Module. export & require

How to Run two JS Files together

app.js
xyz.js

```
app.js

require("./xyz.js")
var a = 10;
```

↑ The code within xyz.js will run first. i.e. it will executed line by line firstly then code within app.js will run.

(*) whenever you create a seperate module and you require that module. This code will run but you cannot access the variable, method function of one module into other simply by requiring.

Note: Modules are protected by default.

(*) Modules protect their variables and functions from leaking.

(*) To access the functions & variables from a module, you need to export them from the module and import that function in another file.

To Export we use ⟶

module.exports function_name;

To Import we use ⟶

const function_name = reqnire("./file_name")

To Export more than one ~~person~~ fn and variables.

module.exports = { }

↑ wrap them in brackets.

points to Remember

From a module you cannot access its variable & functions, unless a module wants them to be acessed on exports them.

If module.exports was not there and we could access the variables and methods by reqnring the whole module. it could lead to redundecies and inconsistency.

require ("./app.js"); } Both works.
or
require ("./app");

The pattern we have seen till now was common JS Module. (CJS)

The other modules used are ES Module (MJS)

By default CJS is used, to enable MJS write "type": "Module" in package.json file. After this we cannot use CJS module method

common JS Module	ES Module
→ module.exports require();	→ import export
→ By default used by node.js.	→ By default used by React, Angular
→ "type": "commonjs"	→ "type": "module"
→ older way	→ Newer way
→ synchronous & Runs in non-strict mode	→ Asynchronous and Runs in script mode.

Note

module.exports is an empty object in JS file.

console.log (module.exports)

we can attach properties to it in two ways.

```
module.exports = { x, ~~file~~ file_name }
          OR
module.exports. x = x;
module.exports. function_name = fn_name
```

→ Require Multiple functions from a same file which was defined in ~~difined~~ different files.

Make a new file:

```
const {function_from_file1} = require("./file1")
const {fn_from_file2} = require("./file2.js");

module.exports{fn_from_file1 . fn_from_f_2}
```