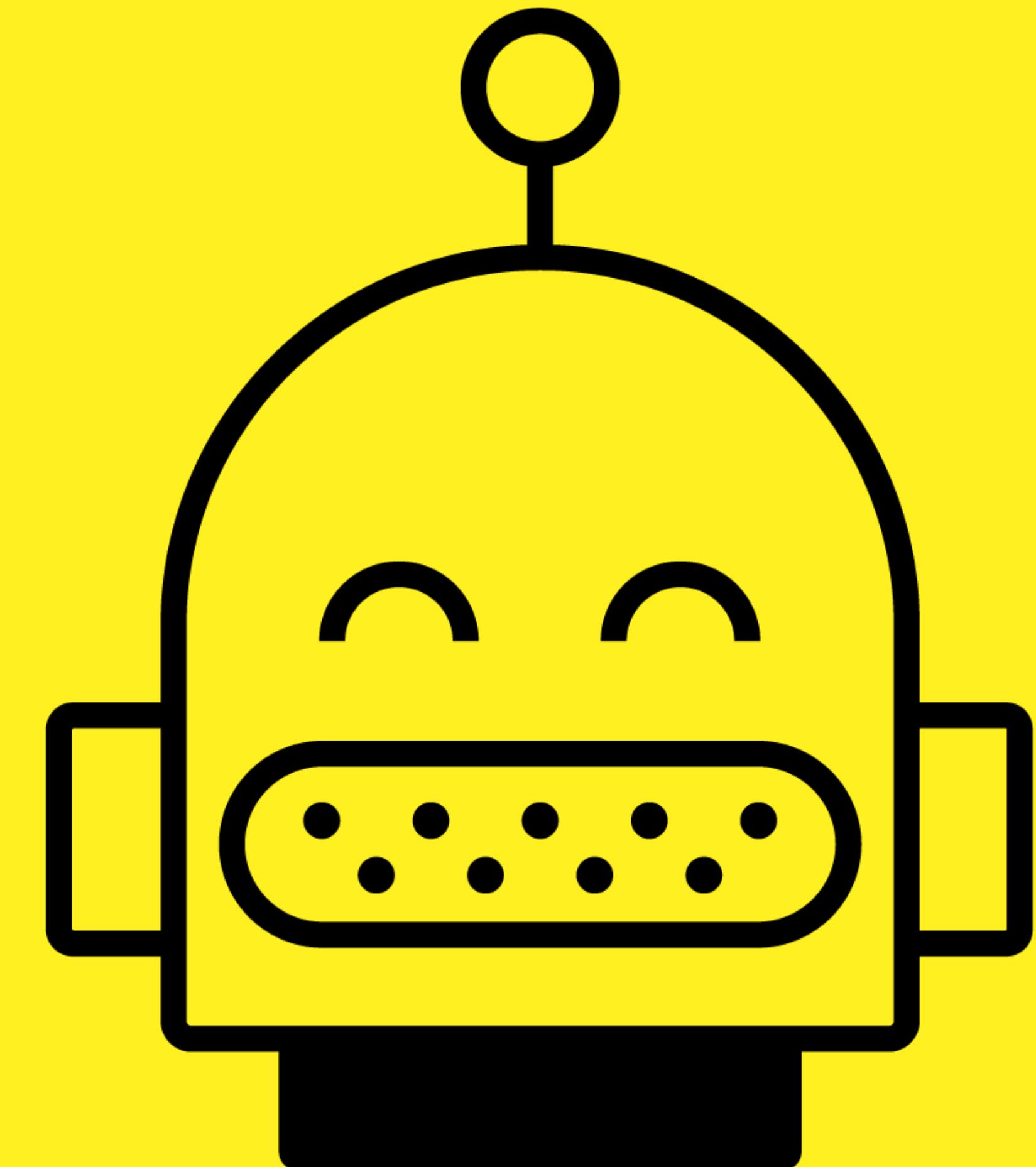


Project Tech

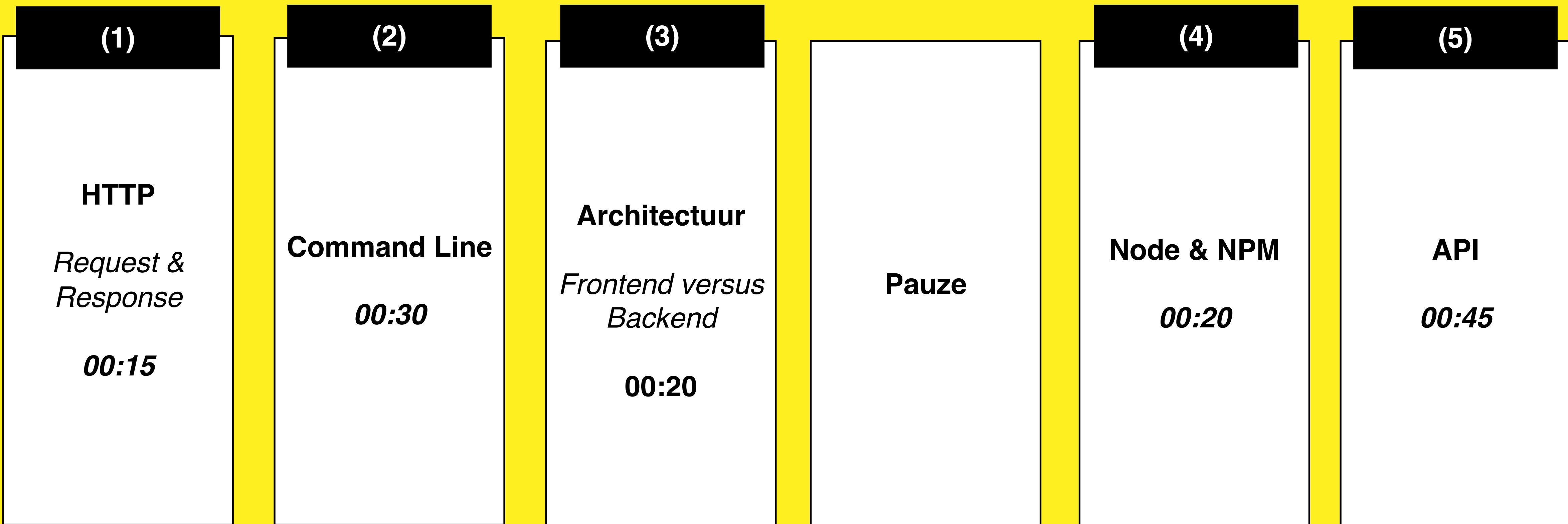
Les 2.1 Backend basis

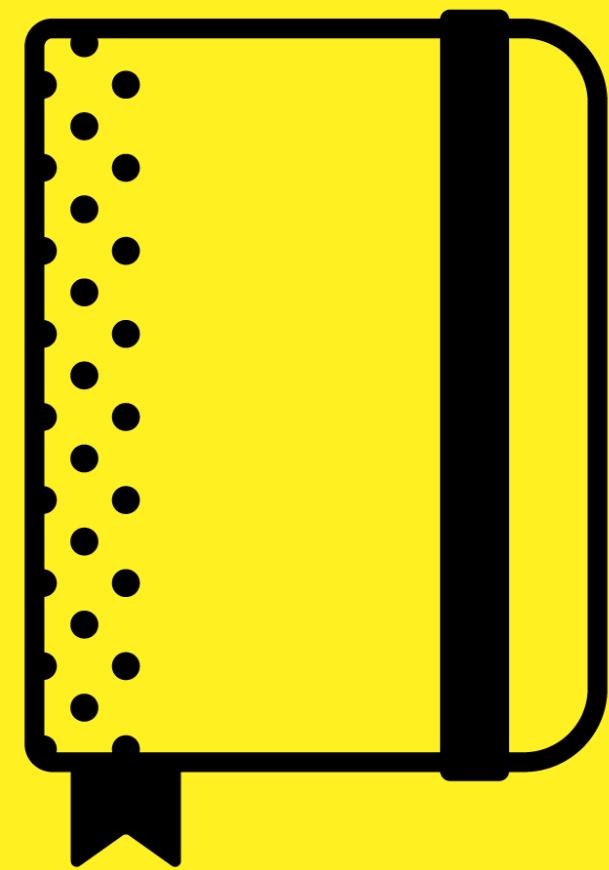
Architectuur en NodeJS



Creating Tomorrow

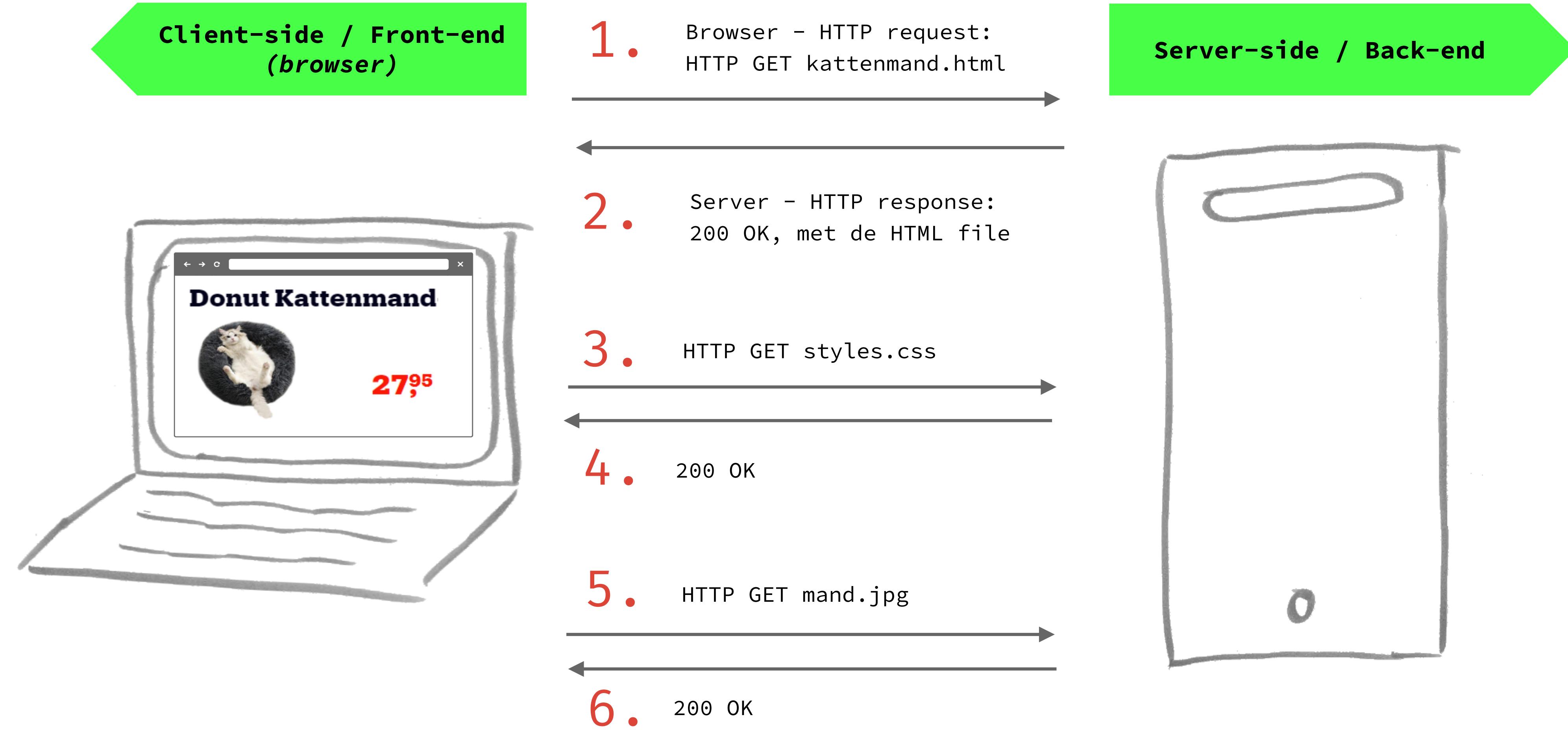
Planning



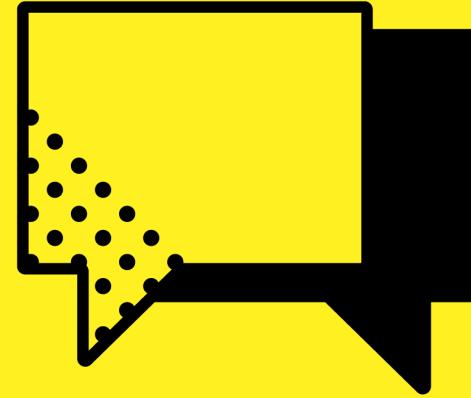


Theorie
HTTP

