# Introduction to NodeJS : Using Environment Variables

## Learning Outcomes

At the end of this laboratory, you should be able to

- Know when to use Environment Variables
- Know how to exploit the *process* object
- Pass Environment Variables to your code in Command Line
- Pass Environment Variables to your code with the dotenv package.

## Lab Material and first assignment

You will start your development stage with the 'Task' web app given to you as correction of Lab #2. This code implements a Task model (Service + DAO) with lowDB.

**Your first assignment** is to implement the same Task model (the service must have the same interface) with a simple Array or Map, as seen in Lab #1. WARNING, the service's methods and properties must have the same name so that switching from one implementation to the other is as simple as comment/uncomment the following lines in index.js:

```
const TaskService = require("./model/TaskService_ArrayImpl.js");
//const TaskService = require("./model/TaskService_LowDbImpl.js");
```

## The Process object and second assignment

**The *process* object** in Node.js is a global object that can be accessed inside any module without requiring it. It is an essential component in the Node.js ecosystem as it provides various information sets about the runtime of the program (Remember that Node is single thread). As such it is **a bridge between the Node process and the operating system**. See https://nodejs.org/api/process.html for complete overview. Here are some of those infos (and see code sample 'envDemo.js' attached, it will usefull for the next assignment) :

- events :
  - *exit* is emited when the process is about to exit. Node normally exits with a 0 status code when no more async operations are pending. There are other exit codes.
  - *beforeExit* is emited when node empties its event loop and has nothing else to schedule.
  - *UncaughtException* is emitted when an exception bubbles all the way back to the event loop.
  - *Signal Events* is emitted when the process receives a signal such as SIGINT, SIGHUP, etc.(from the system)
  - ...
- properties :
  - *stdout* is a writable Stream to stdout (used by console.log, ...)
  - *stderr* is a writable Stream to stderr (used by stderr,...)
  - *argv* is an array containing the command line arguments.
  - ***env* is an object containing the user environment.**
  - *pid* is the PID of the process.
  - ...
- methods :
  - *nextTick(callback)* Once the current event loop turn runs to completion, call the callback function.
  - ...

**Your second assignment** is to modify the index.js file in order to retrieve from the environment the values of Port and Host variables used in :

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

If the sought environment variables are not set, default values should be given (localhost/3001).

## The package.json scripts and third assignment

See https://docs.npmjs.com/cli/v7/using-npm/scripts for an overall explanation.

The “scripts” field in the package.json file allow you to store user-defined commands to execute with npm.

```
npm run mySetOfCommand)
```

Those commands are Command Line commands

```
“scripts”:{
      “mySetOfCommands”: "echo \"Here is my special command\" && exit 0"
}
```

**Your third assignment** is to modify the package.json and index.js files to launch your app whether with the lowdb implementation or the array implementation . The package.json file will contain two scripts : *arrayImpl* and *lowdbImpl* that will launch your webApp with an environment variable TS_IMPL set . The index.js file will contain a mechanism to read TS_IMPL and switch from one implementation to the other (see first assignement).

## The dotenv package and fourth assignement

See https://www.npmjs.com/package/dotenv for an overview.

Dotenv is a zero-dependency module that loads environment variables from a `.env` file into `process.env`. Storing configuration in the environment (database access token, passwords, keys, ...) separate from code is a good practice.

**Your fourth assignment** is to access the path of the *db.json* file from *TaskService_LowDbImpl.js* with dotenv. You wan’t to transform the following line.

```
const adapter = new FileAsync("./model/db.json");
```

and pass the path given in a .env file to the FileAsync method.