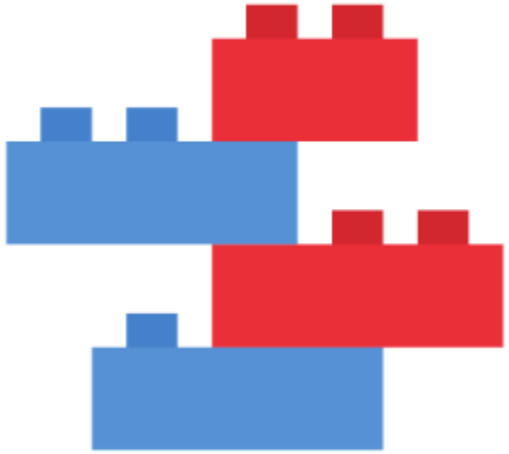
FRONT END DEVELOPMENT

WEEK 3 – Modules & Packages



CADT IDT

HOMEWORK

NEED TO BE DONE
BEFORE THE CLASS

Read documents & Answers question

✓ Documents to **READ** :

https://www.w3schools.com/js/js_modules.asp
https://www.freecodecamp.org/news/javascript-modules-explained-with-examples/
https://www.scaler.com/topics/nodejs/require-vs-import-nodejs/

✓ Questions to SUBMIT :

Q1 – What is a **module** ?

Q2 – What are the benefit of using **modules** ?

Q3 – What is the difference between **core modules** and **local modules**?

Q4 - What is a **NPM package** ?

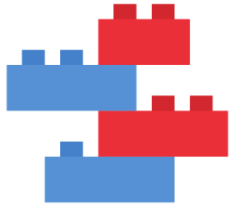
What will you learn today?

- ✔ Use **Local modules** — *"Programmer'' modules*
- ✔ Use **Core modules** — *Node build-in modules*
- ✔ Use **Third-Party Modules** *(NPM)* — *NPM imported modules*

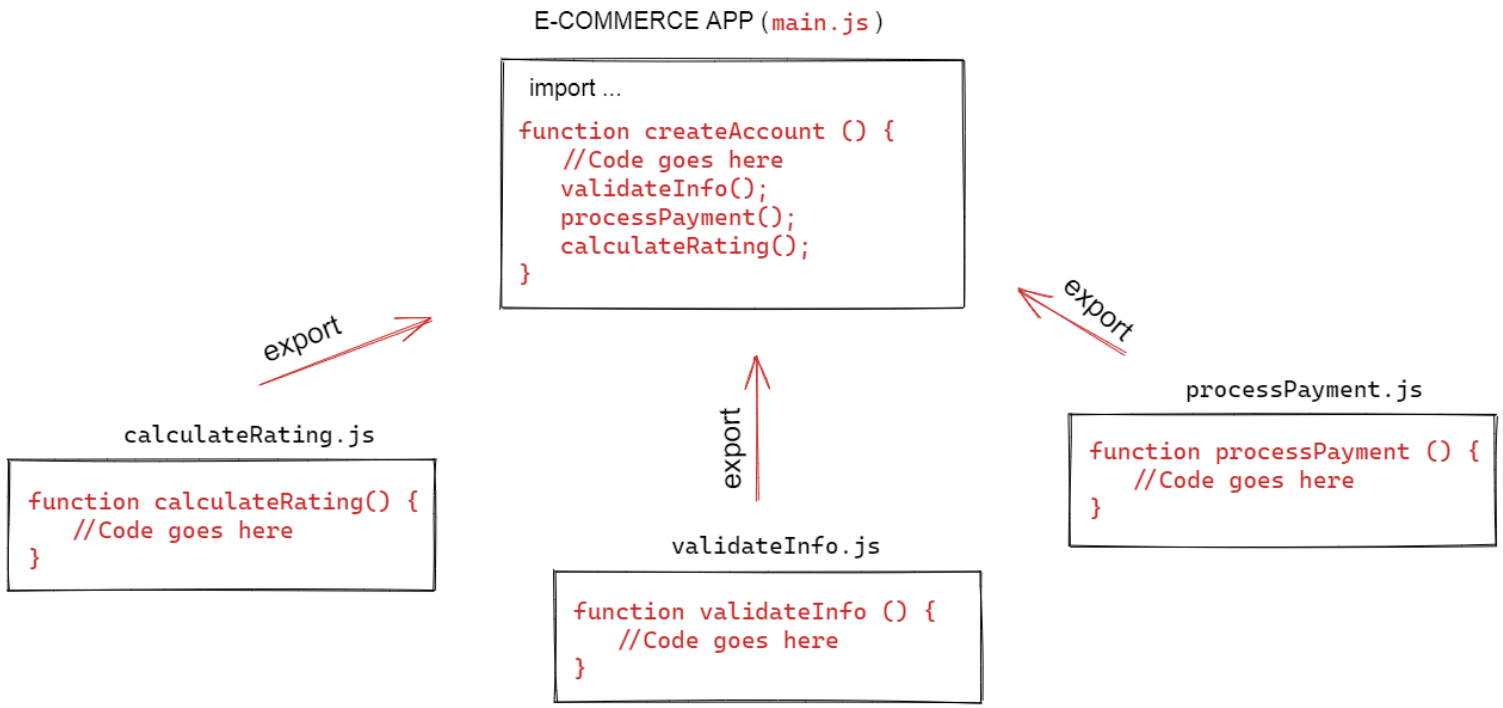
- ✔ Install a **npm package**
- ✔ Understand the differences btw **local**, **global** & **dev** packages

EXPLAIN

Why **Modules** ?



- ✓ Modules are **reusable blocks** of code that encapsulate some functionality
- ✓ They are **grouped together** within a file.
- ✓ We create modules to better organize and **structure** our code.



Here main.js has been broken down into four modules

EXPLAIN

3 types of modules

1

LOCAL MODULES

- ✓ Modules created by programmers **locally**

2

CORE MODULES

- ✓ **NodeJS built-in** modules

like: http, fs …

3

THIRD-PARTY MODULES

- ✓ Modules available after **NPM installation**

like: express, mongoose ...

EXPLAIN

2 module **systems**

We will use ES6 on this course !

NODE JS
MODULE SYSTEM

require | module.exports

ES6
MODULE SYSTEM

import | export

lib.js

```
const a = 25
module.exports = { a };
```

```
export const a = 25
```

main.js

```
const a = require('./lib.js')
console.log(a);
```

```
import a from "./myData.js"
console.log(a);
```

package.json

```
{
  "type": "module"
}
```

EXPLAIN

Use a LOCAL MODULE

```
{
  "type": "module"
}
```

package.json

Required for ES6 syntax

1

Export function or variables make them accessible

```
export const add = (a, b) => a + b;
export const subtract = (a, b) => a - b;
export const multiply = (a, b) => a * b;
export const divide = (a, b) => a / b;
```

Math.js

2

Import the functions or variable you need to use

```
import { add, subtract, multiply, divide } from './math.js';

// Use the functions from the math module
console.log(add(5, 3));      // Output: 8
console.log(subtract(10, 4)); // Output: 6
console.log(multiply(2, 6));  // Output: 12
```

Main.js

EXPLAIN

Different syntaxes *to import modules*

```
// Importing the entire module
import * as name from 'module_name'

// Importing the default export from the module
import name from 'module_name'

// Importing a single export from the module:
import { name } from 'module_name'

// Importing multiple exports from the module:
import { name1, name2 } from 'module_name'
```

ACTIVITY

10 MIN

Activity 1

*Use a **LOCAL MODULE***

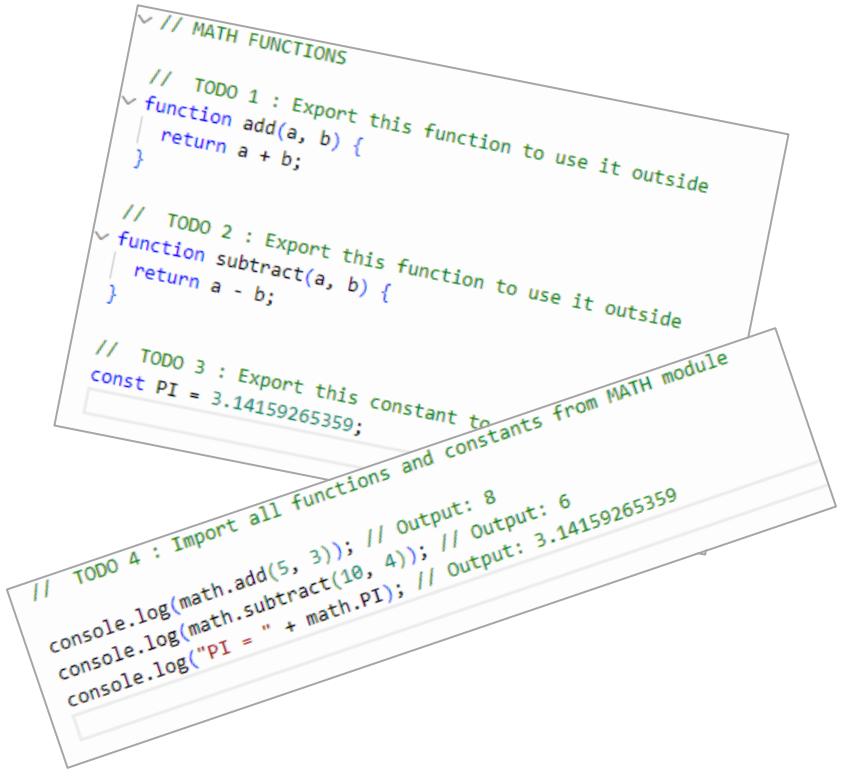
ES6 syntax

YOUR CHALLENGE :

1. In math.js : **Export** all function and constants
2. In main.js : **Import** all functions and constants from MATH module
3. In **package.json**: add the following property

```
{
  "type": "module"
}
```

4. **Run** the main.js to check the code run properly



ACTIVITY

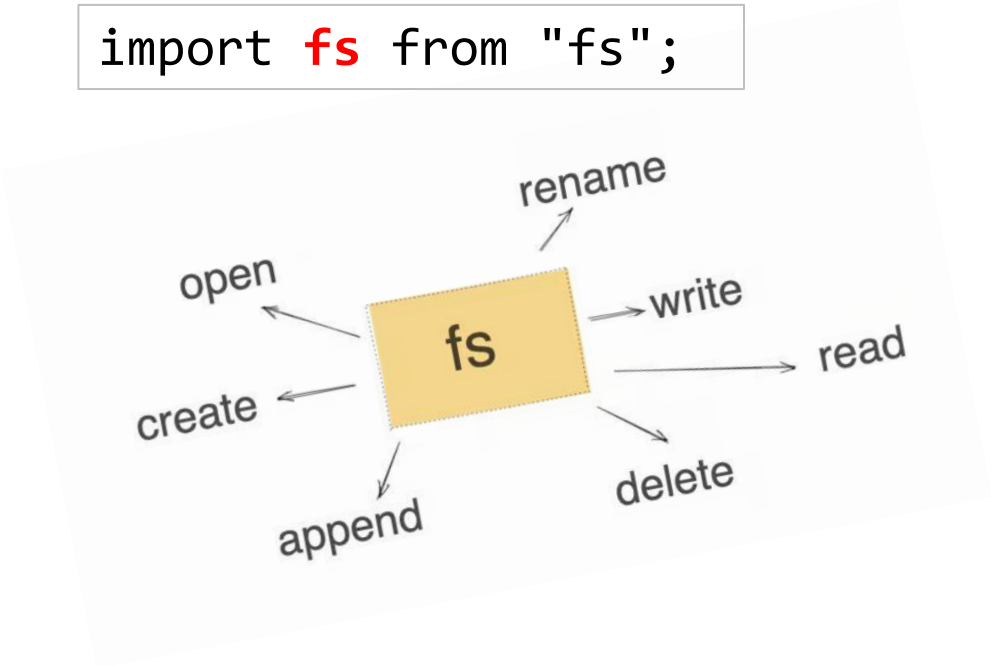
10 MIN

Activity 2

Use a CORE MODULE

FS is a **core module** containing functions to read and write files

```
import fs from "fs";
```



YOUR CHALLENGE :

1. Fix **bugs :**
   1 bug to fix in main.js
   3 bug to fix in logger.js

2. Re-write the function **writeLog**() using :
   `fs.appendFilesync()`

https://www.geeksforgeeks.org/node-js-fs-appendfilesync-function

EXPLAIN

Use a THIRD-PARTY MODULE - *LOCALY*

1

Install a package from **NPM** repository

https://www.npmjs.com

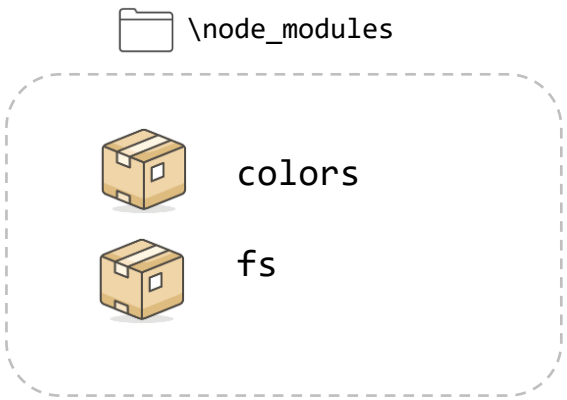
```
npm install colors
```

2

The **package dependencies** is added to your package.json

```
{
  "dependencies": { colors : ^1.4.0 }
}
```

The package modules are downloaded in node_modules



3

After installation you **can import** the colors modules in your code

```
// Import the colors module
import colors from 'colors';

// Apply colors to console output
console.log('Hello, '.green + 'world!');
```

ACTIVITY

10 MIN

Activity 3

Use a THIRD-PARTY MODULE - *LOCALY*

YOUR CHALLENGE :

1. Initialize npm in this project, by typing **npm init**
2. Install **npm package** "**colors**"
3. Import this package in main.js
4. Update package.json file adding the following line : "type": "module“

If everything is done great, you should be able to display colors on you logs !!

- For fun, display you best logs using this package

Want know more?

https://www.npmjs.com/package/colors

Npm package colors

EXPLAIN

*Understand **package.json** file*

Project information

Packages installed locally

Name of the package

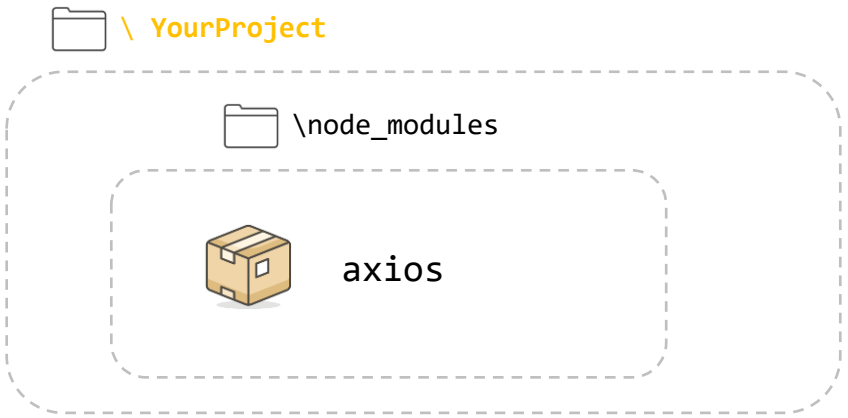
Version of the package

EXPLAIN

Use a **THIRD-PARTY MODULE** - *GLOBALY*

```
npm install axios
```

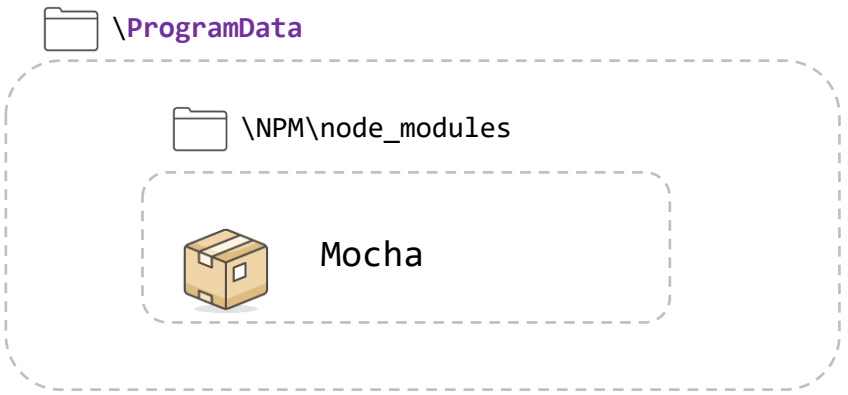
Available only
for your project



Local packages are installed in
your project folder

```
npm install -g mocha
```

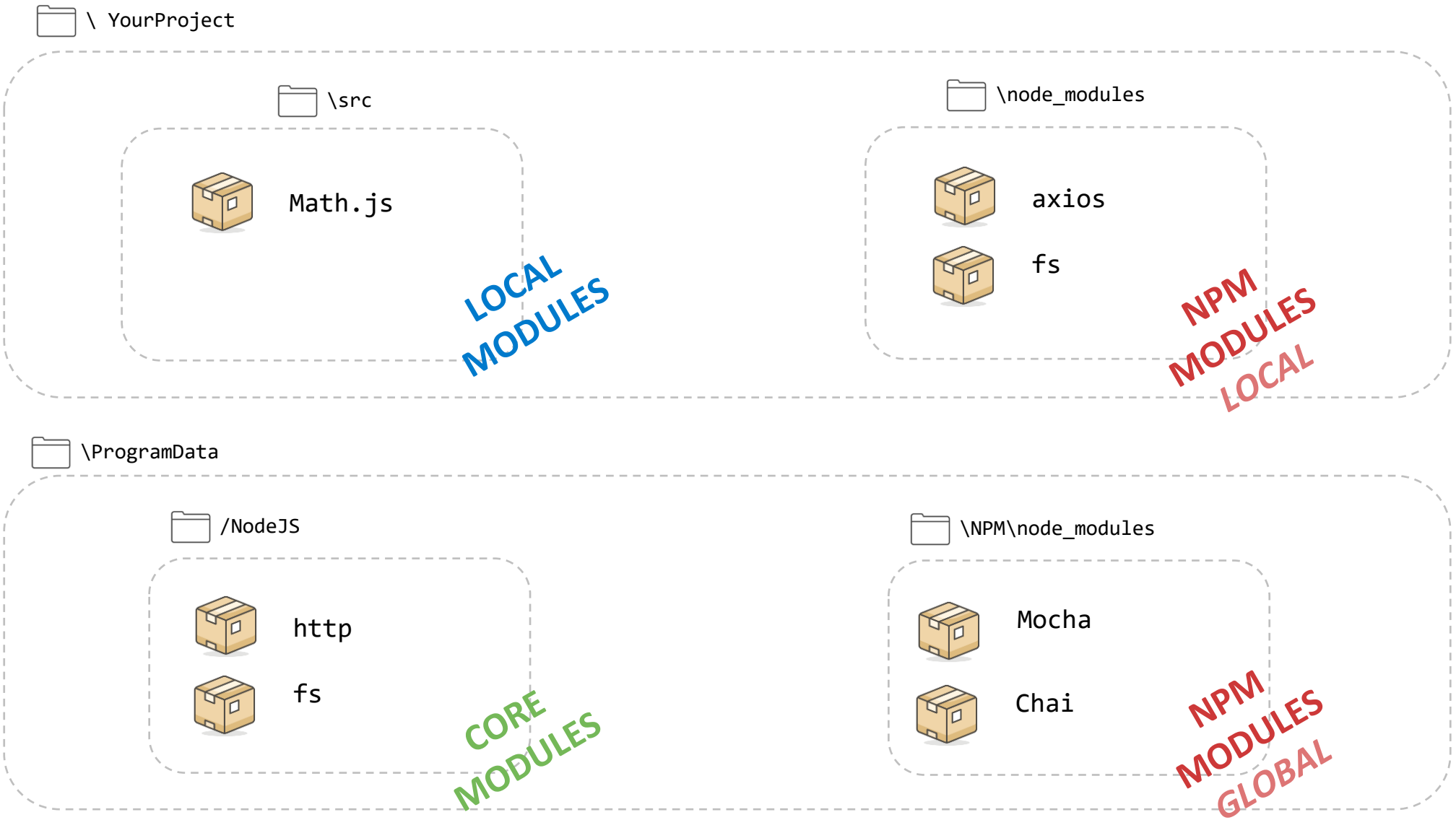
Available for
Any project



Global packages are installed in
your computer NPM folder

EXPLAIN

*Where are modules **located**?*



ACTIVITY

10 MIN

Activity 4

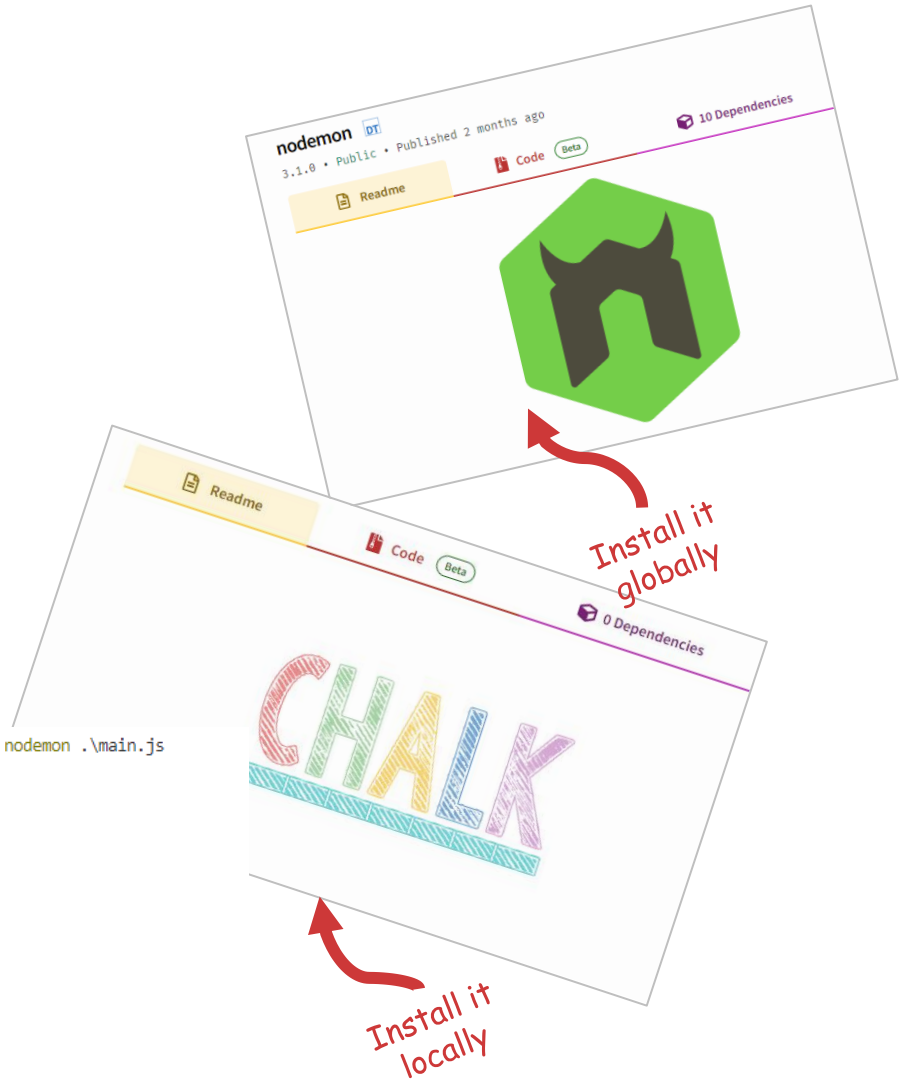
Use a THIRD-PARTY MODULE - *GLOBALY*

1. Install **Chalk** PNM package **locally**
   You can search it on : https://www.npmjs.com

2. Install **Nodemon** PNM package **globally**
   You can search it on : https://www.npmjs.com

3. Run the code using the command line :
   `nodemon ./main.js`

4. Check that **the execution re-start** if you change the main.js code



Nodemon restarts If the code has changed

```
PS C:\Users\LENOVO\Desktop\REACT JS\MY-CODE\PNV\C1\S4\Final code\exercice-4> nodemon .\main.js
[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node .\main.js`
Hello !
Success!
[nodemon] clean exit - waiting for changes before restart
[nodemon] restarting due to changes...
[nodemon] starting `node .\main.js`
Hello  ROnan !
Success!
[nodemon] clean exit - waiting for changes before restart
```

EXPLAIN

*NPM **CLI commands***

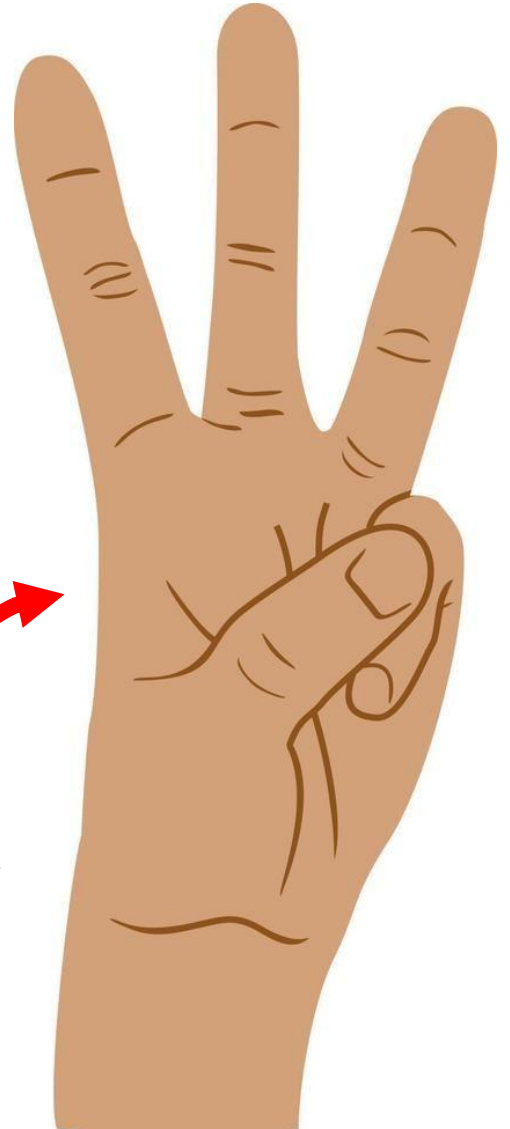
https://docs.npmjs.com/cli/v6/commands

`npm init`	Initializes a new package.json
`npm install <package>`	Installs a package to the project
`npm install`	Installs all dependencies listed in package.json
`npm uninstall <package>`	Uninstalls a package from the project
`npm update`	Updates dependencies to their latest versions.
`npm list`	Lists installed packages and their versions
`npm run <script>`	Executes custom scripts defined in package.json

Time For

QUIZ !

Use your fingers to vote !



Q1

What will we see on **console** ?

weather.js

```
export const paris = 25

const danang = 31
```

main.js

```
import * as weather from “weather.js”

console.log(weather.danang);
```

1. 31
2. 25
3. Error Cannot use import statement outside a module
4. undefined

Q1

What will we see on **console** ?

weather.js

```
export const paris = 25

const danang = 31
```

main.js

```
import * as weather from “weather.js”

console.log(weather.danang);
```

1. 31
2. 25
3. Error Cannot use import statement outside a module
4. undefined

Q2

What is the purpose of **package.json** file in Node.js project?

1. To store code
2. To configure server settings
3. To define project metadata and dependencies
4. To serve as the entry point for the application

Q2

What is the purpose of **package.json** file in Node.js project?

1. To store code
2. To configure server settings
3. To define project metadata and dependencies
4. To serve as the entry point for the application

Q3

What is the difference between PNM local and global packages ?

1	Local	installed on your project
	Global	installed on your system
2	Local	Required in project code
	Global	Specific import path.
3	Local	Managed by package.json.
	Global	Managed by global NPM registry.
4	Local	Auto-updated
	Global	Manually updated.

Q3

What is the difference between PNM local and global packages ?

1
Local installed on your project
Global installed on your system

2
Local Required in project code
Global Specific import path.

3
Local Managed by package.json.
Global Managed by global NPM registry.

4
Local Auto-updated
Global Manually updated.