back-end

Node & NPM

📸

Introductie

today

I. Course

II. Node

III. Modules

IV. NPM Package

Course

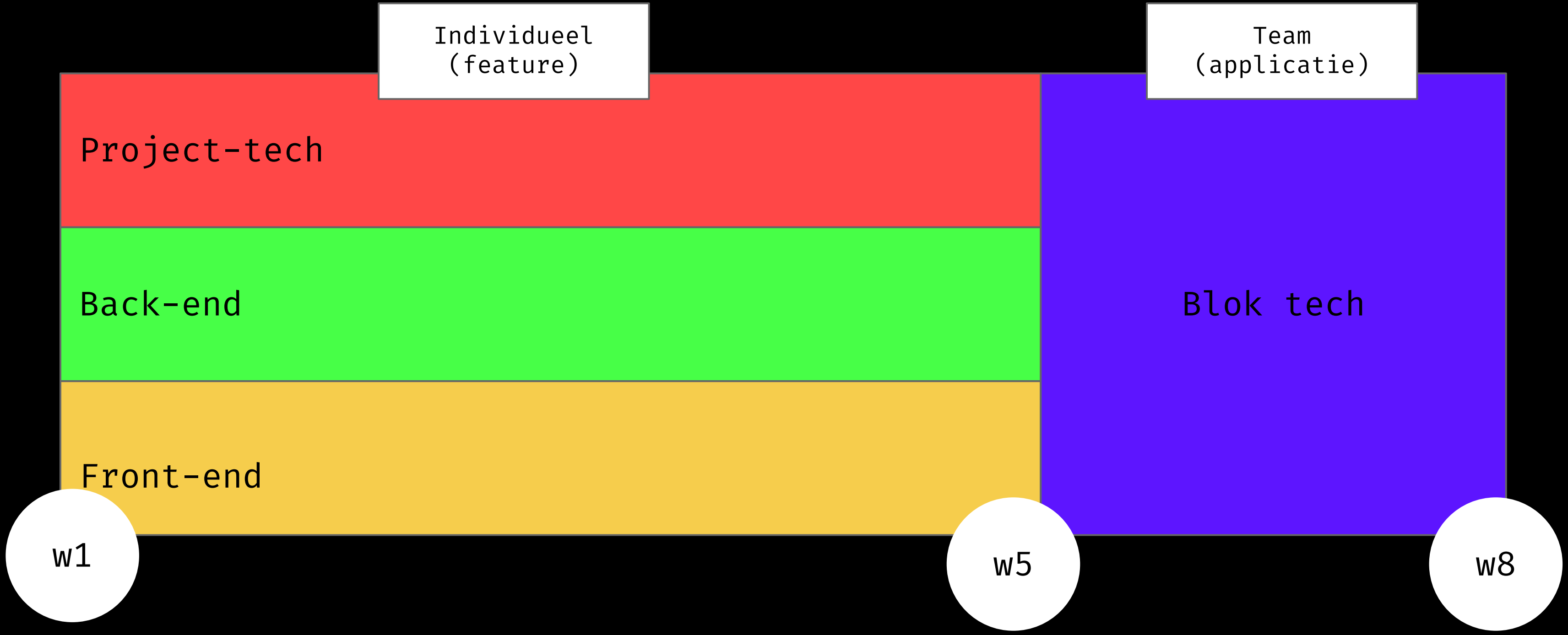
course description

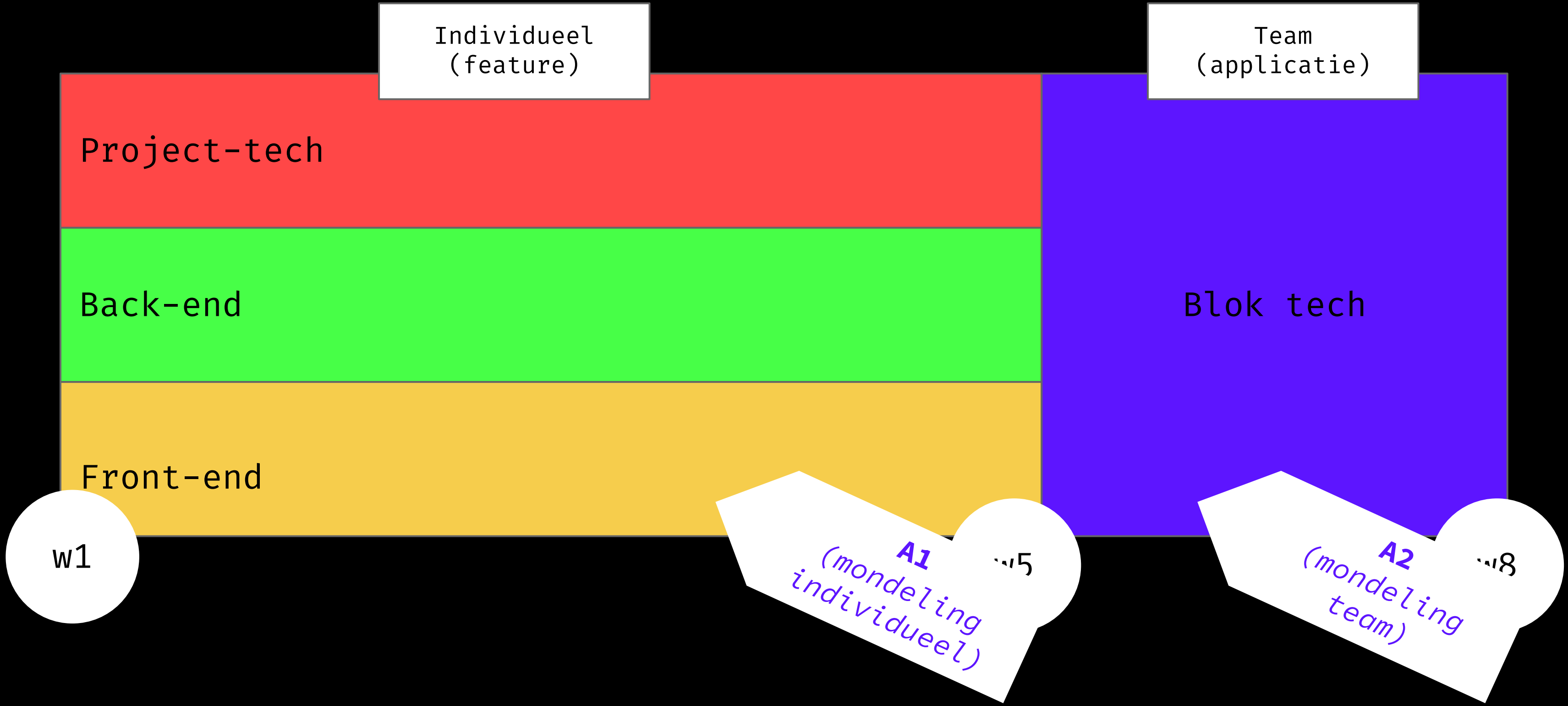
In Back-end we peek behind the curtains and inspects what’s behind the web. You build web app with Node.js, communicate with HTTP, and store data in a database with MongoDB. You’ll learn to use computers to actually make what you design work, people can actually fill in forms and upload files.

/readme.md

course goals

- You can build web apps with Node and NPM
- You understand client/server flow (http)
- You can render data server-side with a template engine
- You can store data in a database and update that data
- You can write docs and explain your code cohesion


Individueel
(feature)
Team
(applicatie)
Project-tech
Back-end
Front-end
Blok tech
w1
w5
w8


Individueel (feature)
Team (applicatie)
Project-tech
Back-end
Front-end
Blok tech
w1
w5
w8
A1 (mondeling individueel)
A2 (mondeling team)

course deliverables

- **Individual Prototype:** working interactive feature for matching application
- **Team Prototype:** working interactive matching application
- **Process book (wiki):** that provides insight into the weekly iterative process

Node.js

node

javascript?

