The var. & fn. we define in a module are scoped to that module, 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, initially it was an empty obj. but we reset it to just a fn. So now in app.js, logger is no longer an object it's a fn. that we can call directly.

* Module Wrapper Function:

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

- require: to be scoped global, but its local to each module.
- module: Name of the file
- __filename: path of the file.

This fn. is called the module wrapper fn.

Node does not executes our code directly, it always wraps our 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 its local to each module, so in every module require is one of the arguments that is passed to this function.

→ if we want to add a fn. to module that exports object we can either write as eg. module.exports.log = log;
exports.log = log;.

* we cannot write like this e.g.

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 dirpath

i.e. console.log(__dirname);