# EPYTODO BOOTSTRAP

DISCOVER THE WORLD OF NODE.JS AND EXPRESS

# EPYTODO BOOTSTRAP

**language:** Node.js

## Introduction and installation of node and npm

Have you heard of Javascript ? Maybe you never heard of it or, like most people, you think that Javascript is only used to make a web page more interactive and that it is only used in front-end development.

Well, not anymore. Thanks to node.js, we can now use Javascript to build web servers, mobile apps, desktop apps, web apps, and much more.

Soooo, what is node.js? Long story short, node is simply a javascript runtime environment based on Chrome’s V8 engine. To learn more about it, you should take a look at this article and read more great resources on the subject.

Now that you read about Node, you most probably read “npm” somewhere. Just like node, we will provide you with a short definition, but deeper research on the subject will help you understand and follow this bootstrap with more ease.

Soooo, what is npm? Long story short, npm stands for “node package manager”. You guessed it, npm is the tool we use to start a node project and install dependencies (or package) for our node app and many other things you should read about.

## Starting a project

To start a project, move to your working directory and init a project by using npm (obviously, this means that you installed node + npm beforehand).

```
~/B-WEB-200> npm init
```

Here, you will be asked to answer some questions about your project, answer accordingly. In most cases, default answers are the right ones.
If done correctly, you should that a new file has been generated, the package.json.

Take a lock at the file, what do you think is its purpose?

package.json

## Follow recommended guidelines before you get lost

To help you build a good project, we will ask you to respect some rules that will help you maintain a clean architecture and introduce you to the recommended guidelines.

That's why we will ask you to use a .env file for your configuration variables (such as the port of your app, your database's name and password, etc...)

Have you heared of Dotenv ?

## Setting up express + First route

For this project, we will be using Express, a fast, unopinionated, minimalist web framework for Node.js.

Express is the most used framework out there to build node apps. It perfectly suits our needs for this project.

Again this module is introducing you to another language and another way of thinking and applying new ideas. Do not hesitate to take a break and do some research on the subject or ask the assistants if you have a hard time following along.

Let's install express, shall we?

What better than a Getting started to get things started? Getting started

If you followed the instructions correctly, you should end up with this:

```
Terminal
~/B-WEB-200> cat index.js
const express = require("express");
const app = express();
const port = 3000; // this should be in the .env file /!\

app.get("/", (req, res) => {
  res.send("Hello World!");
});

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`);
});
```

You can now run this app.

```
Terminal
~/B-WEB-200> node index.js
```

The app starts a server and listens on port 3000 for connections. The app responds with "Hello World!" for GET requests to the root URL (/) or route. For every other path, it will respond with a 404 Not Found.

You should now be asking yourself, what is a GET request ? Check This out.

In another terminal, you can use **curl** to reach your route. You should receive "Hello World!" as response, as it is the only route created.

```
Terminal
~/B-WEB-200> curl 127.0.0.1:3000
Hello World!
```

You can also test your routes with Postman or insomnia

## Exploring Express.js, Greetings + name

Now that we have our first route created, try to create another one, but this time create a route that takes your name as params and return "Hello name !" where `name` corresponds to the param sent to with the route.

It should be working as such:

```
Terminal
~/B-WEB-200> curl 127.0.0.1:3000/name/Thomas
Hello Thomas !
```

Now that you got that, try to build another route. But this time, return the current date as a response.

It should be working as such:

```
Terminal
~/B-WEB-200> curl 127.0.0.1:3000/date
2021-03-07
```

## Content type

So far, you created routes that respond with text only, but we'll no doubt need a better way to structure and pass along information, maybe some JSON or HTML ?. Let's see how we'd do that.

To do so, edit the routes you have previously created to respond according to the corresponding content type.

for example, the name route should be working as such:

Content-Type: Text

```
~/B-WEB-200> curl 127.0.0.1:3000/name/Thomas
Hello Thomas !
```

Content-Type: Json

```
~/B-WEB-200> curl 127.0.0.1:3000/name/Thomas
{
  "msg": "Hello Thomas !"
}
```

Content-Type: HTML

```
~/B-WEB-200> curl 127.0.0.1:3000/name/Thomas
<p>Hello Thomas !</p>
```

## Working with a database

This module will also introduce you to databases. In this part of the project, we will mainly focus on working with SQL databases using Node. In order to do this part of the project, you will need to install a SQL database.

To install your MySQL server, refer to the official documentation

To configure your MySQL server, refer to the official documentation

Query examples can be found here

Now, you can either create a simple database with a few tables and rows (name them as you wish) or you can use one that you already used for another project.

Now that you have a database up and running, we will try to interact with it using our node app. There are few ways to do this, but we will be using the package called “mysql2” to connect to our database.

Time to build some real routes:

Create a GET route that gets all the info in a table, this should return in an array all the records of a table.

Now, it will be more fun if we could actually insert a record inside a table, to do so, you will first need to send some info to your app.

POST request

But wait, how do we receive data in Express.js?

Have you heard of a Body parser ?

v 3