JavaScript (JS) is a **lightweight interpreted or JIT-compiled** programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat.

developer.mozilla.org

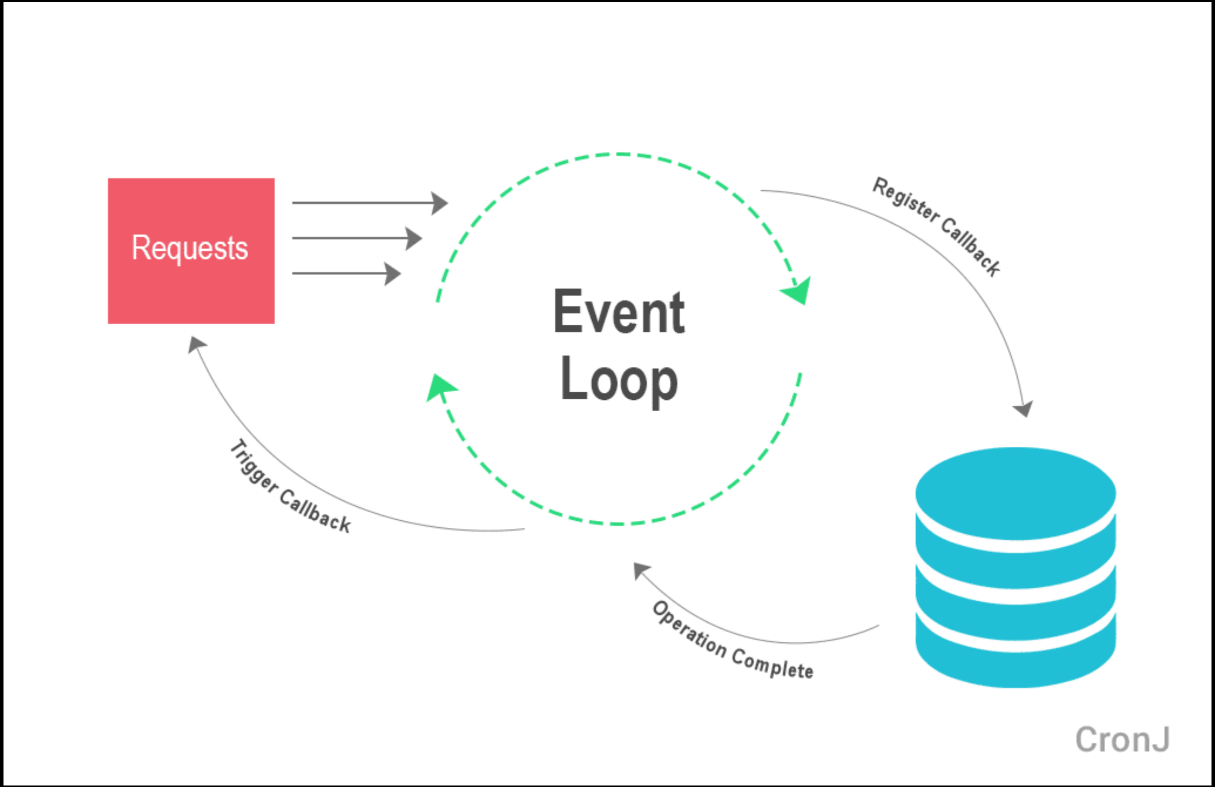
node

Node.js is an open-source, cross-platform […] run-time environment for **executing JavaScript code server-side.** […] Node enables JavaScript to be used for server-side scripting, and runs scripts server-side to produce dynamic web page content before the page is sent to the user’s web browser.

wikipedia.org

node

Non-blocking



Site à la Internetstandaarden

Client-side / Front-end
(browser)

Donut Kattenmand
27,95

1. Browser: ik wil de kattenmand bekijken. HTTP request:

   HTTP GET kattenmand.html

2. Server: hier is de pagina! HTTP response:

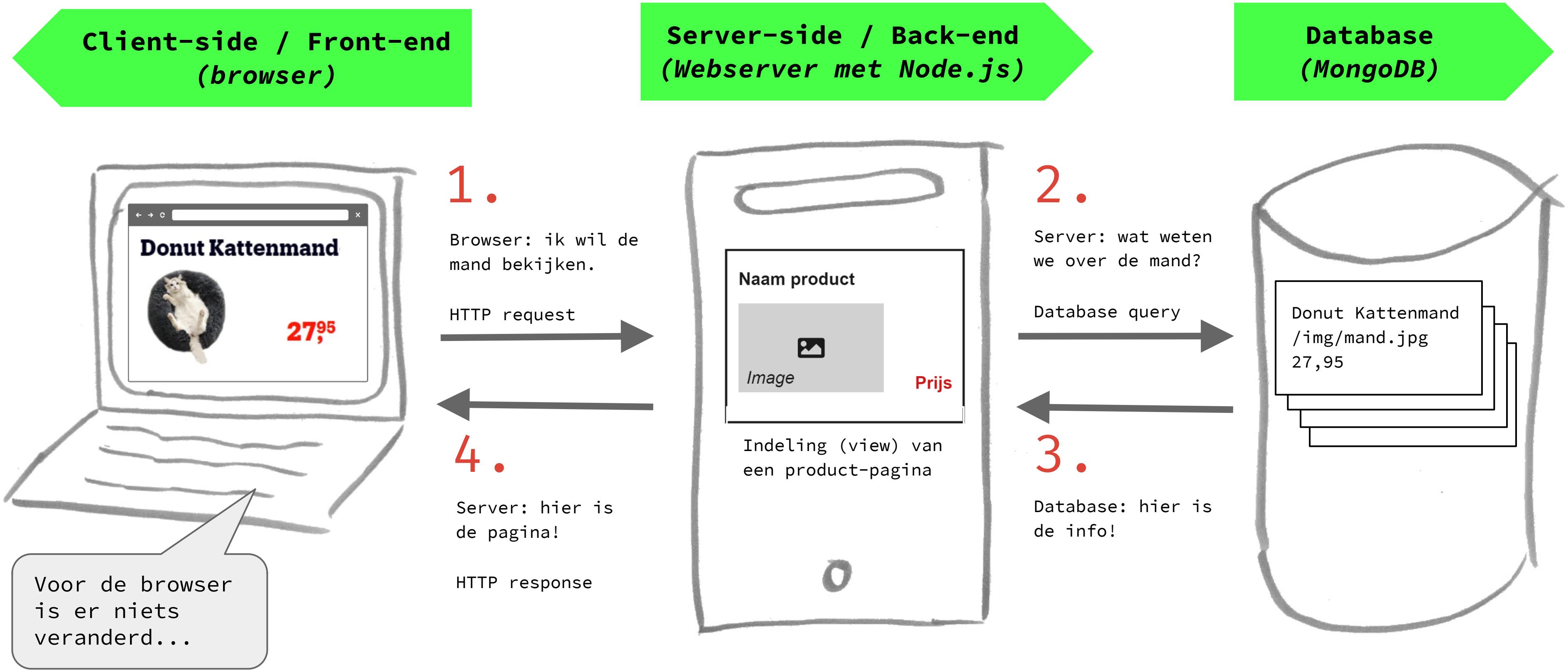
   200 OK, met de HTML file

Server-side / Back-end
(webserver)

- kattenmand.html
- kattenvoer.html
- krabpaal.html
- waterbakje.html
- speelgoedmuis.html
- *en nog 100en anderen....*

Uiteindelijk kan een programmeur dit niet meer allemaal met de hand doen

Site à la Blok Tech



Client-side / Front-end
(browser)

Server-side / Back-end
(Webserver met Node.js)

Database
(MongoDB)

1. Browser: ik wil de mand bekijken.

HTTP request

2. Server: wat weten we over de mand?

Database query

Donut Kattenmand
/img/mand.jpg
27,95

3. Database: hier is de info!

Indeling (view) van een product-pagina

4. Server: hier is de pagina!

HTTP response

Voor de browser is er niets veranderd...