Request / Response



- Hypertext Transfer Protocol (**HTTP**) wordt gebruikt voor het uitwisselen van bestanden tussen browsers (clients) en webservers
- De browser doet voor elk bestand een **HTTP request** en de server antwoordt steeds met een **HTTP response**
- HTTPS (secure) werkt hetzelfde, maar gebruikt op de achtergrond een beveiligde verbinding. (*En dat is fijn, want met gewoon HTTP kan iedereen zonder veel moeite meeluisteren.*)



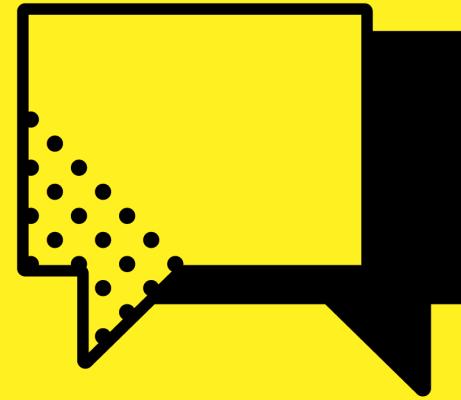
HTTP request method

Elke HTTP request heeft een method:

- **GET** - voor het opvragen van data van de server
- **POST** - om (formulier)data naar de server te sturen.

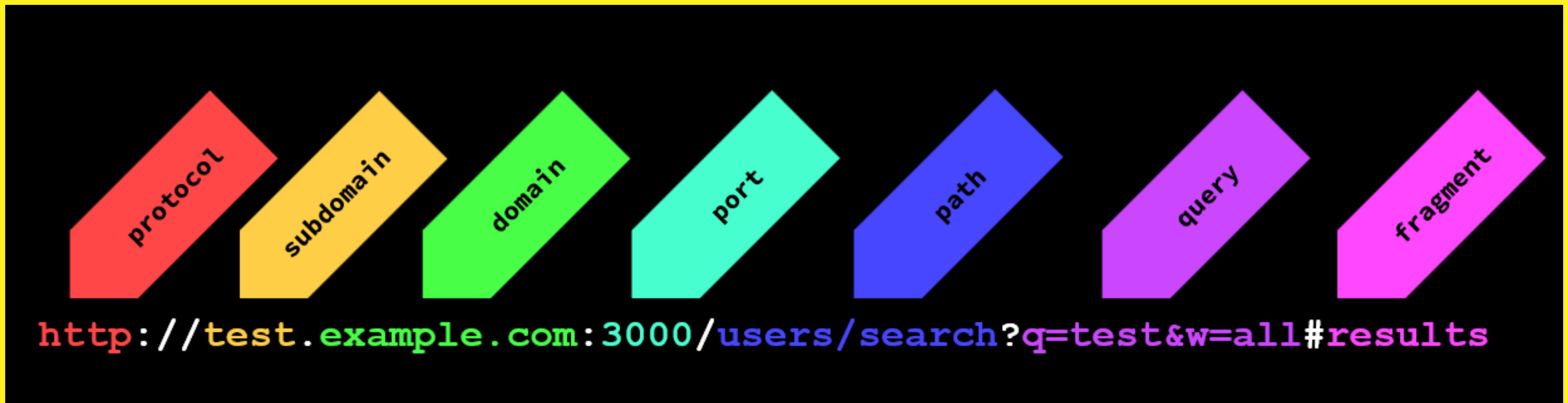
Ook dan krijg je altijd een response terug.

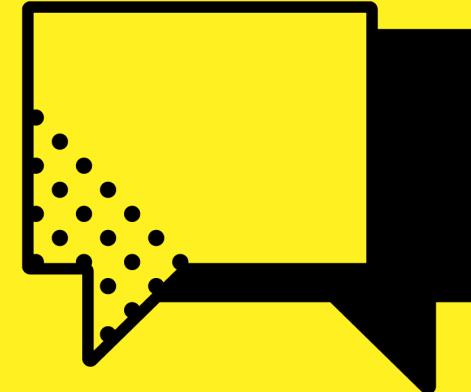
- Er zijn meer methods, bv: **PUT** (aanmaken data op server), **PATCH** (update data) of **DELETE**



