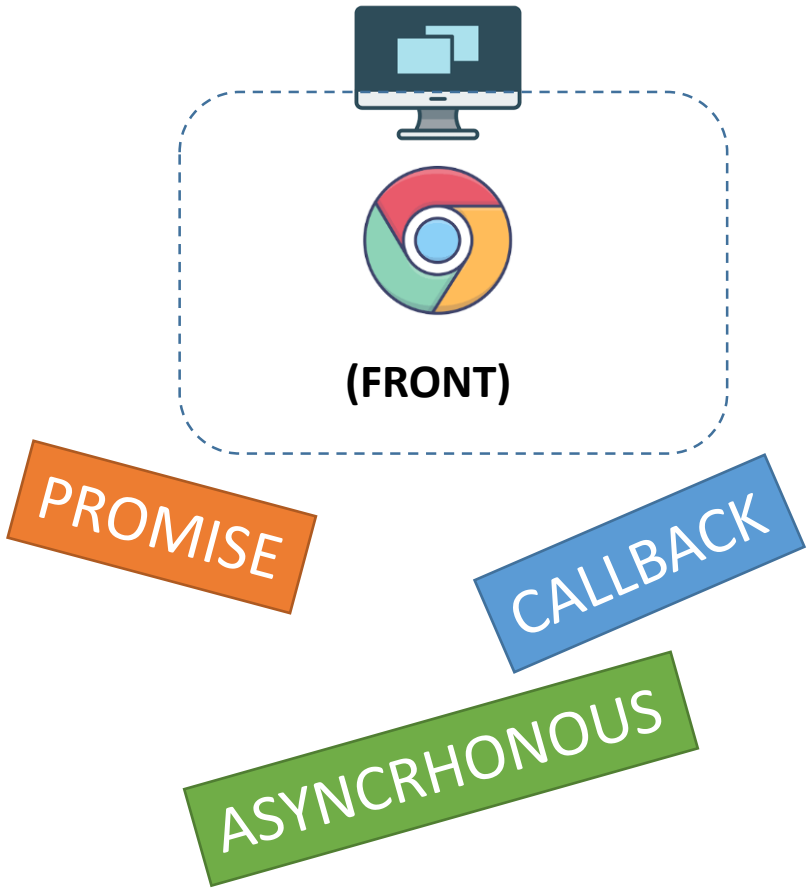
JS

NODE

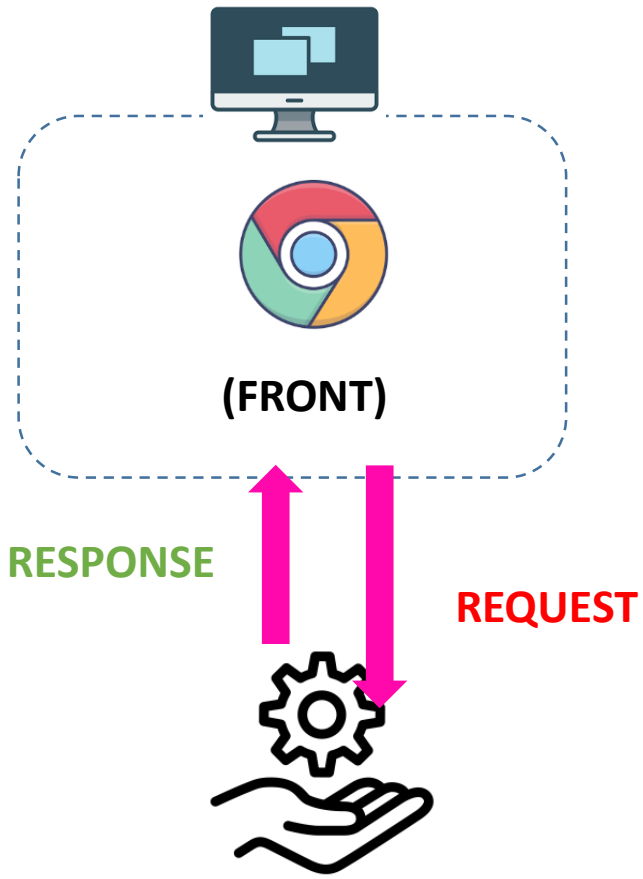
Introduction - Packages

What we have learnt last week ?