Browser vs node

libs

In the **Browser** you get JS and…

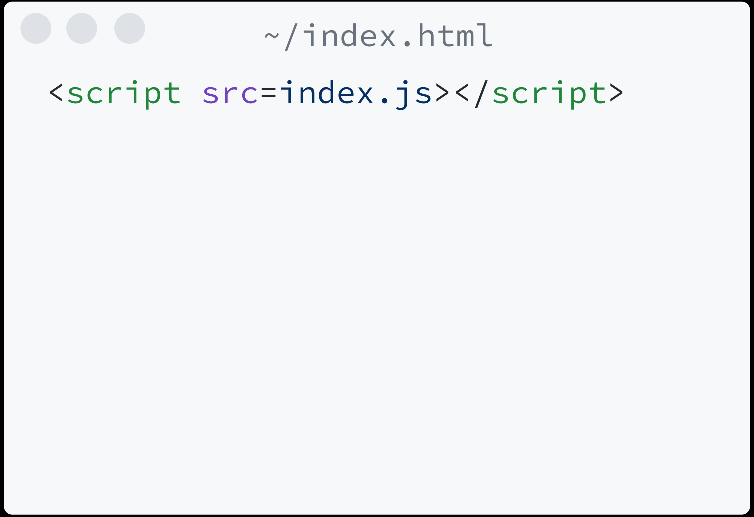
- ❖ Console (console.log, …)
- ❖ Timers (setTimeout, …)
- ❖ window
- ❖ document (DOM)
- ❖ XMLHttpRequest (or fetch)
- ❖ <script>
- ❖ Canvas / WebGL
- ❖ …

In **Node.js** you get JS and…

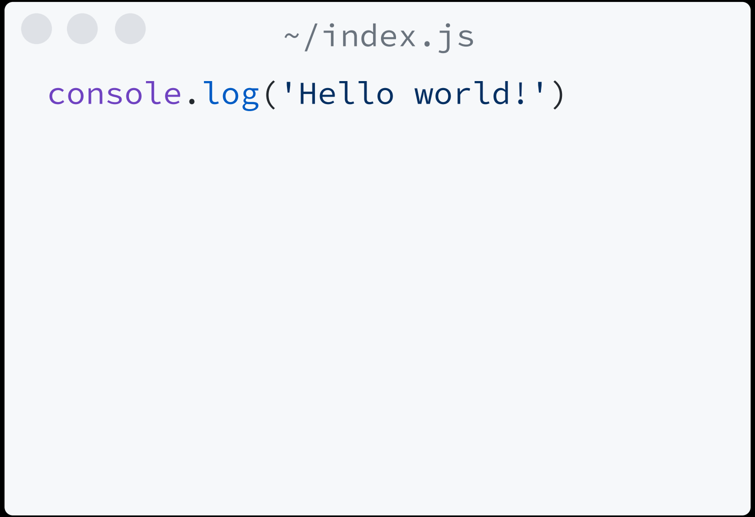
- ❖ Console (console.log, …)
- ❖ Timers (setTimeout, …)
- ❖ global
- ❖ File System (fs)
- ❖ http
- ❖ require / module
- ❖ Buffer
- ❖ …

browser

node

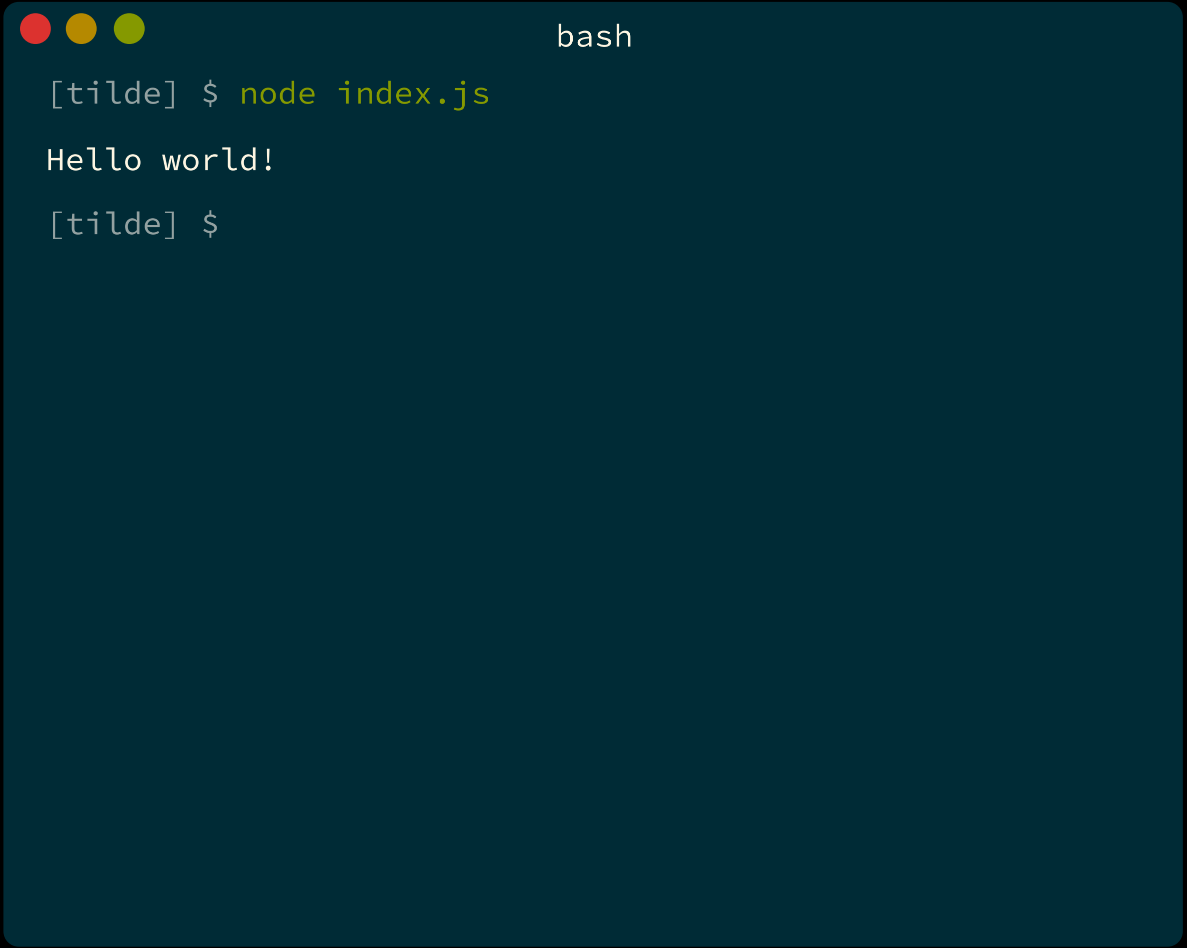

~/index.html

```html
<script src=index.js></script>
```


~/index.js

```js
console.log('Hello world!')
```

node



```
[tilde] $ node index.js

Hello world!

[tilde] $
```

terminal

node

why learn node?

