# Server-Side Web Development - Exercise 02

Student: Markus Ijäs

10.02.2022

Help to complete the tasks of this exercise can be found from the chapter 3 "Creating a Node.js Module" of our course book "Get Programming with Node.js" by Jonathan Wexler and from the chapter "Exploring Node.js Modules" of the supplementary course book "Node.js Web Development" by David Herron. The aims of this exercise are 1) to learn to understand and to use Node.js modules and 2) to give the skills needed to work with NPM package manager.

Embed your theory answers, drawings, codes, and screenshots directly into this document. Always immediately after the relevant question. Return the document in itsLearning by the deadline.

It's also recommendable to use Internet sources to supplement the information provided by the course book.

The maximum number of points you can earn from this exercise is 10.

*Any answers without references mentioned are usually from one of the course books, of just from my head*

## Tasks:

**1. Modules in Node.js. (4 * 0,5 = 2 points)**

**a. What are the benefits of using modules?**

- Development can be more easily divided (between developers, in time, etc)
- It makes programs more readable
- Errors are easier to detect and fix
- Allows easier re-use of code
- Improves manageability, maintainability and collaboration.

*Source: http://gwentechembedded.com*

**b. What is a module in Node.js?**

A basic building block for constructing Node.js applications.

**c. What are the module formats that can be used with Node.js?**

- Traditional CommonJS standard modules
- ES6 modules.
  - The ES6 modules are the new standard way of implementing Node.js modules, as per Node.js Technical Steering Committee.

**d. What are the main differences between these module formats?**

Maybe I'll just add a link here and don't expect to get a point from this task, as LogRocket's blog already has an excellent post on this subject: see blog.LogRocket.com.

Just to recap the main points for easier identification of module type:

- ES Modules depend on the `import` and `export` statements, and are official part of JavaScript.
- CommonJS modules are loaded using `require()` and attributes are exported with `module.exports`.

**2. Understanding and using NPM. (4 * 0,5 + 1 = 3 points)**

*Use books and visit the web site https://docs.npmjs.com/. Answer the questions below.*

**a. What is NPM? Explain the purpose and the three main components of it. (0.5 points)**

Node Package Manager, for managing (installing, updating and removing) JavaScript packages.

The main components are:

- the website for finding packages
- the Command Line Interface (CLI) for management
- the registry as the actual database of the packages.

**b. For what can npm be used for? (0.5 points)**

For managing packages, as stated above.

**c. Explain the purpose and the details of the (fake) package.json file below. (0.5 points)**

```
{
  "name": "test-project",
  "version": "1.0.0",
  "description": "An example project",
  "main": "src/main.js"
  "scripts": {
    "start": "npm run dev",
    "test": "npm run unit",
  },
  "dependencies": {
    "express": "^4.5.2"
  },
  "devDependencies": {
    "babel-core": "^6.22.1",
    "babel-loader": "^7.1.1"
  },
  "engines": {
    "node": ">= 6.0.0",
    "npm": ">= 3.0.0"
  }
}
```

Most of it is self-explanatory, but here's explanations for some parts:

- **scripts**: The cli-runnable scripts (basically determines how to run and test the package)
- **dependencies**: The production dependencies, with package `express` on minor release up to 4.5.2 in this case.
- **devDependencies**: Dependencies for development.
- **engines**: The engines and their minimum versions.

**d. Write npm CLI commands to (1 point)**

**a. initialize a Node.js project directory for your application so that package.json file is also created**

**b. install a package as a dependency to your application**

**c. install a package as a development dependency to your application**

**d. install a certain version of a package as a development dependency to your application**

**e. try to update one of the installed packages**

**f. remove one of the installed packages**

**g. modify the package.json so that you can start your application by writing npm start**

**h. install a package globally on your computer**

**i. download a node.js application and install all its dependencies with one npm command**

**e. Explain shortly. (0.5 points)**

**a. What does the folder node_modules contain**

**b. What does the global installation mean?**

**c. Should the contents of the folder node_modules be uploaded into a version control repository and why?**

Yes. That way you'll always have the working dependencies to install, even if some of the depended packages (or the required version) cease to exist.

**3. Program a Node.js module with CommonJS format (2 points)**

Program a Node.js module "evaluator", which can be used to give a grade for a student by using a custom evaluation scale.

The module exports two functions. The first function's signature is setEvaluationScale(scale). The argument scale is of a type array. The example below clarifies the structure of the scale argument.

```
[
  {grade: 1, points: 20},
  {grade: 2, points: 35},
  {grade: 3, points: 50},
  {grade: 4, points: 65},
  {grade: 5, points: 80}
]
```

The feature points is the minimum number of points that the student needs to collect to get the grade given in the feature grade. In case all the points are more than the student has collected, then the student gets the grade 0.

The second function's signature is getGrade(points). The argument points is the number of points a student has collected. The functions returns the grade for the given points.

Please note that if someone tries to call function getGrade before the evaluation scale is set, then a message 'There is no evaluation scale defined.' is returned.

Require the module and test that it functions as expected by calling its methods.