How JS can wait
For the requested data



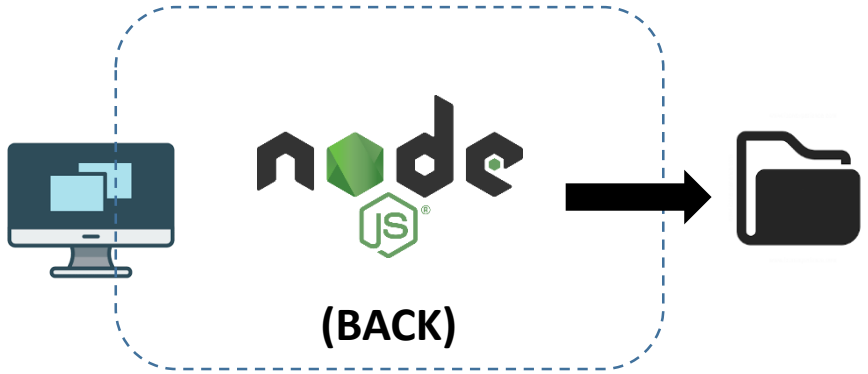
Request (GET) to a web service using HTTP



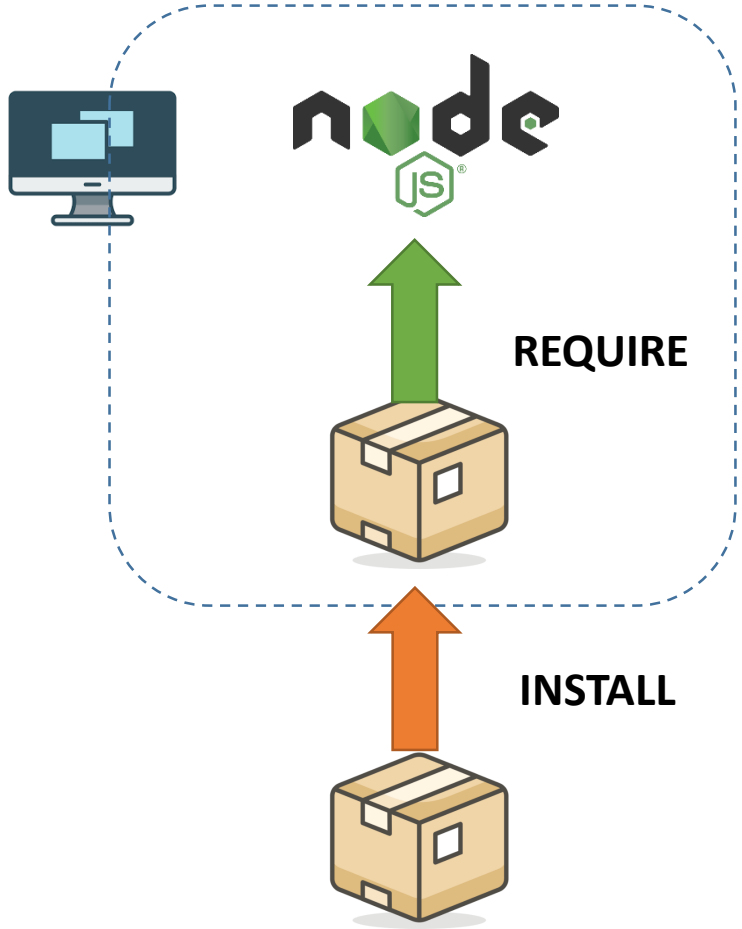
What we are learning this week ?