Yes, you should know Node (but it depends)

Frontend developers **should…**

- ❖ Know what backend developers do
- ❖ Be comfortable with Node-based tooling
- ❖ Have a basic understanding of web servers and databases
- ❖ Be able to build a basic prototype
- ❖ Same language on front-end and back-end (js)

assignment (+10 min)

Create a .js file somewhere on your computer and run it with node.js.

- Check if node is installed (node -v)
- Create a server.js that outputs ‘Hello World’
- Run it with node and see the output
- What happens when you log something like *document*?

Modules

folder/index.js

```
console.log(sum(1, 2, 3))

function sum() {
  var args = arguments
  var total = 0
  var index = -1

  while (++index < args.length) {
    total += args[index]
  }

  return total
}
```

modules

scripts

folder/sum.js

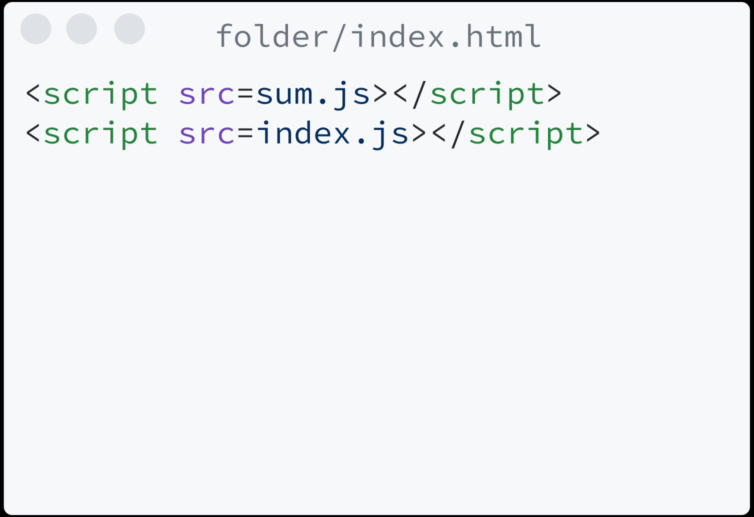
```js
function sum() {
  var args = arguments
  var total = 0
  var index = -1
  while (++index < args.length) {
    total += args[index]
  }
  return total
}
```

folder/index.js

```js
console.log(sum(1, 2, 3))
```

modules

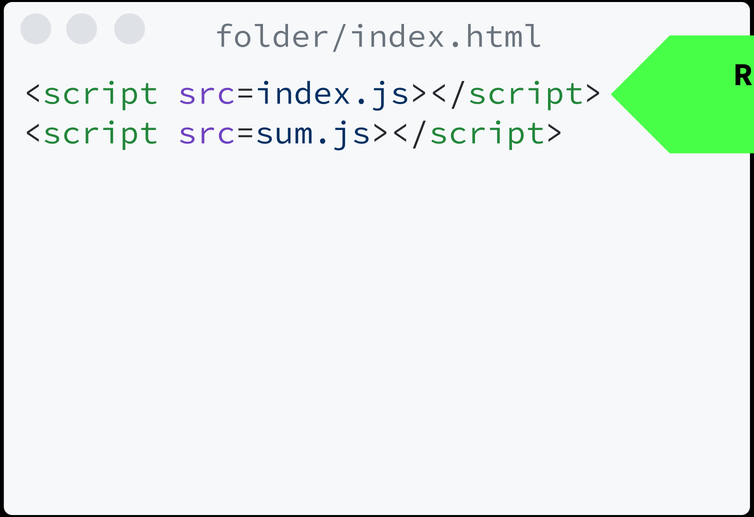
scripts


folder/index.html

```
<script src=sum.js></script>
<script src=index.js></script>
```

modules

errors



folder/index.html

```html
<script src=index.js></script>
<script src=sum.js></script>
```

ReferenceError: Can't find variable: sum

modules

require

folder/sum.js

```
module.exports = sum

function sum() {
  var args = arguments
  var total = 0
  var index = -1
  while (++index < args.length) {
    total += args[index]
  }
  return total
}
```

folder/index.js

```
var sum = require('./sum.js')

console.log(sum(1, 2, 3))
```

modules

folders

```
// Files
folder/
├─ index.js
└─ modules/
   └─ sum.js

// Command line
[folder] $ node index.js
6
```

```js
// index.js
var sum = require(                    )

console.log(sum(1, 2, 3))

// sum.js
module.exports = sum

function sum() {
  var args = arguments
  var total = 0
  var index = -1

  while (++index < args.length) {
    total += args[index]
  }

  return total
}
```

assignment (+20 min)

Create your own module by creating 2 javascript files and importing them

- Create a student.js that has an object with your info
- Export the student object as a module
- Import (require) that file in a js file called class.js
- Log the output of the student object

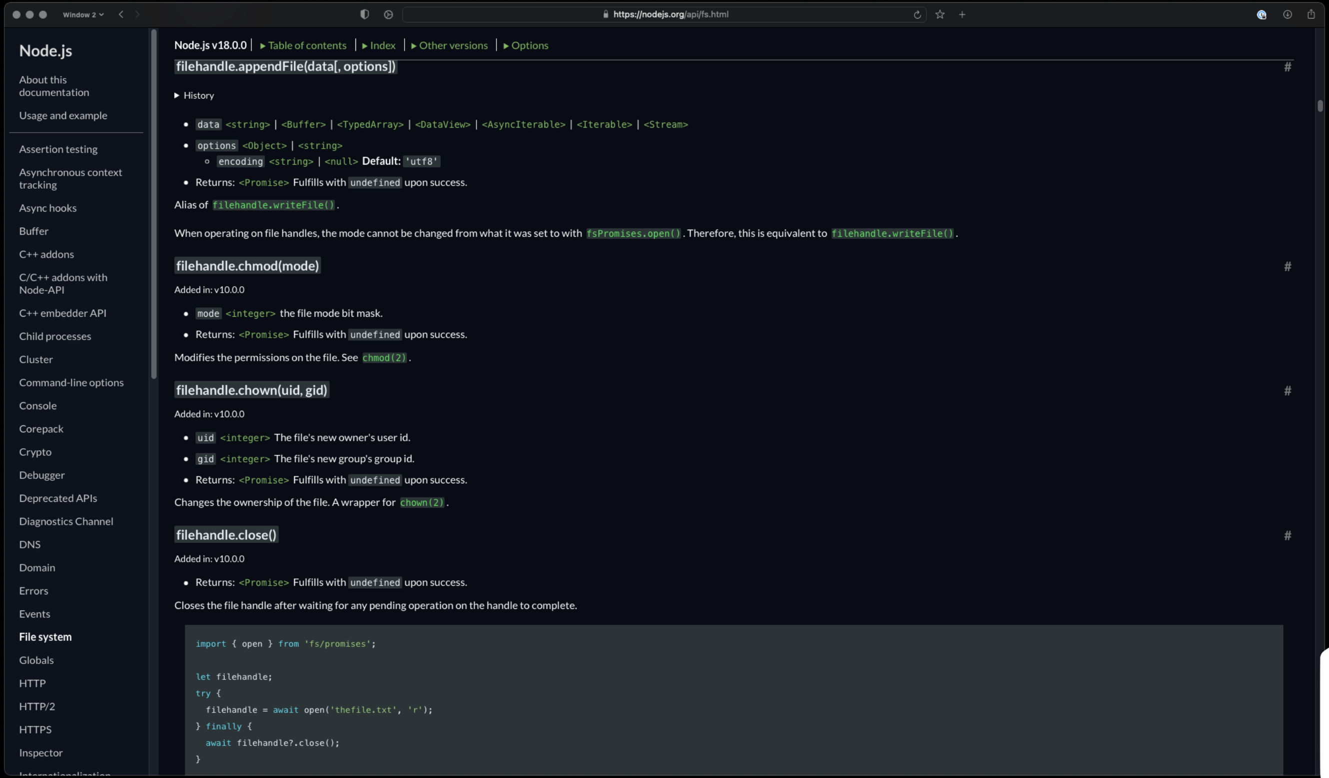
modules

?