**Answers**

**evaluator.js:**

```
let evaluationScale = null;

exports.setEvaluationScale = function (scale) {
  if (Array.isArray(scale)) {
    evaluationScale = scale;
  }
};
```

```
exports.getGrade = function (points) {
  if (!Array.isArray(evaluationScale)) {
    return "There is no evaluation scale defined.";
  }

  grade = 0;

  evaluationScale.forEach((element) => {
    if (points >= element.points && grade < element.grade) {
      grade = element.grade;
    }
  });

  return grade;
};
```

**index.js:**

```
const evaluator = require("./evaluator");

let scale = [
  { grade: 1, points: 20 },
  { grade: 2, points: 35 },
  { grade: 3, points: 50 },
  { grade: 4, points: 65 },
  { grade: 5, points: 80 },
];

console.log("Points 50, no scale set. Got grade: " + evaluator.getGrade(50));

evaluator.setEvaluationScale(scale);

console.log("Points 50, scale set. Got grade: " + evaluator.getGrade(50));
console.log("Points 101, scale set. Got grade: " + evaluator.getGrade(101));
console.log("Points 19, scale set. Got grade: " + evaluator.getGrade(19));
```

**package.json:**

```
{
  "name": "t4",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Markus Ijäs",
  "license": "MIT"
}
```

**And simple run with `npm start`:**

```
$  npm start

> t3@1.0.0 start
> node index.js

Points 50, no scale set. Got grade: There is no evaluation scale defined.
```

```
Points 50, scale set. Got grade: 3
Points 101, scale set. Got grade: 5
Points 19, scale set. Got grade: 0
```

### 4. Program a Node.js module with ES6 format (1 point)

*Program the previous module "evaluator" again. This time, implement it as a ES6 module. Also the module that imports it should be an ES6 module. Test that everything works.*

**evaluator.js:**

```
let evaluationScale = null;

function setEvaluationScale(scale) {
  if (Array.isArray(scale)) {
    evaluationScale = scale;
  }
}

function getGrade(points) {
  if (!Array.isArray(evaluationScale)) {
    return "There is no evaluation scale defined.";
  }

  grade = 0;

  evaluationScale.forEach((element) => {
    if (points >= element.points && grade < element.grade) {
      grade = element.grade;
    }
  });

  return grade;
}

export { setEvaluationScale, getGrade };
```

**index.js:**

```
import { setEvaluationScale, getGrade } from "./evaluator.js";

let scale = [
  { grade: 1, points: 20 },
  { grade: 2, points: 35 },
  { grade: 3, points: 50 },
  { grade: 4, points: 65 },
  { grade: 5, points: 80 },
];

console.log("Points 50, no scale set. Got grade: " + evaluator.getGrade(50));

evaluator.setEvaluationScale(scale);

console.log("Points 50, scale set. Got grade: " + evaluator.getGrade(50));
console.log("Points 101, scale set. Got grade: " + evaluator.getGrade(101));
console.log("Points 19, scale set. Got grade: " + evaluator.getGrade(19));
```

**package.json:**

```
{
```

```
  "name": "t4",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "start": "node index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Markus Ijäs",
  "license": "MIT"
}
```

**And simple run with `npm start`:**

```
$  npm start

> t4@1.0.0 start
> node index.js

Points 50, no scale set. Got grade: There is no evaluation scale defined.
Points 50, scale set. Got grade: 3
Points 101, scale set. Got grade: 5
Points 19, scale set. Got grade: 0
```

**5. Module variables `**dirname` and `**filename`. (1 point)**

**a. Program a Node.js module with CommonJS format that prints the contents of these variables to the console.**

*This time with no npm packaging*

**a_module.js:**

```
exports.printVars = function () {
  console.log("Contents of __dirname: " + __dirname);
  console.log("Contents of __filename: " + __filename);
};
```

**a_index.js:**

```
const a_module = require("./a_module");

console.log("Printing vars from A module (CommonJS)");
a_module.printVars();
```

**Run:**

```
$  node a_index.js
Printing vars from A module (CommonJS)
Contents of __dirname: /home/markus/projects/turkuamk/sswd-22/e02/t5
Contents of __filename: /home/markus/projects/turkuamk/sswd-22/e02/t5/a_module.js
```

**b. Program a Node.js module with ES6 format that creates these variables and prints their contents of to the console.**

*This time with no npm packaging*

**b_module.mjs:**

```
import { fileURLToPath } from "url";
import { dirname } from "path";
```

```
/**
 * Solution from kindacode.com, thanks to A. Goodman.
 * www.kindacode.com/article/node-js-using-__dirname-and-__filename-with-es-modules/
 */
const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);

function printVars() {
  console.log("Contents of __dirname: " + __dirname);
  console.log("Contents of __filename: " + __filename);
}

export { printVars };
```

**b_index.mjs:**

```
import { printVars } from "./b_module.mjs";

console.log("Printing vars from B module (ES6)");
printVars();
```

**Run:**

```
$  node b_index.mjs
Printing vars from B module (ES6)
Contents of __dirname: /home/markus/projects/turkuamk/sswd-22/e02/t5
Contents of __filename: /home/markus/projects/turkuamk/sswd-22/e02/t5/b_module.mjs
```

### 6. Node.js core modules (4 * 0.25 = 1 point)

#### a. What are Node.js core modules?

*Visit the address https://nodejs.org/api/index.html. Answer the following questions.*

#### b. What is that site?

Node.js documentation.

#### c. Check the link About this documentation. What do you find there?

- Instructions on how to contribute to node.js.
- Stability index for documentation.
- Stability overview of different node.js modules.

#### d. Check the link Path. What do you find there?

Documentation for the Path module.