Run a back with node

Read/Write files

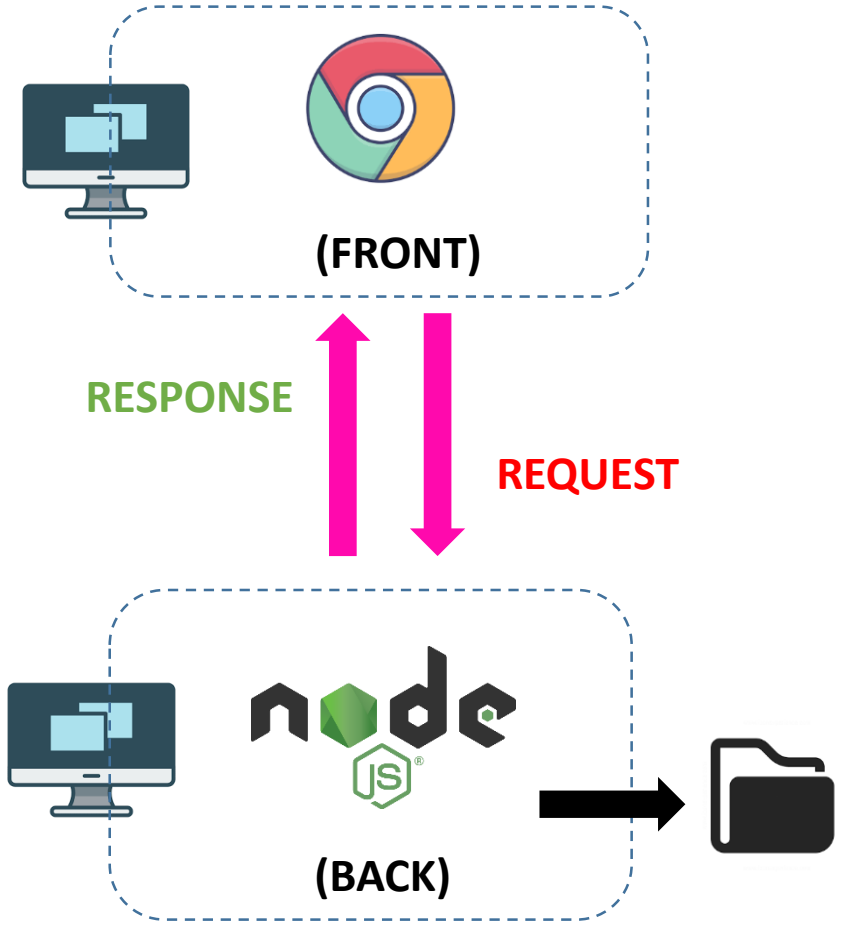


Install and request modules



What we will learn next week ?

Commutate between your FRONT and your BACK



OBJECTIVES FOR TODAY

- ✓ What is **NODE** ?
- ✓ What is **a MODULE** ?
  - ✓ **EXPORT** and **REQUIRE**
- ✓ What is **a BUILD-IN MODULE** ?

EXPLORE

ACTIVITY 1 10 MIN

My first node project !

0 – Check **node** is installed : “node –v”

1 – Create a new **javascript file**, index.js

2 – Write the following line :

```
console.log('Hello Node.js!')
```

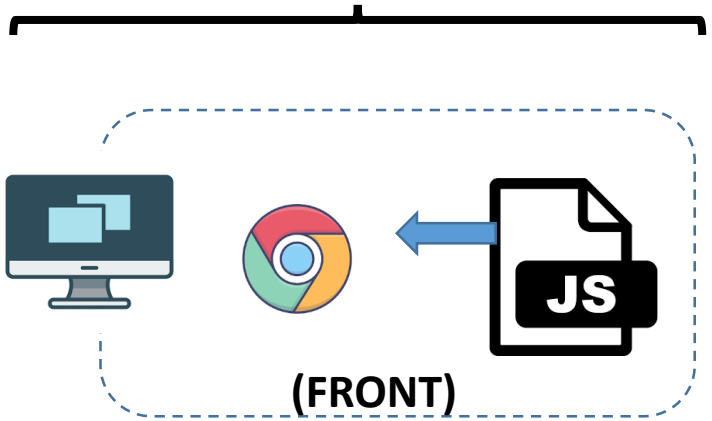
3 – Open the **terminal** and type : **“node index.js”**

What do you see ?
What can you conclude about Node ?

