**Node.js**

**Why Modules?**

To organize code into separate files => so we can reuse them through Node.js app.

**How to create modules?**

1. Create .js file. **Ex:** product.js
2. **Export** the module to use it in another file

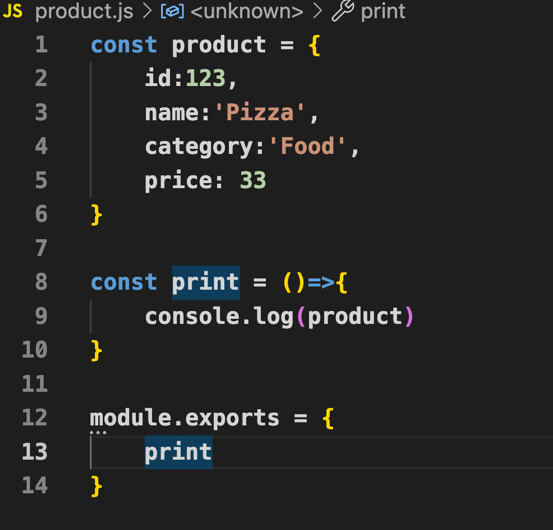
**Ex: 🡪** in product.js 🡪**module.exports** = {print}

that means that product.js file allow other files to reuse print method

1. **Import** the file you need to use it in your code.

**Ex:** 🡪in app.js 🡪 const {print} = **require**(‘./product.js’)

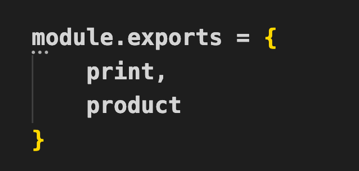
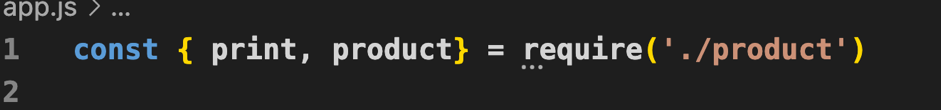
using built-in **require** function to include the file you want to reuse

 Text

Description automatically generated

**product.js app.js**

🡪 to add more objects to export

**app.js**

**product.js**

**There are three types of modules:**

1. **Core Module**
2. **Local Module**
3. **Third party module**

**🡪Core Module :** built in modules of node.js🡺 Ex: http, fs, path, url

<https://nodejs.org/docs/latest/api/>

How to use node.js core? **🡺 const http = require(‘http’)**

**🡪 local module:** you can create your local module like examples above (in app.js & product.js)

🡪 Third party modules: modules or package made by others (third party) and you can reuse them. Ex: express

How to use third party modules?

🡺 1- **install** the package using **npm 🡪** npm i express 🡪 (i= install)

2- **import** 🡪 const express = **require**(‘express’)

Note: **npm** (node package manager) is the command-line tool that allows you to install or uninstall node.js packages

**Let’s start our first node project**

**npm init 🡪** to start your project and creating **package.json 🡪 your new npm package**

**npm init -y 🡪 yes** to answer all questions using default

**Text

Description automatically generated Text

Description automatically generated**

**npm i express --save 🡪** to install express

**Text

Description automatically generated**

Note: if you want to uninstall 🡪 you can use **ui** instead of **i**

**Text

Description automatically generated**

Note:  **node\_modules** is created to save all downloaded packages

**Text

Description automatically generated**

**package-lock.json show the dependencies and the version**

**node\_modules. 🡪 big**

**So, we don’t push it to github**

create **.gitignore** file and add all files we don’t need to push it to github

**To initialize the server**

This ex from <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction> to initialize the server

The good news is **Express** (fast, lightweight framework)will help us to build the web server and listen without this code

**Graphical user interface, text, application, email

Description automatically generated**

1. **Build a web server**
2. **Import express module which is a third-party module**

**const express = require('express')**

**🡪 express () 🡪 return a function**

1. **Create app object of the express to use all the returning from express**

**🡪 we called it app by convention**

**const app = express ()**

1. **Identify the Port 🡪 You can’t run two apps in the same port**

**const port = 3000**

**app.listen(port, ()=> {**

**console.log('Listening from port ....')**

**})**

1. **Build RESTful APIs with Express:**

**RESTful** APIs defines a set of conventions to create http services

**GET, POST, PATCH, DELETE** 🡺 **CRUD** operations

**Ex: GET**

**🡪1st arg** is the **path** or url 🡪 in this ex, the path is the root (‘/’) 🡪 called the **endpoint**

**🡪2nd arg** is the **call back function** which will be called when **HTTP GET request happens.**

🡪Call back function takes two args **(req, res).**

**🡪req** object gives us information about the request. Ex🡪 req.body, req.params, req.cookies

🡪**res** object to respond or send an information or result. Ex🡪 res.send(‘…’)

**app.get('/', (req, res) => {**

**res.send('Hello All!')**

**})**

Text

Description automatically generated

Note 🡪 type: Request type

Get 🡪view

post🡪create

patch(put) 🡪update

delete 🡪 remove data

Localhost 🡪 to talk to your computer

Graphical user interface, text, application, email

Description automatically generated

header. 🡪 meta information

status code: 200, 204 ,404, 304, 500

you can see headers, status code, response 🡺 open **Network** tab 🡪 All

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated