Node.js

Ch 3: Node Module System

D. HOSTENS & M. DIMA

hogeschool vives

— Wat is NPM?

Is a command-line tool

But also a

Registry for *third party libraries* which can be added to our projects
https://www.npmjs.com/

Libraries are free and open-source, ~~+/- 500K~~ ~~2.3M~~ 3.4M

— Registry

Surf to https://www.npmjs.com/

+/- ~~500K~~ ~~2.3M~~ 3.4M building blocks

You can create your own Node modules and push them to NPM

Via command-line tool

npm is installed together with Node

In the terminal:

```
~ ➤ npm -v
11.1.0
~ ➤ node -v
v22.13.1
~ ➤
```

— Upgrade or downgrade npm

```
milan@~ᗰ npm i -g npm@7.4.3
```

npm = the package manager

i = short for install (install also works)

-g = global (not locally in the app)

npm = name of the npm package you want to install

@versie = optional, the version

Similar syntax to install all packages (not only npm)

sudo possibly needed on Mac/linux

— package.json file

Keeps information about your appliction

When npm is used all Node apps have this file

```
milan@nodeVbᛗ mkdir npm-demo
milan@nodeVbᛗ cd npm-demo
milan@npm-demoᛗ npm init
This utility will walk you through creating a
package.json file.
…
milan@npm-demoᛗ ls
package.json
milan@npm-demoᛗ
```

`npm init --yes` (or -y if you want to skip the wizard)

— Add a 3rd Party library to your app

We will install `underscore` (popular js library)

https://www.npmjs.com/package/underscore

in the terminal (mind the path)!

```
milan@npm-demo〽 npm i underscore
added 1 package, and audited 2 packages in 2s
found 0 vulnerabilities
milan@npm-demo〽
```

Before: `npm i package --save` : now not needed anymore!

Check the dependencies under your package.json file

```
"dependencies": {
    "underscore": "^1.12.0"
  }
```

— Import a 3rd party library in your app

In VS Code, we create a new file: index.js

```
var _ = require('underscore');
```

On `require` packages will be searched in this sequence:

1. node module `require('library')` or `require('node:library')`
   file or directory `require('./library')`
2. node_modules `require('library')`

See http://underscorejs.org/ for more documentation

we use the `.contains()` operation on arrays

```
var result = _.contains([1,2,3], 2);
console.log(result);
```

```
milan@npm-demo~ node index.js
true
```

— var, let or const?

What to use?

- Since ES6 let and const added
- Var variables can be redeclared and changed. They are function-scoped

```
var greeter = "hey hi";
var greeter = "say Hello instead”;
greeter = "say Hello instead";
```

- Let variables can be changed but not redeclared. Are also block-scoped, inside {}

```
let greeting = "say Hi";
let greeting = "say Hello instead"; // error: Identifier 'greeting' has already been declared
```

- Const variables can not be changed nor redeclared. Also block-scoped, inside {}

```
const greeting = "say Hi”;
greeting = "say Hello instead";// error: Assignment to constant variable.
const greeting = "say Hello instead";// error: Identifier 'greeting' has already been declared
```

MongoDB Library - Lab

Install mongoose in a new application

What is the installed version?

Check the two package.json files (app and mongoose)

What do you see inside the node_modules folder?

— Dependencies and the node_modules directory

Dependencies: Some libraries also install their own dependencies.

Before: Dependencies were placed in the package folder.

Problem: Dependencies were installed multiple times in a nested structure.

Additional problem: Limited path length by Windows.

Solution: All dependencies are placed at the root level, unless a different version of a package is needed than an already installed dependency.

— NPM en Source control

Don’t commit node_modules! (often 100+ MB)

How? In .gitignore *exclude all unwanted files*

```
node_modules/
```

> You create the .gitignore file and add all files and directories you do NOT wish to commit to it.

Everything else will be *committed*

When you clone a new node project: `npm i` **of** `npm install`

Tip: with `git status` you can see what files will be committed

— Semantic versioning

Semantic versioning (SemVer)

E.g.: ^1.8.3

^ : caret sign or roof, sometimes ~ (tilde) means at least this version or highest minor version

1 : Major (breaking update)

8 : Minor (non-breaking update)

3 : Patch (small bug-fixes)

Sometimes noted as: 1.X instead of ^1.8.3

Without caret ^ or ~ : 1.8.3 = only this version

— Check version

in the package.json file in the root of the project

Or

With the following commands:

```
milan@npm-demo〽 npm list
npm-demo@1.0.0 /Users/milan/Dev/nodeVb/npm-demo
├── mongoose@5.11.17
└── underscore@1.12.0
```

```
milan@npm-demo〽 npm list --depth=1
npm-demo@1.0.0 /Users/milan/Dev/nodeVb/npm-demo
├─┬ mongoose@5.11.17
│ ├── @types/mongodb@3.6.7
│ ├── bson@1.1.5
..
```