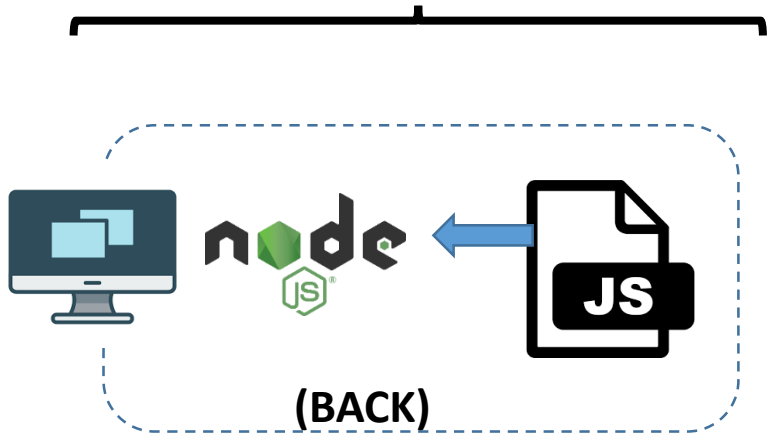
```
node <YOU FILE JS>
```

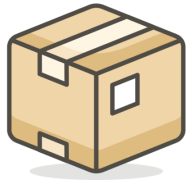
Run your JS using Node

On the FRONT
JS files are executed on **browser**



ON the BACK
JS files are executed on **Node**

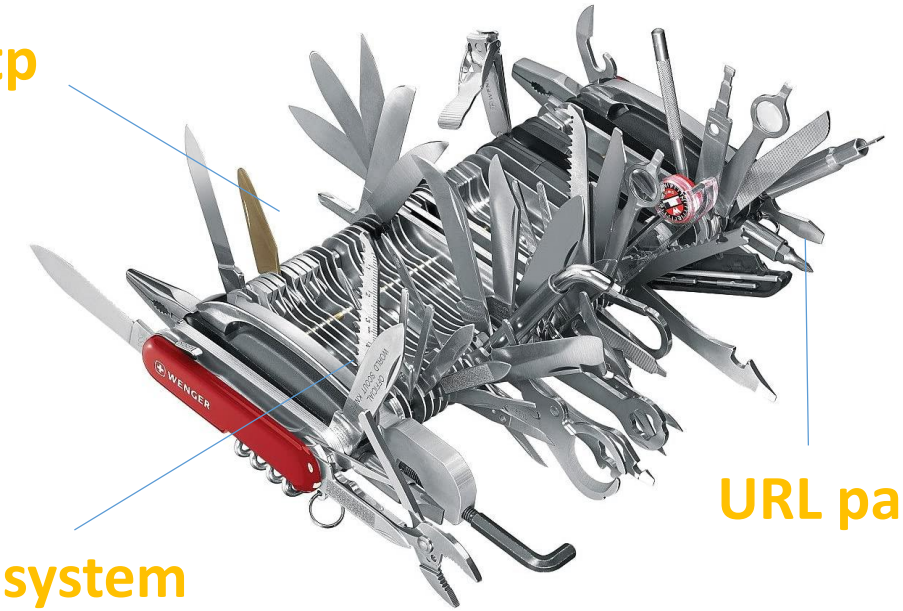




MODULES are **functions** to add to your code

JS

JS + MODULES



You cannot “see” functions / variables of another file

operations.js

```
function addNumber(a, b) {
    return a +b;
}
```

test.js

```
console.log( addNumber(4, 5));
```

What is the result on the console ?

Let put a value in the module export

operations.js

```
module.export.dayInWeek = 7
```

test.js

```
const myModule = require("./myModule.js");
console.log(myModule);
console.log(myModule.dayInWeek);
```

What do you see on the first line ? On the second line ?

Let put a function in the module export

operations.js

```
function addNumber(a, b) {
    return a +b;
}
module.export.myAdd = addNumber(a
```

test.js

```
const myModule = require("./myModule.js");
console.log(myModule);
console.log(myModule.myAdd(2, 3);
```

What do you see on the first line ? On the second line ?

To use a module data or function: export it, require it

library.js

```
let a= 2
let b= 2
let c= 2

module.export.a =a;
module.export.b =b;
```

main.js

```
let lib = require("./library.js");
```

lib.a ← AVAILABLE

lib.b ← AVAILABLE

lib.c ← NOT AVAILABLE

EXPLORE

ACTIVITY 2 15 MIN

Create a module and use it in a main file

library.js

- write a function min(a, b) to compute the min of 2 numbers
- write a function max(a, b) to compute the max of 2 numbers

- create a constant PI = 3.14
- create a constant COVID = 19

- export : min, COVID (but not other ones)

main.js

require library.js

check you can use min, COVID
check you cannot use max, PI

NO MODULES

ALL IN THE SAME PLACE

MODULES

SEPARATE THE JOBS

Benefits of modules

- ✓ **Less code lines** in one file
- ✓ Each file has **its own job**
- ✓ Easier to **understand**

What are build-in modules ?

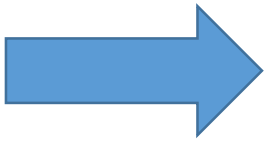
Node.js has a set of built-in modules which you can use **without any further installation**.

build-in

Read a file with FS

bob.txt

```
Toto
Tata
Tutu
```



```
let  fs = require("fs");

let content =
fs.readFileSync(“bob.txt").toString();


console.log(first);
```

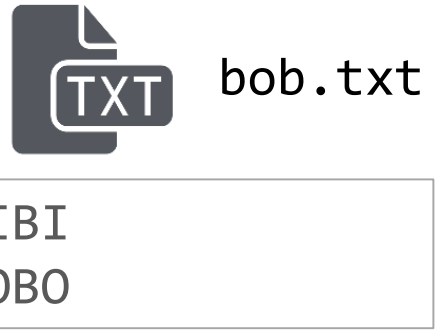
build-in

Write a file with FS

```
let  fs = require("fs");

let text = ‘BIBI\nBOBO’;

fs.writeFileSync(“bob.txt”, text)
```



EXPLORE

ACTIVITY 3

Read/Write a file with FS

See instructions

NOW YOU KNOW

- ✓ What is **NODE** ?
- ✓ What is **a MODULE** ?
  - ✓ **EXPORT** and **REQUIRE**
- ✓ What is **a BUILD-IN MODULE** ?

LINKS

Nodejs documentation :
https://nodejs.org/api/modules.html

NodeJS documentation module oriented :
https://nodejs.org/dist/latest-v11.x/docs/api/fs.html

Nodejs File System guide :
https://www.tutorialsteacher.com/nodejs/nodejs-file-system