

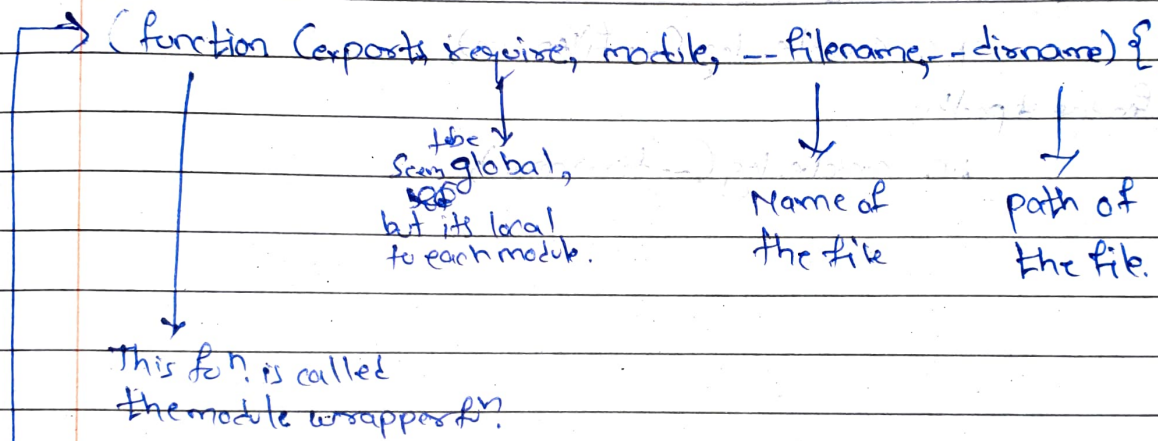
The var. & fn. we define in a module are scoped to that module, ~~they~~ they're private & not visible from the outside.



In `logger` module we don't need an object because we've a single method, so instead of exporting an object we can export a single function. So we ~~remove~~ `log` from `'module.exports.log = log;'`.

So, ~~it~~ initially it was an empty obj. but we reset it to just a fn. So now in app.js, `logger` is no ~~longer~~ ^{longer} an object it's a fn. that we can call directly.

* Module Wrapper Function:



Note: ~~does not~~ executes our code directly, it always wraps ~~the~~ code inside each module & something like this inside of a fn.

→ `require()`: It appears to be global but actually it's not global, in fact it's local to each module, so in every module `require` is one of the arguments that is passed to the function.

→ if we want to add a fn. to module that exports object we can either write e.g. `module.exports.log = log;`
`exports.log = log;`

* we cannot write like this eg
`exports = log; // module.exports`



because
this `exports` is a
reference to `module.exports`. (we can't
change that
reference).

→ Finding filename

i.e. `console.log(__filename);`

Finding path

i.e. `console.log(__dirname);`