— Dependencies

See npmjs.org or

```
milan@npm-demo ᨐ npm view mongoose
mongoose@5.11.17 | MIT | deps: 12 | versions: 641
Mongoose MongoDB ODM
https://mongoosejs.com
...
dependencies:
@types/mongodb: ^3.5.27                mquery: 3.2.4
bson: ^1.1.4                           ms: 2.1.2
...
```

— Dependencies cont.

See only the dependencies:

```
milan@npm-demo〽 npm view mongoose dependencies
{
  '@types/mongodb': '^3.5.27',
  bson: '^1.1.4',
  kareem: '2.3.2',
  mongodb: '3.6.4',
…
```

All *released* versions:

```
milan@npm-demo〽 npm view mongoose versions
[
  '0.0.1',      '0.0.2',      '0.0.3',
'0.0.4',      '0.0.5',      '0.0.6' ...
```

— Upgrade and downgrade npm package

Install a specific version (up- or downgrade)

```
milan@npm-demo〽 npm i mongoose@2.4.2
added 2 packages, removed 34 packages, changed 2
packages, and audited 6 packages in 3s
found 0 vulnerabilities
```

in package.json of the mongoose package we can now see:

```
    "name": "mongoose"
  , "description": "Mongoose MongoDB ODM"
  , "version": "2.4.2"
```

in package.json of the app:

```
    "mongoose": “^2.4.2",
```

— Lab

downgrade underscore to 1.4.0

— Versions

What new packages versions are available in the meantime?

```
milan@npm-demo〽 npm outdated
Package     Current  Wanted   Latest  Location                 Depended by
mongoose      2.4.2  2.9.10  5.11.17  node_modules/mongoose    npm-demo
underscore    1.4.0  1.12.0   1.12.0  node_modules/underscore  npm-demo
milan@npm-demo〽
```

`npm update` : only minor updates and patches!

```
milan@npm-demo〽 npm outdated
Package   Current  Wanted   Latest  Location                Depended by
mongoose   2.9.10  2.9.10  5.11.17  node_modules/mongoose   npm-demo
```

Tip: also see the package: `npm-check-updates`

— Dev dependencies

Until now: application dependencies

Sometimes also *development-only* dependencies needed

We install `jshint` (finds syntax errors inside your code)

```
milan@npm-demo〽 npm i jshint --save-dev
added 31 packages, and audited 37 packages in 4s
...
milan@npm-demo〽 jshint index.js
```

in package.json

```
  "devDependencies": {
    "jshint": "^2.12.0"
```

— Uninstall package

`npm un package` or `npm uninstall package`

```
milan@npm-demo〽 npm un mongoose
removed 4 packages, and audited 33 packages in 2s
...
milan@npm-demo〽
```

Gets deleted in package.json and from the node_modules folder

— Global packages

Some packages are not app specific

Usually commandline-tools such as npm

`npm i -g npm` (update to latest version op npm)

```
milan@npm-demo〽 npm -g outdated
Package  Current  Wanted  Latest  Location             Depended by
npm        7.4.3   7.5.4   7.5.4  node_modules/npm     global
milan@npm-demo〽
```

uninstall global package: `npm un -g package`

— Publish package

We create a new project: `mkdir vives-lib` + `npm init`

We create a new file index.js

```
module.exports.add = function(a, b) { return a + b };
```

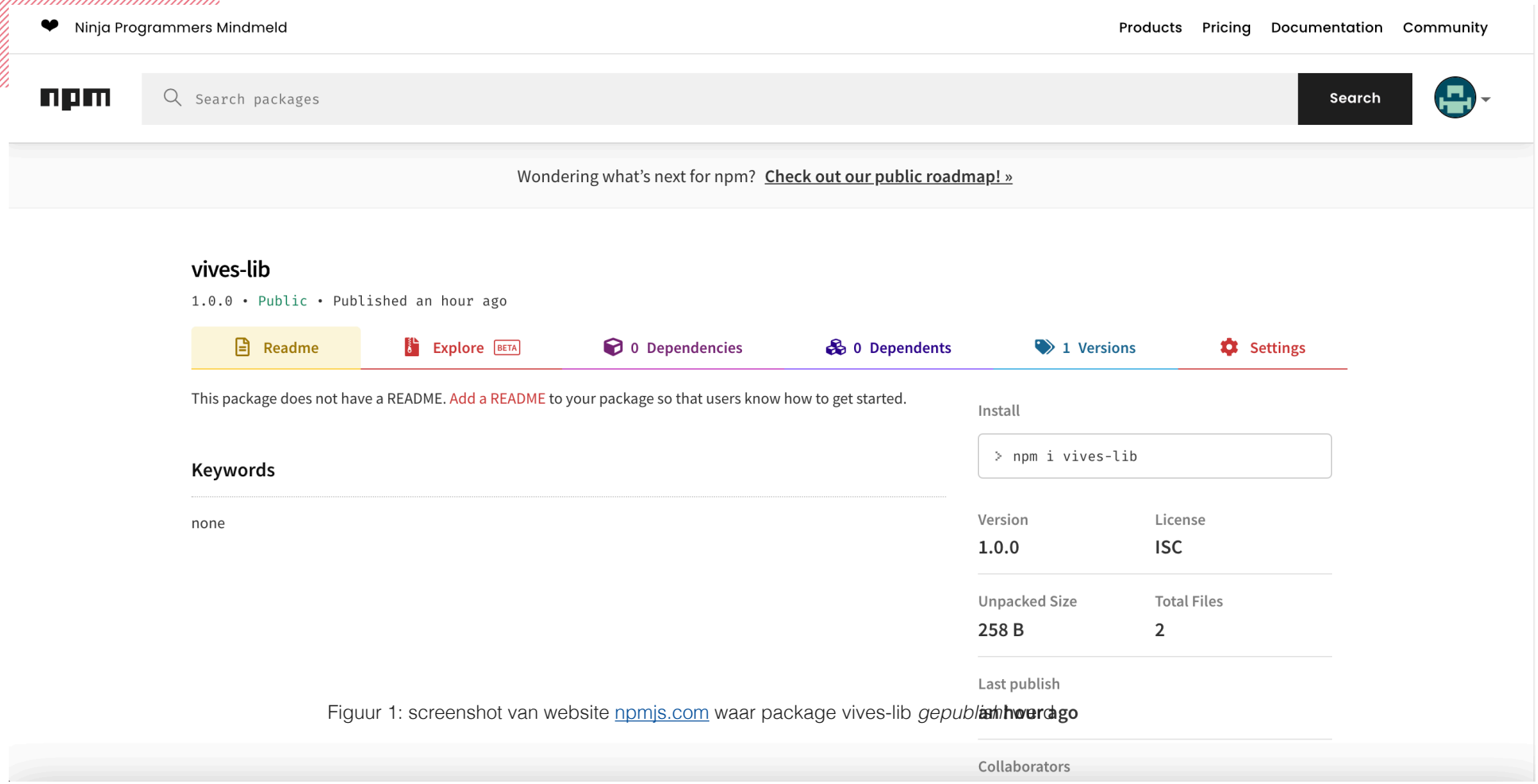
We log in `npm login`

If no account yet: `npm adduser` or register on npmjs.com

```
milan@vives-lib〽 npm login
npm notice Log in on https://registry.npmjs.org/
Username: vives
Password:
Email: (this IS public) milan.dima@vives.be
Logged in as vives on https://registry.npmjs.org/.
milan@vives-lib〽 npm publish
npm notice
npm notice 📦  vives-lib@1.0.0
...
```

Note that the package name should be unique! Change package name in [package.json]

https://www.npmjs.com/package/vives-lib



Figuur 1: screenshot van website npmjs.com waar package vives-lib *gepubliceerd* is

— Lab

Create a new folder "adder"

install the vives-lib package

Import my vives-lib package inside your index.js file and use the add function

— Import package

```
milan@nodeVb〽 mkdir viveslib-vb
milan@nodeVb〽 cd viveslib-vb
milan@viveslib-vb〽 npm i vives-lib
added 1 package, and audited 2 packages in 3s
found 0 vulnerabilities
```

```
var vives = require('vives-lib');
var result = vives.add(1,2)
console.log(result);
```

```
milan@viveslib-vb〽 node index.js
3
milan@viveslib-vb〽
```

hogeschool vives

— Publish update

in our vives-lib index.js file we add an extra function:

```
module.exports.multiply = function(a, b) { return a * b };
```

`npm publish` throws an error

in package.json change version number to 1.1.0 **or**

npm version major/minor/patch

```
milan@vives-lib〽 npm version minor
1.1.0
milan@vives-lib〽 npm publish
npm notice
npm notice 📦  vives-lib@1.1.0
...
```

— Lab

See Github Classroom Les 3