Q: So, do I need to write all my modules?
A: **No!**

modules

Built-in



https://nodejs.org/api/fs.html

Node.js

About this documentation

Usage and example

Assertion testing

Asynchronous context tracking

Async hooks

Buffer

C++ addons

C/C++ addons with Node-API

C++ embedder API

Child processes

Cluster

Command-line options

Console

Corepack

Crypto

Debugger

Deprecated APIs

Diagnostics Channel

DNS

Domain

Errors

Events

File system

Globals

HTTP

HTTP/2

HTTPS

Inspector

Node.js v18.0.0 | ▸ Table of contents | ▸ Index | ▸ Other versions | ▸ Options

filehandle.appendFile(data[, options])

▸ History

- `data` <string> | <Buffer> | <TypedArray> | <DataView> | <AsyncIterable> | <Iterable> | <Stream>
- `options` <Object> | <string>
  - `encoding` <string> | <null> **Default:** `'utf8'`
- Returns: <Promise> Fulfills with `undefined` upon success.

Alias of `filehandle.writeFile()`.

When operating on file handles, the mode cannot be changed from what it was set to with `fsPromises.open()`. Therefore, this is equivalent to `filehandle.writeFile()`.

filehandle.chmod(mode)

Added in: v10.0.0

- `mode` <integer> the file mode bit mask.
- Returns: <Promise> Fulfills with `undefined` upon success.

Modifies the permissions on the file. See `chmod(2)`.

filehandle.chown(uid, gid)

Added in: v10.0.0

- `uid` <integer> The file's new owner's user id.
- `gid` <integer> The file's new group's group id.
- Returns: <Promise> Fulfills with `undefined` upon success.

Changes the ownership of the file. A wrapper for `chown(2)`.

filehandle.close()

Added in: v10.0.0

- Returns: <Promise> Fulfills with `undefined` upon success.

Closes the file handle after waiting for any pending operation on the handle to complete.

```
import { open } from 'fs/promises';

let filehandle;
try {
  filehandle = await open('thefile.txt', 'r');
} finally {
  await filehandle?.close();
}
```

NPM

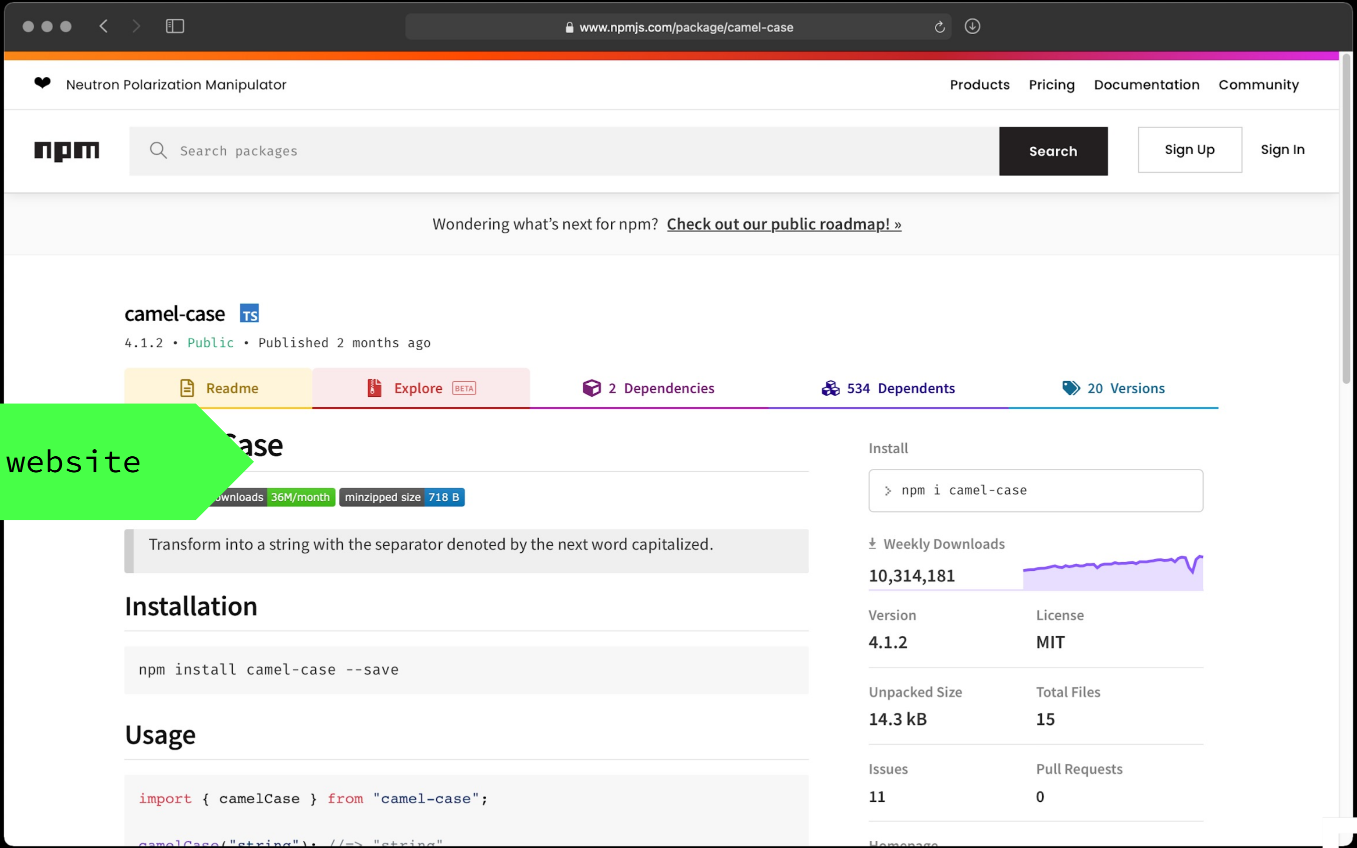
NPM

?

npm is a package manager for the JavaScript programming language. It is the default package manager for […] Node.js. It consists of a command line client, also called npm, and an online database of […] packages, called the npm registry.

wikipedia.org

The **npm** website



www.npmjs.com/package/camel-case

Neutron Polarization Manipulator

Products Pricing Documentation Community

Search packages

Search Sign Up Sign In

Wondering what's next for npm? Check out our public roadmap! »

camel-case TS

4.1.2 • Public • Published 2 months ago

Readme Explore BETA 2 Dependencies 534 Dependents 20 Versions

ase

ownloads 36M/month minzipped size 718 B

> Transform into a string with the separator denoted by the next word capitalized.

