9/12/25

# Diving into the Nodejs github repo #

① require("./xyz.js"); → ye ek fn mai ~~wrap~~ rap karta hai aur fir execute karta hai.

```
function () {
//All the code of the module is run inside this
function
}
```

- why you can't accus the fn and variables of one module to another:

Ans → Because it is wrap inside a function.

- require("./path")
  All the code of the module is wrapped inside a function'

→ function () → IIFE (immediately invoked ~~fn~~ function expression)

IIFE → independent code

```
(function () {
// All code of the module run inside here

})();
```

→ Immediately invokes the code
→ Privacy → keeps variable & function safe

= ~~(functi~~

__/__/____

* (function (exports , require , module , __filename , __dirname) {

});

* How are variables & functions private in different module?

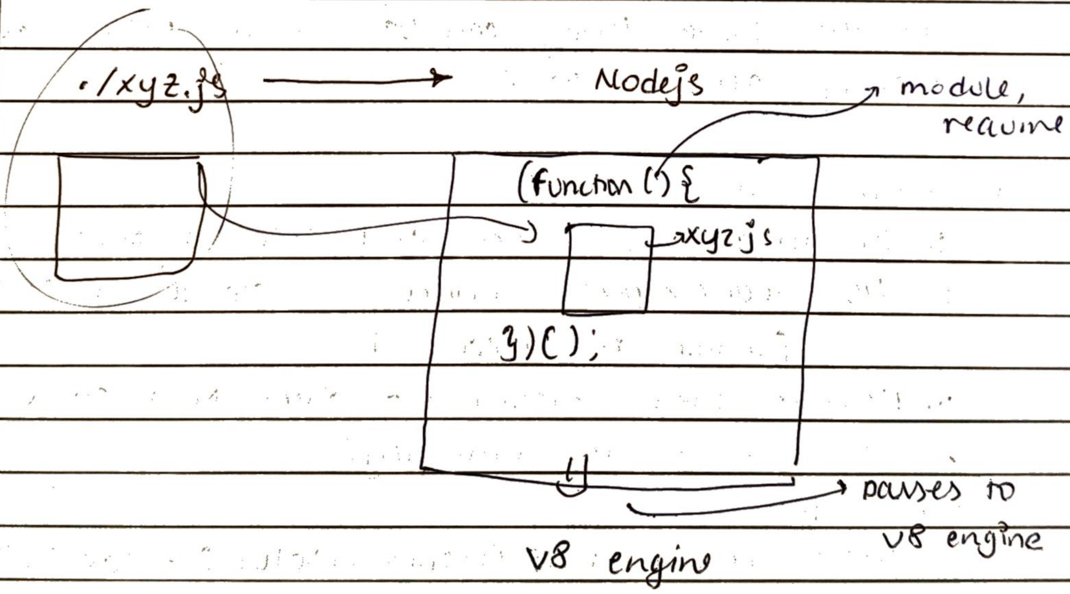
→ IIFE & require (statement)
{ (wrapping code)

* How do you get access to module.export :

• function (module, require, ...) {

}) (module,..);

→ Nodejs passes module as a parameter to the IIFE.



_/_/_

# require (/path)

① Resolving the module.
→ ./localpath → node:module
→ ./json

② Loading the module.
↳ file context is loaded according to file type
~~acc to fil.~~

③ wraps inside IIFE

④ Code evaluation
↳ module.exports

code of require will only run once

⑤ caching
↳ storing data temporarily so that future request for the same thing can be served faster without redoing all the work.

• caching in require():
1. The module is loaded and executed once.
2. Its module.exports object is stored in memory (inside require.cache)
3. Require same module → gives from cache.
↳ to clear cache manually:
~~to~~ de

```
delete require.cache(require.resolve(". /counter"));
```