

## Episode - 04 → Module export And Require

→ We may need to use code of another module into our module, for this 'require' is used.

ex → `require("xyz.js")` → it is basically import

Require → To import a module inside another module

→ is available to us always in module like global object.

→ But by simply require a module, we cannot access directly the variables and methods in that module.

→ Modules protect their variables and functions from leaking.

Module.Export → It is used to give access to variables and functions explicitly of that module

ex → in module which will be exported → calsum.js

```
var x = "Hello"
function sum(a,b){
  const sum = a+b
  console.log(sum)
}
module.exports = sum.
```

In main file → app.js

```
const sum = require('calsum.js')
var a = 10
var b = 20
sum(a,b)
```

⇒ Above works only for single function or variable.

To export more than one function or variable or Both ⇒ Wrap them in Object

```
module.exports = {
  x: x
  sum: sum
}
```

Note: By `require("filename.js")` → only code is executed but we cannot access variables and functions. For accessing ⇒ `module.exports` is needed

In app.js →

```
const obj = require("./calsum.js")
obj.sum(a,b)
console.log(obj.x)
```

• We can also use De-structuring of object

```
const {x, calsum} = require("./calsum.js")
```

• in calsum.js → To export we can also write ⇒

```
module.exports {x, sum}
```

it is same  
as this only

```
module.exports {x: x,
  sum: sum}
```

Just a shortcut  
to write.



→ require allows you to import the module and executes the code of that module but it won't allow the module which is import another module, The master module will not get Access to variables and functions of imported module unless that module allows by using module.exports = { }

Why Modules are Protecting their Variables And functions?  
Ans. To prevent conflicts which may arise due to same name of variables and functions

Note: `require("./calculator")`  $\xrightarrow{\text{same as}}$  `require("./calculator.js")`  
• extension is not Necessary to write

Above way of importing and exporting modules is common JS (CJS) module.

There are two types of module pattern followed →

### ① Common JS Modules (CJS)

To export → `module.exports`

To import → `require()`

- By default used in NodeJS
- Older Way

ex → 

```
function print() {  
  console.log("Hi")  
}  
module.exports = print
```

 → export

const ~~z~~ print = `require("./file.js")`  
↳ import

• Synchronous way of importing modules

• Non-Strict Mode

{ you can define variables without using `let`, `var` and `const` }

ex → `z = "Hello World"` → throws No error

### ② ES Modules (mjs) (ESM)

To export → `export`

To import → `import`

- By default used in React, Angular
- Newer Way

ex → 

```
export function print() {  
  console.log("Hi")  
}
```

 → export

`import { print } from "./file.js"`  
↳ import

• Asynchronous way of importing Modules

• Strict Mode

{ you cannot define variables without using `let`, `var` and `const` }

ex → `z = "Hello World"` → throws error

Hence we

`var z = "Hello World"`

What is Module.exports? → `console.log(module.exports)` → `{ }`

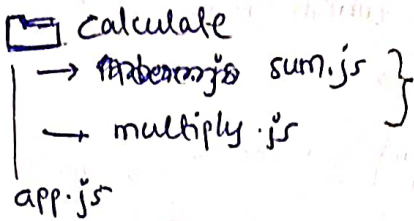
It is an empty object.

Hence another way to write exports is: →

`module.exports.x = x;`

`module.exports.sum = sum;`

## # Organizing Modules →

① into folders ⇒ ex:  `calculate`  
→ `sum.js`  
→ `multiply.js`  
`app.js` } two modules

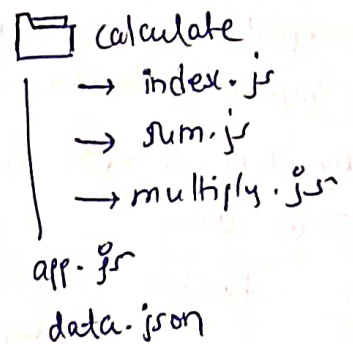
How To import in `App.js`: `const { calsum } = require("./calculate/sum.js")`  
`const { calmul } = require("./calculate/multiply.js")`

## ② Making the folder as a Module.

① create `index.js` in that folder → import the two functions from their module in this `index.js` and export it

ex: `index.js` →

```
const { calsum } = require("./sum.js")
const { calmul } = require("./multiply.js")
module.exports = { calsum, calmul }
```

  
`calculate`  
→ `index.js`  
→ `sum.js`  
→ `multiply.js`  
`app.js`  
`data.json`

`app.js` →

```
const { calsum, calmul } = require("./calculate")
```

Hence The `calculate` folder becomes a Module which gives two functions.

→ efficient for a larger No. of Modules

## # Importing .json file

→ No need to write export in .json file, we can directly import it in our main file.

ex: with reference to above example →

`data.json` `app.js`

```
{ "name": "Ankit",
  "city": "Hyderabad"
}
```

```
const data = require("./data.json")
console.log(data)
```