Installation

```
npm install camel-case --save
```

Usage

```
import { camelCase } from "camel-case";

camelCase("string"); //=> "string"
```

Install

```
> npm i camel-case
```

Weekly Downloads

10,314,181

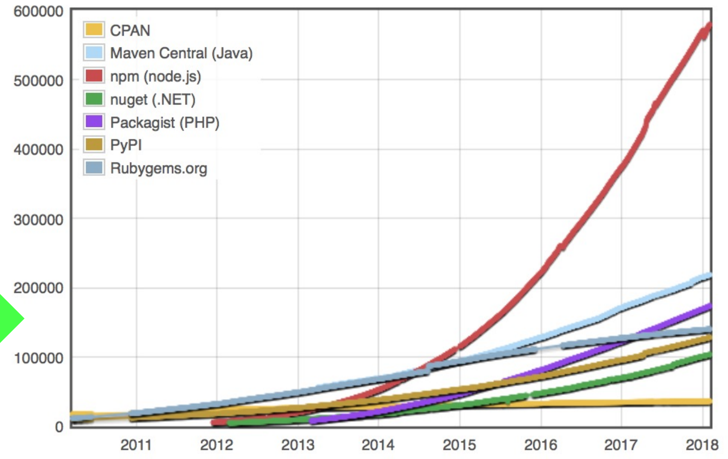
Version	License
4.1.2	MIT
Unpacked Size	Total Files
14.3 kB	15
Issues	Pull Requests
11	0

Homepage

npm is huge

www.modulecounts.com

Module Counts

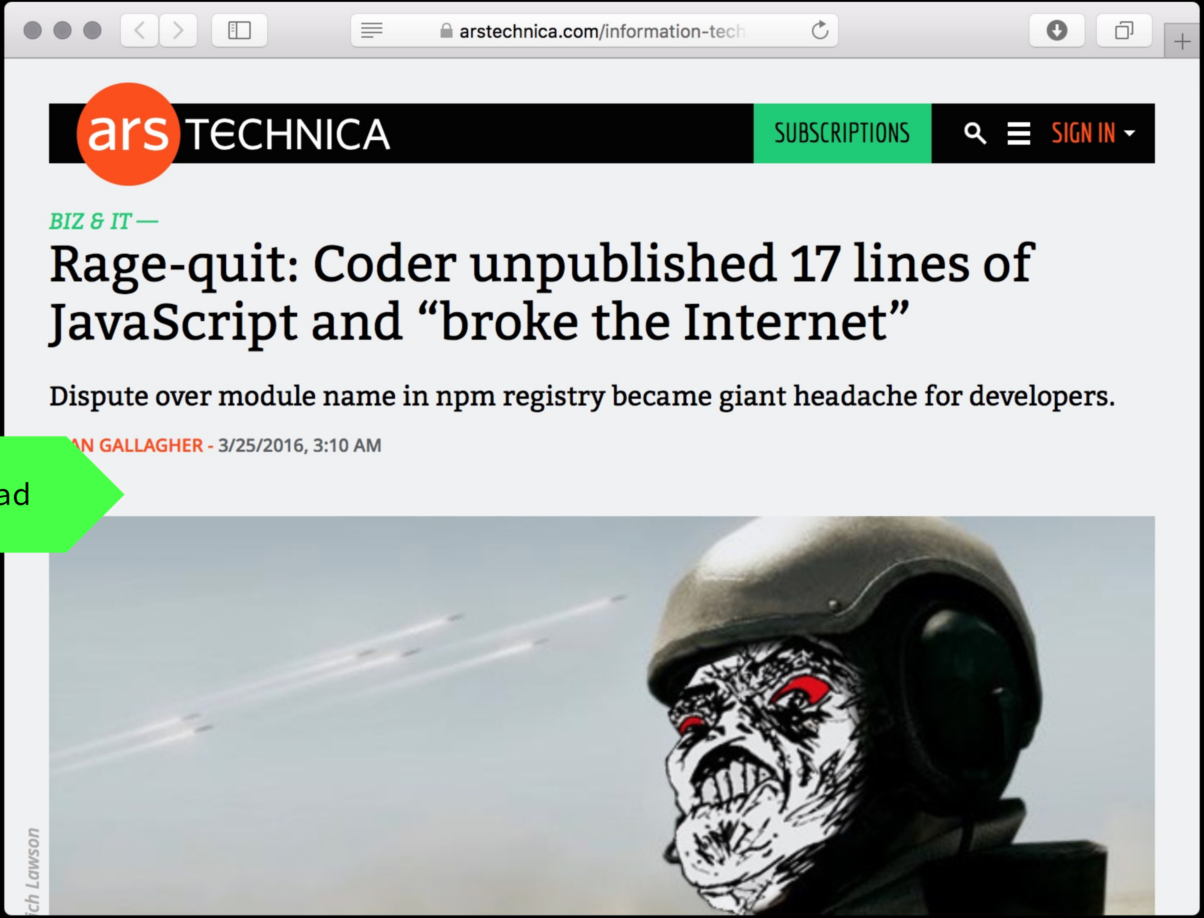


Include

- [ ] Clojars (Clojure)
- [✓] CPAN
- [ ] CPAN (search)
- [ ] CRAN (R)
- [ ] Crates.io (Rust)
- [ ] Drupal (php)
- [ ] DUB (dlang)
- [ ] Gopm (go)
- [ ] Hackage (Haskell)
- [ ] Hex.pm (Elixir/Erlang)
- [ ] Julia
- [ ] LuaRocks (Lua)
- [✓] Maven Central (Java)
- [ ] MELPA (Emacs)
- [ ] Nimble (Nim)
- [✓] npm (node.js)
- [✓] nuget (.NET)
- [✓] Packagist (PHP)
- [ ] Pear (PHP)
- [ ] Perl 6 Ecosystem (perl 6)
- [✓] PyPI
- [✓] Rubygems.org
- [ ] Vim Scripts

time period [✓ all time] last year last 90 days last 30 days last 7 days

	Jan 26	Jan 27	Jan 28	Jan 29	Jan 30	Jan 31	Feb 1	Avg Growth
Clojars (Clojure)	20942	20947	20952	20957	20969	20976	20984	7/day



arstechnica.com/information-tech

ars TECHNICA

SUBSCRIPTIONS

SIGN IN

BIZ & IT —

Rage-quit: Coder unpublished 17 lines of JavaScript and “broke the Internet”

Dispute over module name in npm registry became giant headache for developers.

