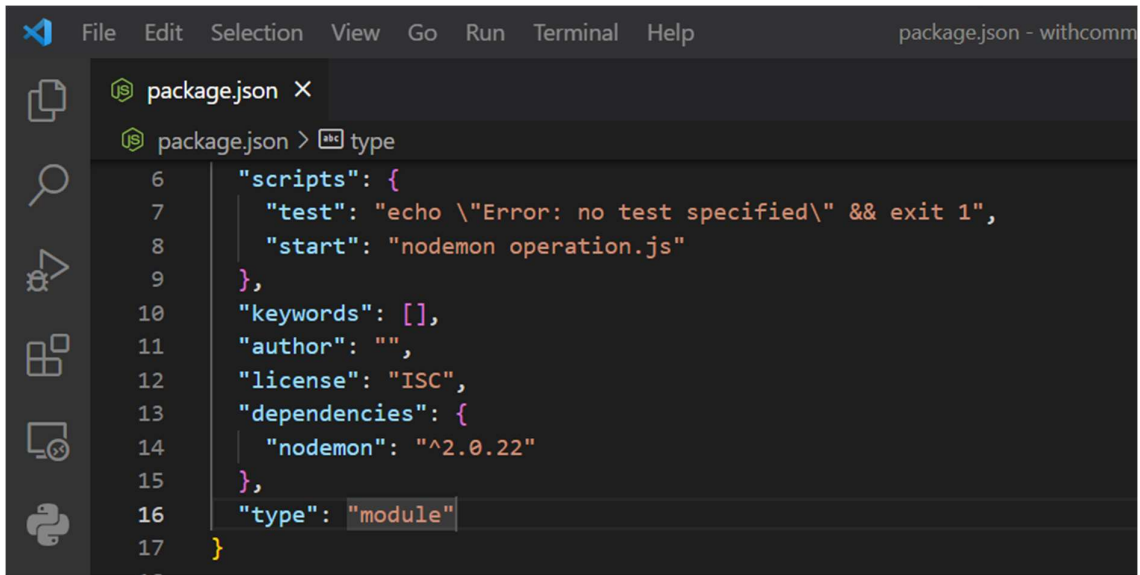
```
1  //var f1= require('./index')
2  //var f2= require('./index')
3  const {fun1, fun2} = require('./index')
4  const add1=fun1(10,20);
5  const mul1=fun2(10,20);
6  console.log(add1)
7  console.log(mul1)
```

Approach2 : By using ESM Modules

Step1: Create a simple node.js project

>npm init -y

Step2: Change “type”:”module” inside the package.json to use ESMModules approach.



The screenshot shows the Visual Studio Code editor with the 'package.json' file open. The file content is as follows:

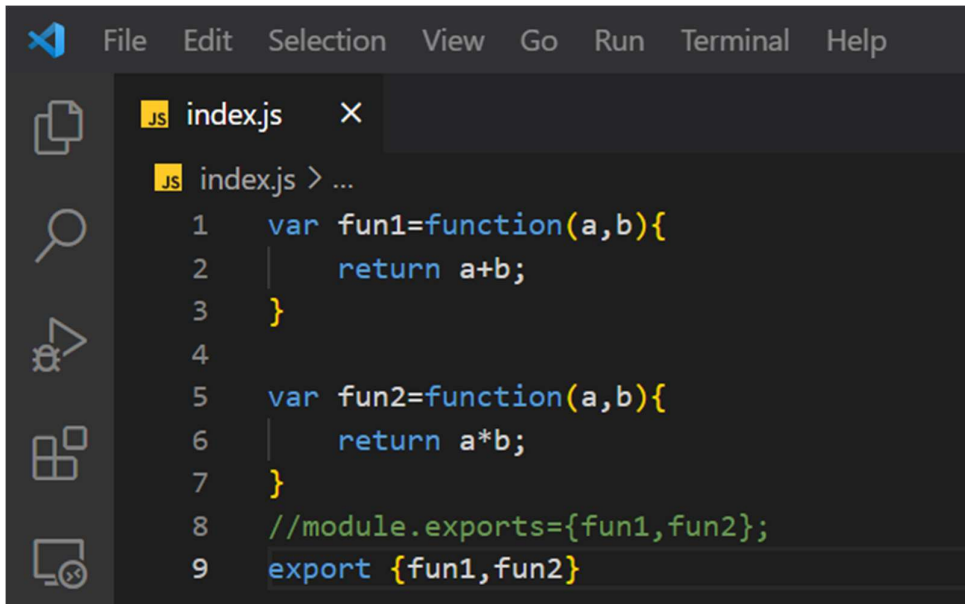
```
6  "scripts": {
7    "test": "echo \"Error: no test specified\" && exit 1",
8    "start": "nodemon operation.js"
9  },
10 "keywords": [],
11 "author": "",
12 "license": "ISC",
13 "dependencies": {
14   "nodemon": "^2.0.22"
15 },
16 "type": "module"
17 }
```