HTTP request URL

Elke HTTP request heeft ook een URL





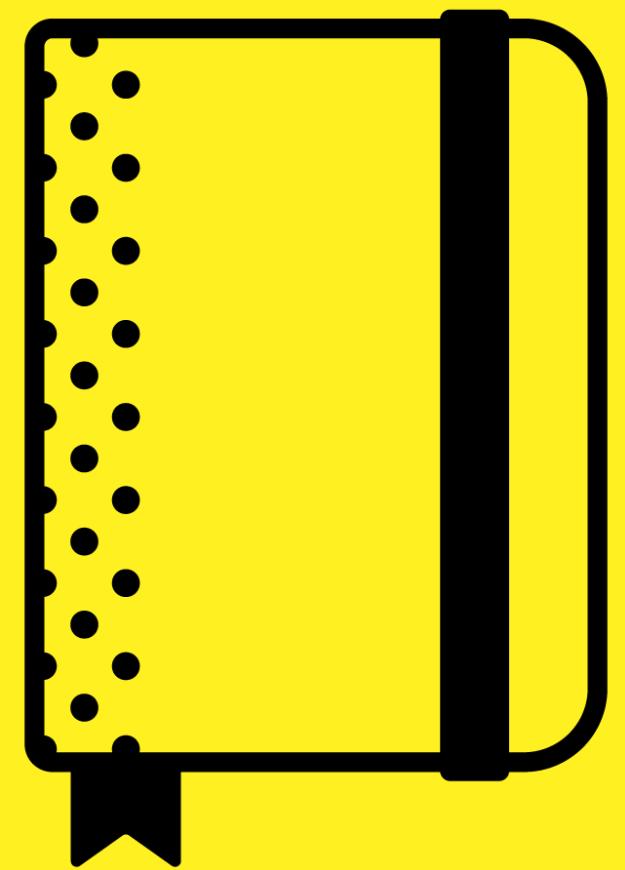
HTTP response

De HTTP response bevat een **status code** en (als alles goed gaat) **het bestand** van de opgegeven URL.

Bekende statuscodes zijn:

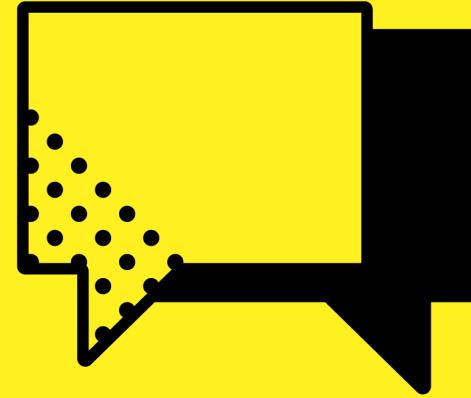
- 200 OK
- 404 Not Found
- 500 Internal Server Error
- 301 Moved Permanently

Er zijn er nog heel veel meer...



Theorie

Command line



Waarom de command line?

Je typt instructies voor de computer. Voor developers is dit vaak:

- Krachtiger, je hebt veel meer opties dan in de GUI
- Sneller, je hoeft niet allerlei venster open te klikken
- Soms de enige manier. De meeste servers hebben geen GUI.

```
[ ~ ] $ rm -f foo.txt
```

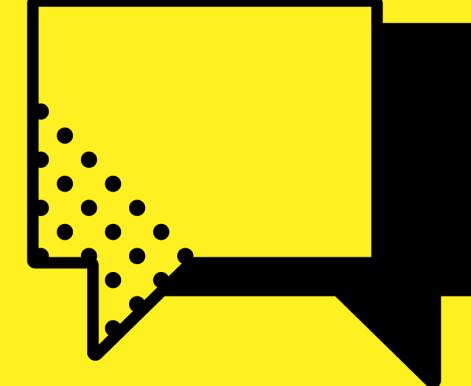
Prompt

Command

options

Arguments

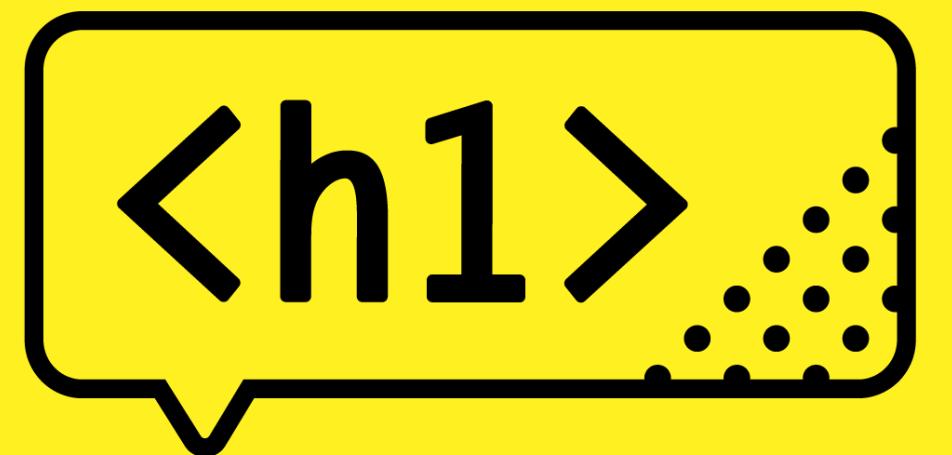
Command line voorbeeld



Hoe start je de command line?