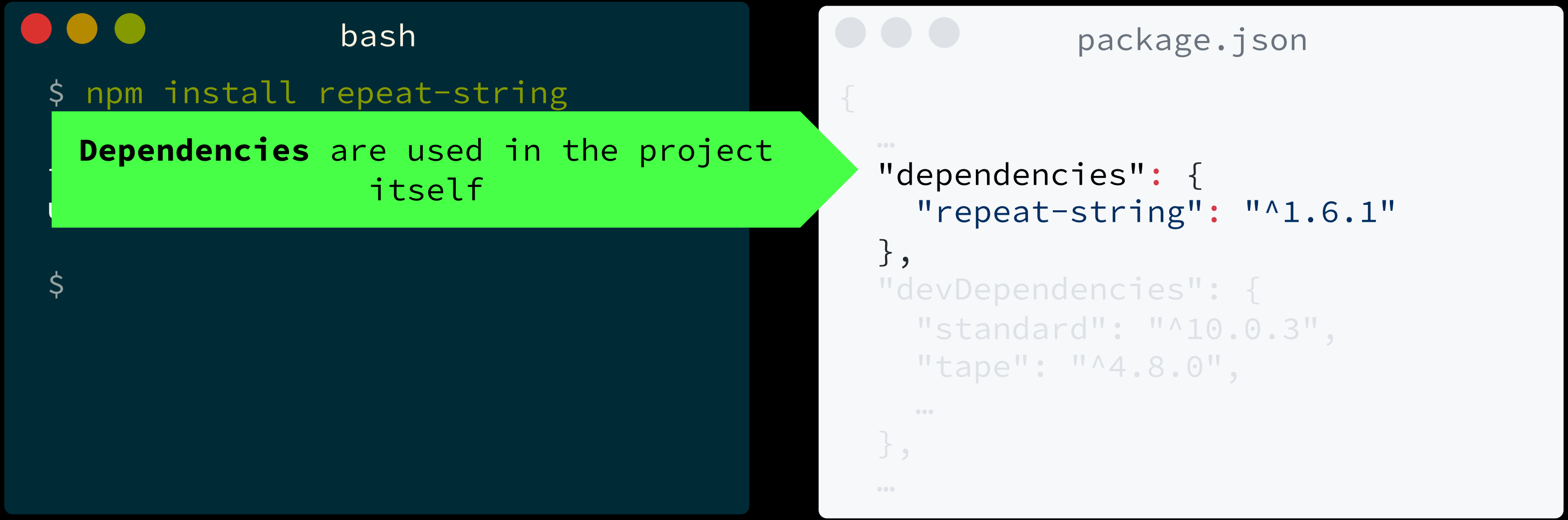
AN GALLAGHER - 3/25/2016, 3:10 AM

Developer got mad

npm

NPM

dependencies



bash

```
$ npm install repeat-string

$
```

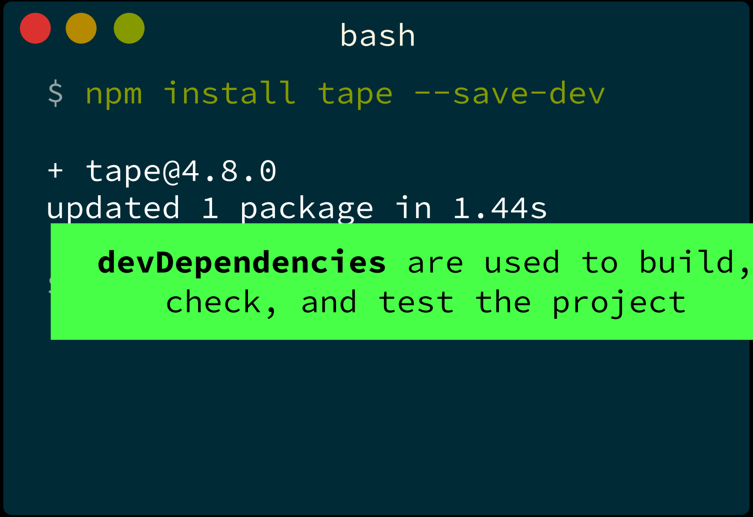
Dependencies are used in the project itself

package.json

```
{
  ...
  "dependencies": {
    "repeat-string": "^1.6.1"
  },
  "devDependencies": {
    "standard": "^10.0.3",
    "tape": "^4.8.0",
    ...
  },
  ...
```

NPM

dependencies

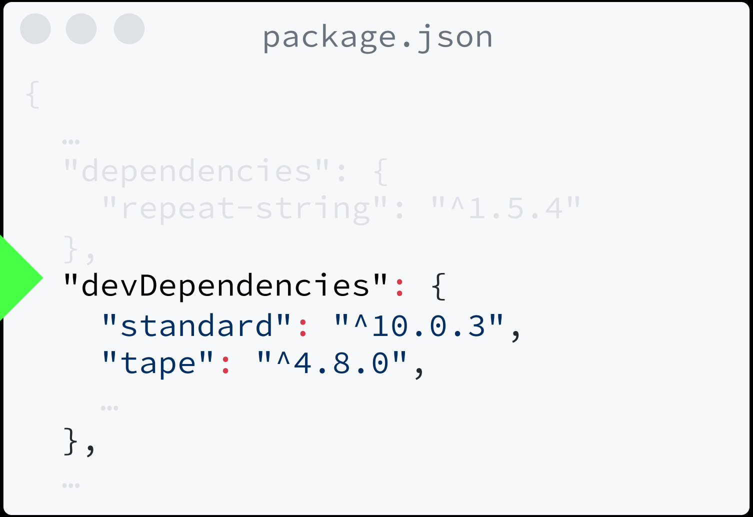


```
bash
$ npm install tape --save-dev

+ tape@4.8.0
updated 1 package in 1.44s

$
```

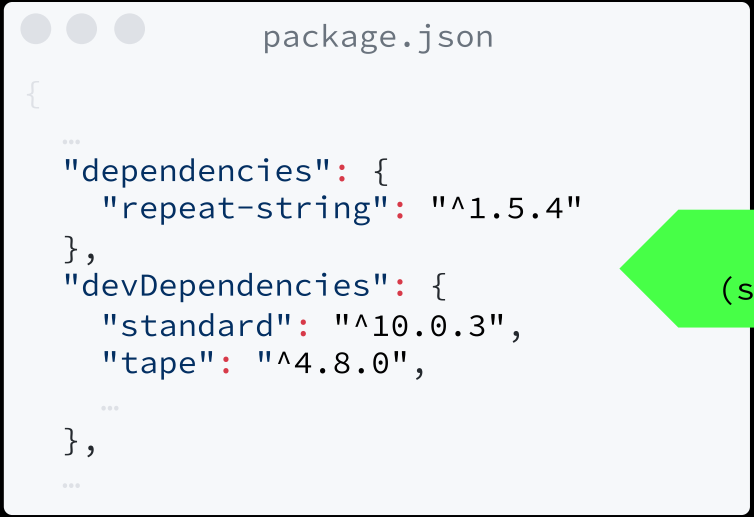
devDependencies are used to build, check, and test the project



```
package.json
{
  …
  "dependencies": {
    "repeat-string": "^1.5.4"
  },
  "devDependencies": {
    "standard": "^10.0.3",
    "tape": "^4.8.0",
    …
  },
  …
```

NPM

semver



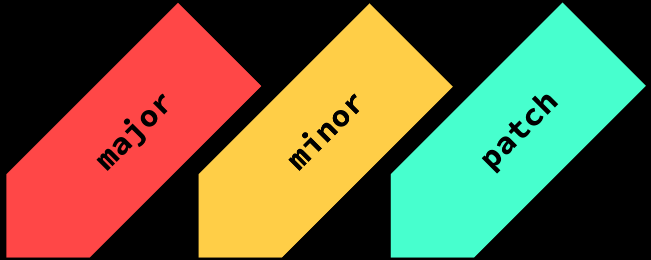
package.json

```
{
  ...
  "dependencies": {
    "repeat-string": "^1.5.4"
  },
  "devDependencies": {
    "standard": "^10.0.3",
    "tape": "^4.8.0",
    ...
  },
  ...
```

