Nitesh

Namaste Node.js

Chapter - 05 - Diving into the NodeJS Github Repository

÷ Module working behind the scenes.

Q) What happens when you required a module ? {require ("./xyz.js");}

A)) i) All the variables inside a function are privately scoped (not accessible outside that function).

ii) Module also behaves in same way as Javascript functions.

iii) Whenever you require a module, NodeJS wraps that module inside a function (an IIFE) (created by NodeJS) and then execute it.

iv) It must be exported to the using module for making it accesible.

v) Before giving the module code to V8, it is wrapped inside an Immediately Invoked Function Expression.
- → immediately invokes the code.
- → privacy (keep variables and functions private).
- → doesn't pollute the scope.

Q) How are variables and functions remain private in different module?

A) Because of the require statement which wraps the code inside module into an IIFE.

Q) How do you get access to module.exports?

A) i) NodeJs made it available to your module.

ii) When our code gets wrapped inside the function has a parameter module (an empty object) and when it gets invoked it passed as an argument.
(same goes for "require" as well).

e.g., `(function(module, require){`

---- Module code

`})();`

these parameter provided by NodeJs.

iii) NodeJs passes module, require inside the IIFE.

module → an empty object

this code gets passed inside V8 Engine

one can also open Node.js github Repo to see all this and their code

Q) What happens when you "require(//path)".

A) There are 5 steps mechanism happens behind the scenes :— (to get code & execute it).

i) Resolving the module
→ from which type of file data is coming and accordingly resolves it.
↳ ./localpath
↳ .json
↳ node : module

ii) Loading the Module
→ file content is loaded according to file type.

iii) Wrapping inside IIFE (Compile)
→ module gets wraps inside IIFE so that its variables and functions remains function scoped and doesn't pollute the main scope. (wrapSafe)

iv) Evaluation
→ module.exports happens.
→ code gets executed.

v) Caching
→ module gets cached.

e.g. Suppose that the same file (or module) is required by multiple different modules.

(require ("/sameFile.js");

→ in this case "require" gets cached.
→ i.e., the code of require run only once.
→ and every time a module requires already cached module (code is transferred from cache, not the whole 5 step mechanism gets followed).

Haven't you learned Dynamic Programming

Haven't you learned COA, OS (Cache memory)

→ makes NodeJS much faster.
(doesn't need to execute same code again & again (if it has been executed once.)

↳ No need to solve the same subproblem again and again

Cache memory (frequently accessed file)

Dynamic Programming

Baby

(OP)