- ☞ Op Mac: open een Terminal
- ☞ Op Windows: start PowerShell
- ☞ Ook kun je een Terminal starten binnen VS Code

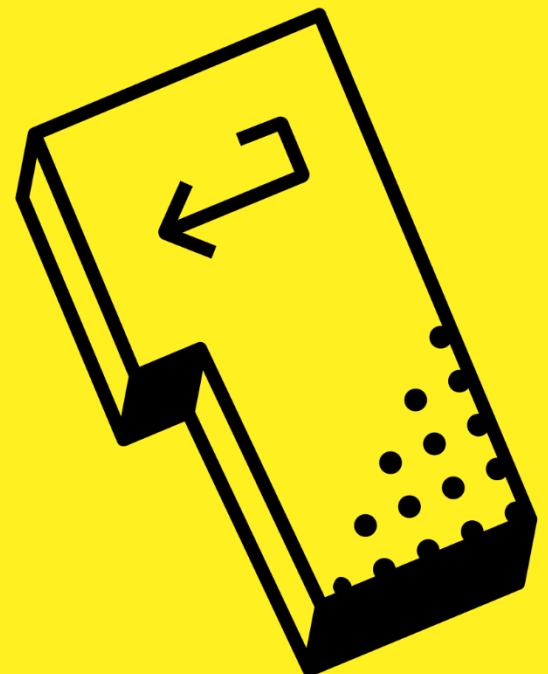


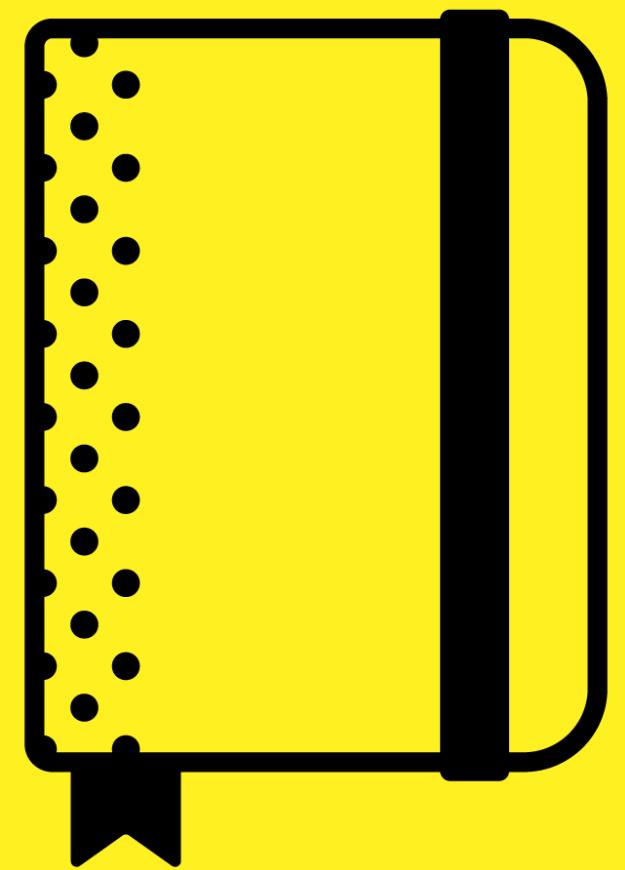


Lesopdracht

Oefenen met Command Line

Maak de oefening op GitHub





Theorie

Architectuur

Frontend & Backend

Site à la Internetstandaarden

Client-side / Front-end
(browser)



Server-side / Back-end
(webserver)

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

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



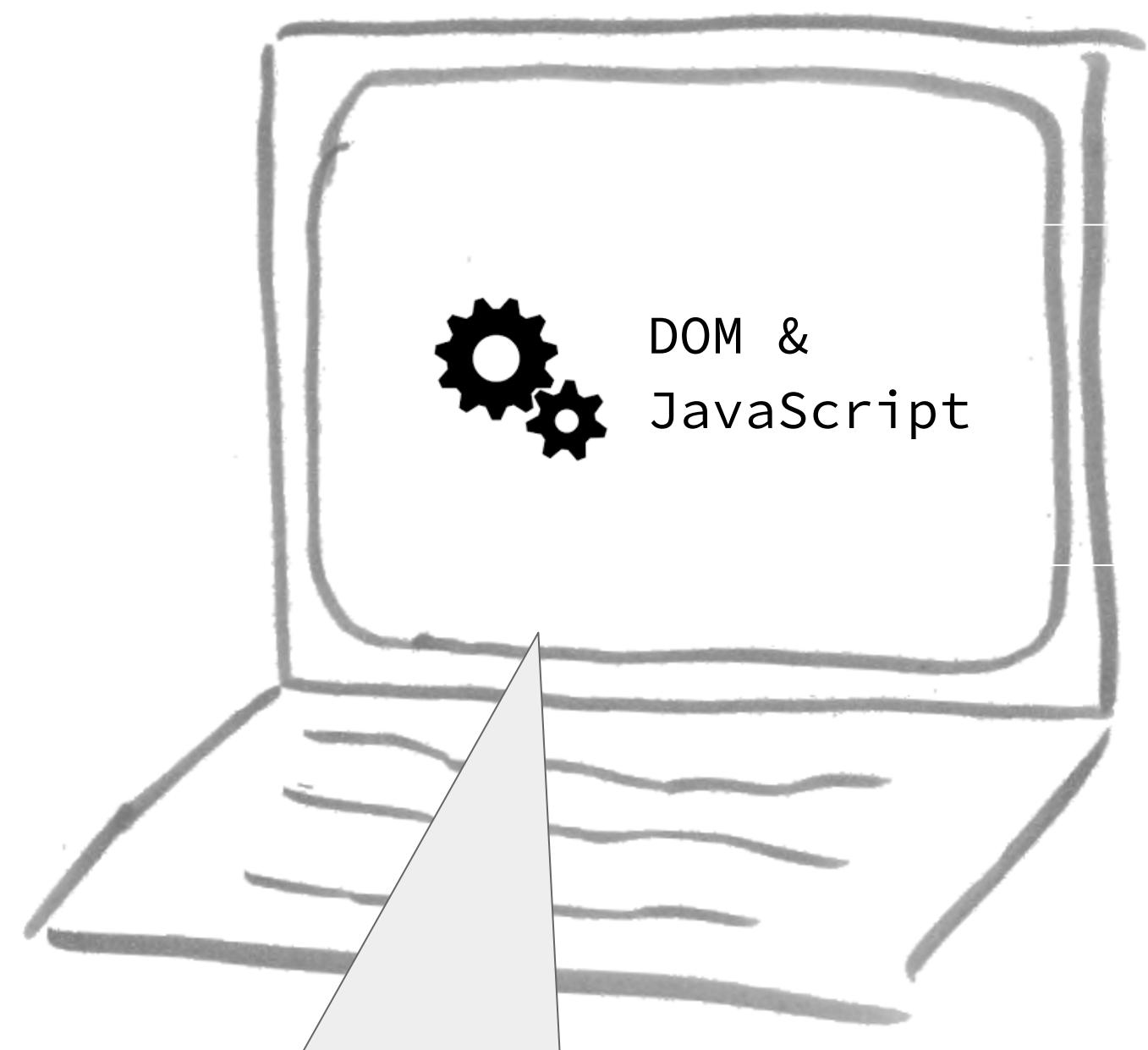
Uiteindelijk
kan een
programmeur
dit niet meer
allemaal met
de hand doen

