So, in real world applications, every module might have several variables & functions. We only want to export a subset of these members to the outside, cuz we want to keep this module easy to use.

So, in our logger module (logger.js file) this URL is implementation detail, other modules don't need to know anything about this, they only need to call log fn.
So, we export these make it public & but keep the URL private

* Loading a Module:

→ To load a module we use the `require fn`

→ This is one of the fn's in node, we don't have this in browsers.
This fn takes one argument & that's the `name` or `path` of the target module we want to load.

→ we use `'./'` to indicate the current folder & then we add the name of our module.
For parent folder we use `'../'`

→ This `require();` fn returns the object that is exported from this target module.

→ When we define a module, we export one or more members & then to load the module we use the `require()` fn.

→ In recent JS version, we can define `constants`..
So, when loading the module using the required fn. its better to store the result in a constant e.g. `const logger = require('./logger');`

Object will be useful only if we've multiple methods or properties.

Reason for this is, we don't want to accidentally overwrite the value of logger

eg

```
var logger = require('./logger');
```

```
logger = 1;
logger.log('message');
```

if we set this to 1 then when we call the log method we're going to get an exception.

O/P: logger.log is not a fn. (error).

→ if you accidently reset this object then we're gonna get an error at compile time instead of at run time.

→ Tools:

One of the popular tools in Js to prevent errors is Jshint

We can scan all our Js code for errors.

Imp:→ Sometimes instead of exporting an object from a module you want to export only a single function.

logger.js

```
var url = 'http: ...';
fn log(message) {
    console.log(message);
}

module.exports = log;
```

app.js

```
Const logger = require('./logger');
logger('message');
```

Date
Page

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 = logs'.

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.