# Lab3 – Node.js & NPM

## 1. What is Node?

Node.js is a JavaScript runtime environment built on the **Chrome V8 JavaScript engine**. Node has a package ecosystem called **NPM (Node Package Manager)**, which is the largest open-source JavaScript library. When writing JS applications, it is often common to use packages from this library, which allows you to benefit from existing JS code without having to "*reinvent the wheel.*"

## 2. Why is Node important?

- Node is very popular, especially in recent years. Every web developer should know it.
- Node is based on JavaScript.
- Node is very useful for web development on the server side.
- Thanks to NPM, Node has a very large open-source library. It is more performant than older web development languages on the server side, such as PHP.

## 3. Installing and using Node

Install node.js on your machine (go to nodejs.org and download node). To make sure it has been installed correctly, type "`node -v`" at the command prompt and you will get the version of node that has been installed.

At the command prompt, type "`node`" to be in the node command prompt. From here you can execute JS code. This code is supposed to be in a server program and therefore the JS functions used in the browser are no longer recognized (alert, prompt, document. getElementBy**, DOM...).

To execute JS code from the command line, you need to:

- Create a file called "`f.js`" in VSCode (or write it directly with "`echo >> f.js`" in Windows).
- At the command prompt, type "`node f.js`" and press Enter. The JS code in the file "f" will execute.

## 4. Node.JS module system

The Node module system allows you to load and use external JavaScript libraries. These libraries can be:

- Built-in modules specific to Node.
- Your own modules.
- Modules developed by third parties as npm modules. These libraries can be used to connect to a database, create web servers, develop different components in a web application, etc.

### 4.1. Importing Core Modules from Node

Node.js has several built-in modules known as core modules (see the list of these modules on https://www.w3schools.com/nodejs/ref_modules.asp). These are modules that already exist in Node, so you do not need to install them. They allow you to access tools for:

- Manipulating the file system
- Making HTTP requests
- Creating web servers, etc.

To import any module, you use the `require()` function, which loads a module and gives you access to its entire contents.

*Example:*

```
var myvar = require('fs');
myvar.writeFileSync('notes.txt', 'I live in Algeria');
```

### 4.2. Importing User Modules

To load JavaScript modules written by the user, you always use the `require()` function with the path to the JavaScript module you created and want to import as a parameter. This path must start with `./`.

*Example:* `app.js`

```
const checkTools = require('./src/utils');
checkTools();
```

For a user module to be importable, you need to add an instruction at the end of its script that specifies the parts that it can export to other modules. For the previous example, the module `utils.js` contains the following:

```
const check = function () {
  console.log('Doing some hard work...');
};
module.exports = check;
```

When you run the script `app.js`, you will see the following output in the console:

```
$ node app.js
Doing some hardwork...
```

### 4.3. Importing NPM Modules

When you install Node.js, you also implicitly install NPM. NPM is a package manager that allows you to install and use npm JS libraries (or packages) in your code. This opens the door to countless possibilities by using the npm ecosystem.

**Initializing NPM:**

A Node.js application requires that you initialize NPM before using it by running the following command from the root of your project:

```
$ npm init
```

This command will create a file called `package.json` in the root of your project.

**Installing an NPM Module:**

Once you have run the `npm init` command, you can install any module by typing the following:

```
npm install packName
```

Effects of installing an NPM module:

- Creates a folder called `node_modules` that npm will use to store the code for all installed npm modules.
- Adds the module `packName` as a dependency by listing it in the `dependencies` property in the `package.json` file. This will allow you to track and manage the modules you have installed.
- Creates a file called `package-lock.json` that contains detailed information about the installed modules.
- You should never modify the `node_modules` folder or the `package-lock.json` file because they are automatically managed by npm and are modified by running npm commands on the console.

## 5. Installing NPM packages

As an example of a package we will install, we will install the `cat-me` package:

1. On the command prompt, type:

```
npm install cat-me
```

*(This requires an internet connection.)*

2. Create a file called `app.js` that uses the `cat-me` package. (Return to the `cat-me` page on the npm website to see how to use this package.)
3. Execute `app.js`.

## Exercise 1:

Create a file called `echo.js` containing an `exf` function that accepts 2 arguments: a string `s` and an integer `n`, and prints the string s, n times, to the console.

Make the following 2 calls to this function in your `echo.js` file:

```
exf("echo", 5) ;
exf("JS from server", 10) ;
```

Save the file and run it on the node command console.

## Exercise 2:

Create a file called `notation.js` containing a function called `mean()` with 1 argument: an array of scores. and gives the average of these scores as the result.

1. Save the file and run it through node.
2. Create a `fileImport.js` file that imports the `notation.js` file and uses these functions. (complete the `notation.js` code for this purpose)

## Exercise 3:

Write an `ReadFile.js` script that reads a file (passed as a parameter) on the local machine and displays its contents on the console.
If a file f.txt contains, for example:
`"Hello, this is a text example".`
Execution of:

```
>node app.js test.txt
```

will display :
`"Hello, this is a text example".`
**Note:** *you'll need node's "`fs`" module*

## Exercise 4:

1. Write an `exo4.js` program that creates a file (.txt) and adds to it the text passed as a parameter.

*Example of execution :*

```
> node exo4.js "Coding for fun, that's my motto!"
The file has been saved!
> cat f.txt
Coding for fun, that's my motto!
```

**Note:** *you'll need node's "`fs`" module*

2. Modify the previous question to use the first parameter passed as the name of the destination file.
3. Add to question 2 the display on the console of the contents of the file created.

*Example of execution:*

```
> node exo4.js destination.txt "text to be saved to destination
file... "
The file has been saved!
> cat destination.txt
text to be saved to destination file...
```

**Note:** *you'll need node's "`fs`" module*

- **Deadline:**

At the end of Lab session (no later than Saturday, November 2 at 23:59)
Via: - Classroom.
- Commit to CAW_Labs repository.

“CAW” Course materials:

SCAN ME