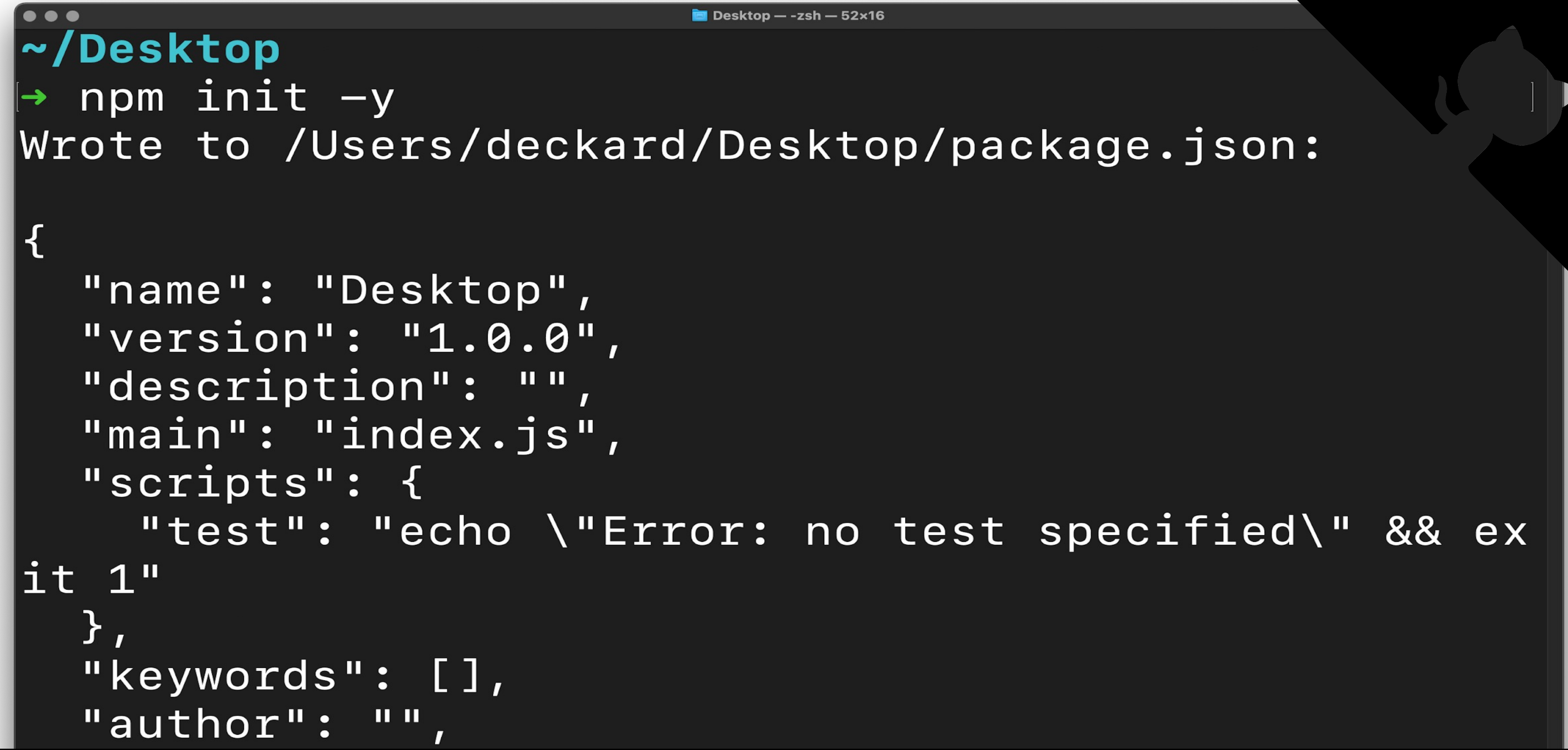
semver
(semantic versioning)

NPM

versions



```
"version": "2.5.1"
```

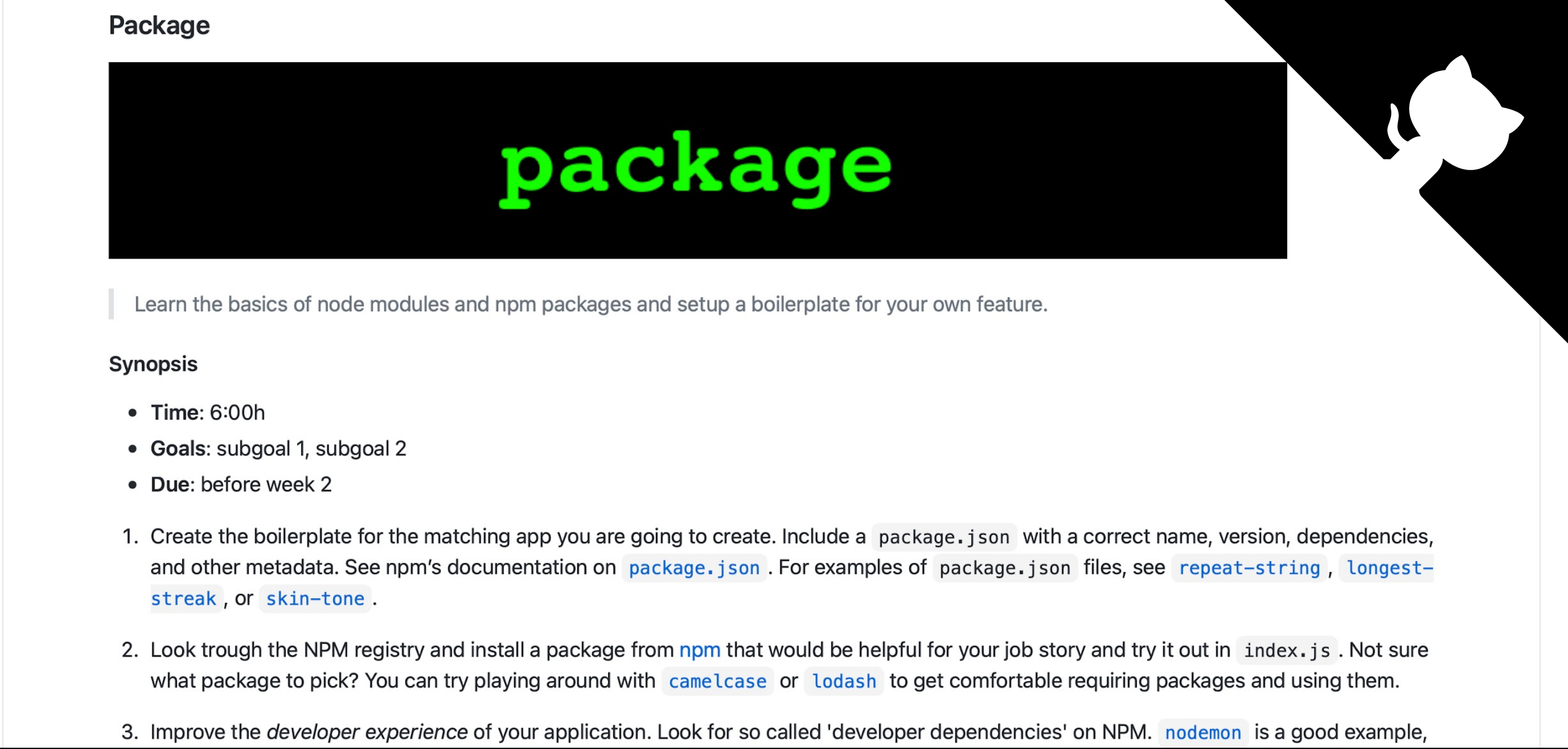


Desktop — -zsh — 52×16

```
~/Desktop
→ npm init -y
Wrote to /Users/deckard/Desktop/package.json:

{
  "name": "Desktop",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && ex
it 1"
  },
  "keywords": [],
  "author": "",
```

Live demo npm en packages



Package

package

> Learn the basics of node modules and npm packages and setup a boilerplate for your own feature.

Synopsis

- **Time**: 6:00h
- **Goals**: subgoal 1, subgoal 2
- **Due**: before week 2

1. Create the boilerplate for the matching app you are going to create. Include a `package.json` with a correct name, version, dependencies, and other metadata. See npm's documentation on `package.json`. For examples of `package.json` files, see `repeat-string`, `longest-streak`, or `skin-tone`.
2. Look trough the NPM registry and install a package from npm that would be helpful for your job story and try it out in `index.js`. Not sure what package to pick? You can try playing around with `camelcase` or `lodash` to get comfortable requiring packages and using them.
3. Improve the *developer experience* of your application. Look for so called 'developer dependencies' on NPM. `nodemon` is a good example,

work on package

exit;

see you in lab-2!