The text `"type": "module"` on line 16 is highlighted with a mouse cursor.

Step3: Install package nodemon

```
>npm i nodemon
```

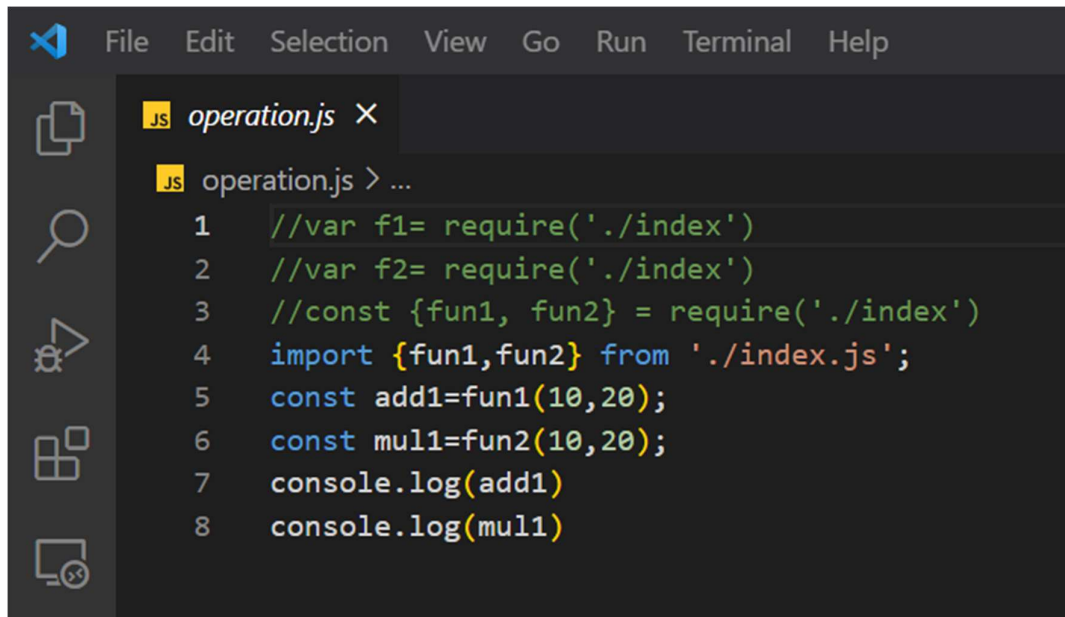
Step4: Create the index.js file



The screenshot shows the Visual Studio Code editor with the 'index.js' file open. The file content is as follows:

```
1  var fun1=function(a,b){
2    |   return a+b;
3  }
4
5  var fun2=function(a,b){
6    |   return a*b;
7  }
8  //module.exports={fun1,fun2};
9  export {fun1,fun2}
```

Step5: Create the operation.js file



The screenshot shows the Visual Studio Code editor with a file named `operation.js` open. The code in the file is as follows:

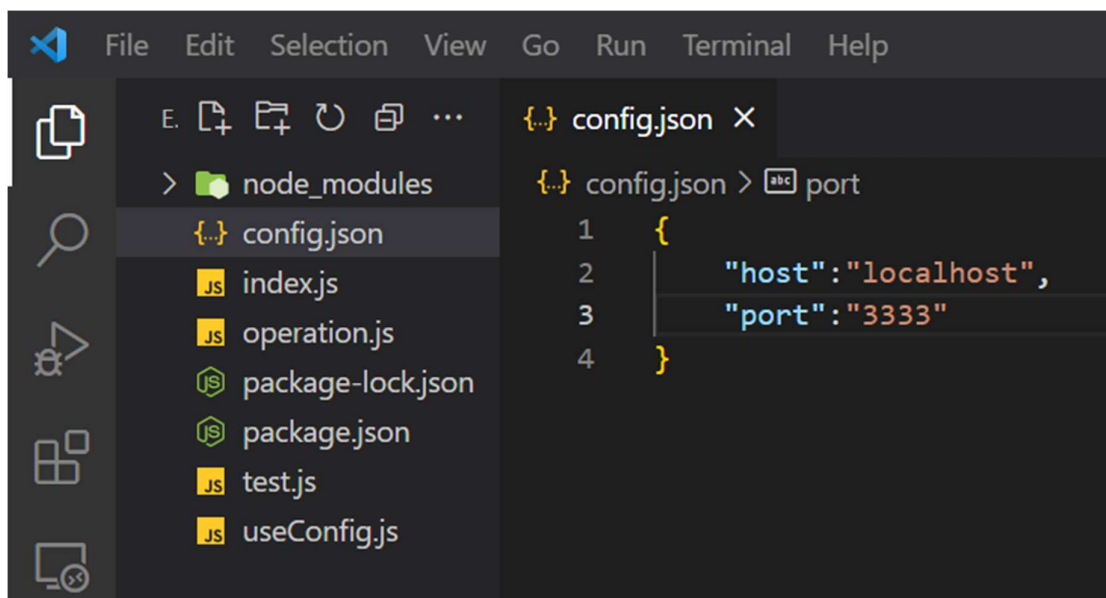
```
1 //var f1= require('./index')
2 //var f2= require('./index')
3 //const {fun1, fun2} = require('./index')
4 import {fun1,fun2} from './index.js';
5 const add1=fun1(10,20);
6 const mul1=fun2(10,20);
7 console.log(add1)
8 console.log(mul1)
```

Config.json file

- We can also work with config.json file where we configuration the information about host , port number and other details.
 - In order to implements the following steps can be involved.
- Step1: Create the node.js project

> npm init -y

Step2: Create the config.json file



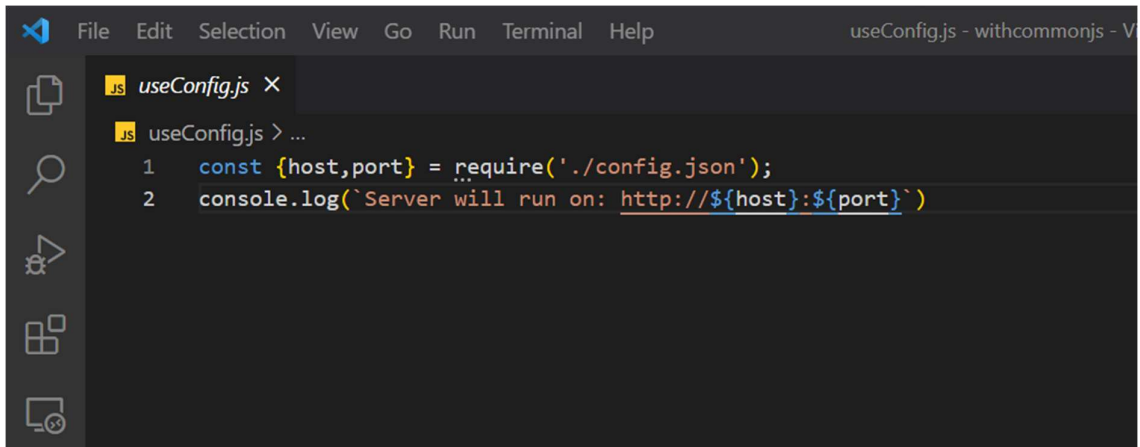
The screenshot shows the Visual Studio Code editor with a file named `config.json` open. The code in the file is as follows:

```
1 {
2   "host": "localhost",
3   "port": "3333"
4 }
```

The file explorer on the left shows the following files in the project:

- `node_modules`
- `config.json`
- `index.js`
- `operation.js`
- `package-lock.json`
- `package.json`
- `test.js`
- `useConfig.js`

Step3: Create the useConfig.js file



The screenshot shows a code editor with a dark theme. The top menu bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The title bar of the editor window reads 'useConfig.js - withcommonjs - V'. On the left, there is a sidebar with icons for Explorer, Search, Run and Debug, Extensions, and a terminal icon. The main editor area displays the content of 'useConfig.js' with two lines of JavaScript code:

```
1  const {host,port} = require('./config.json');  
2  console.log(`Server will run on: http://${host}:${port}`)
```