Optie 1

Functionaliteit in de

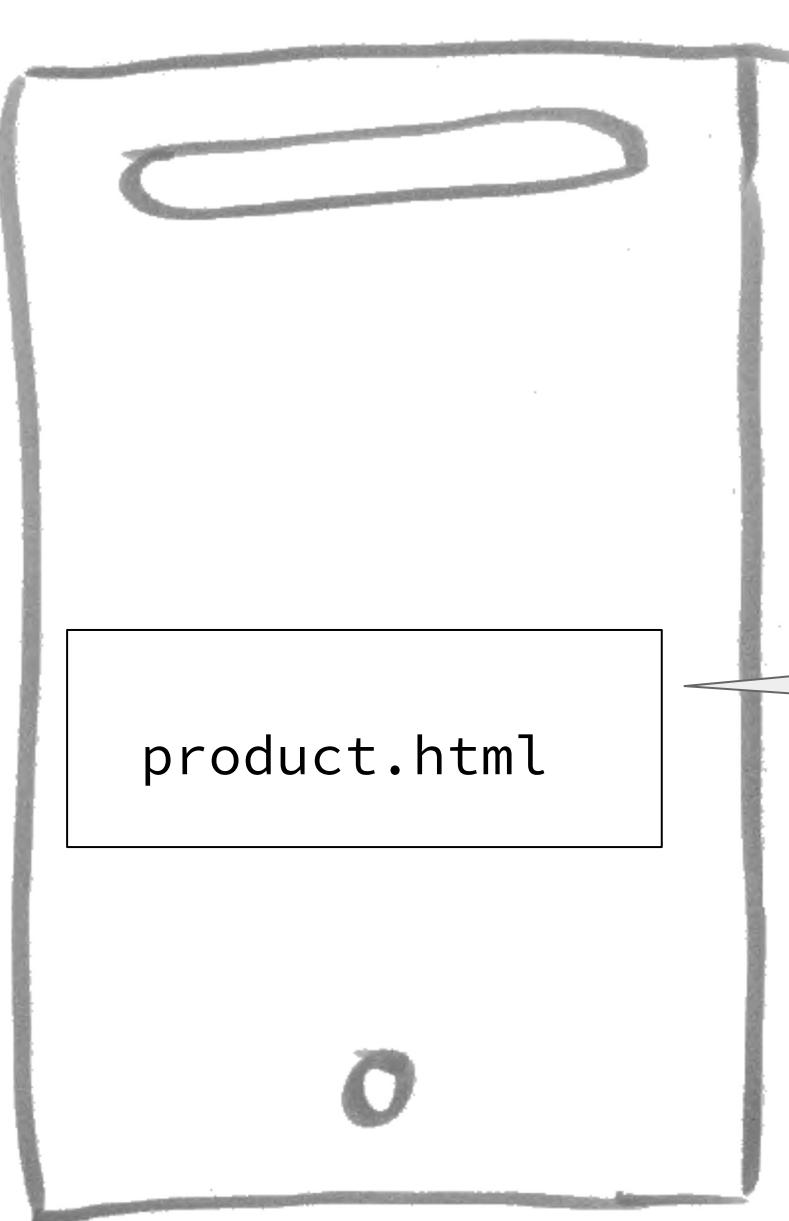
Front-end

**Client-side / Front-end
(browser)**



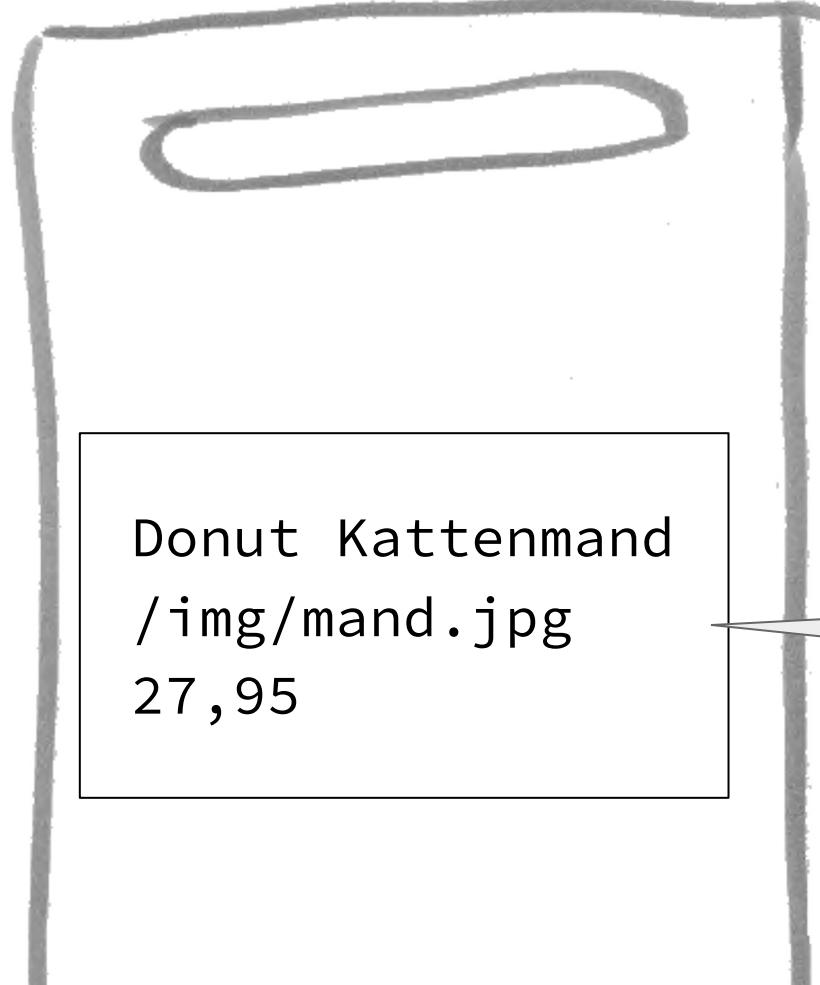
De browser verandert ter plekke de HTML door met JavaScript de DOM te manipuleren

1. Browser: ik wil de generieke productpagina.
HTTP request
2. Server: hier is de pagina!
HTTP response
3. Browser: wat weten we over de mand?
HTTP request naar API
4. API: hier is de info!
HTTP response



**Server-side / Back-end
(Webserver)**

Onze webserver serveert nog steeds statische HTML



API

De API is eigenlijk ook een webserver die data serveert

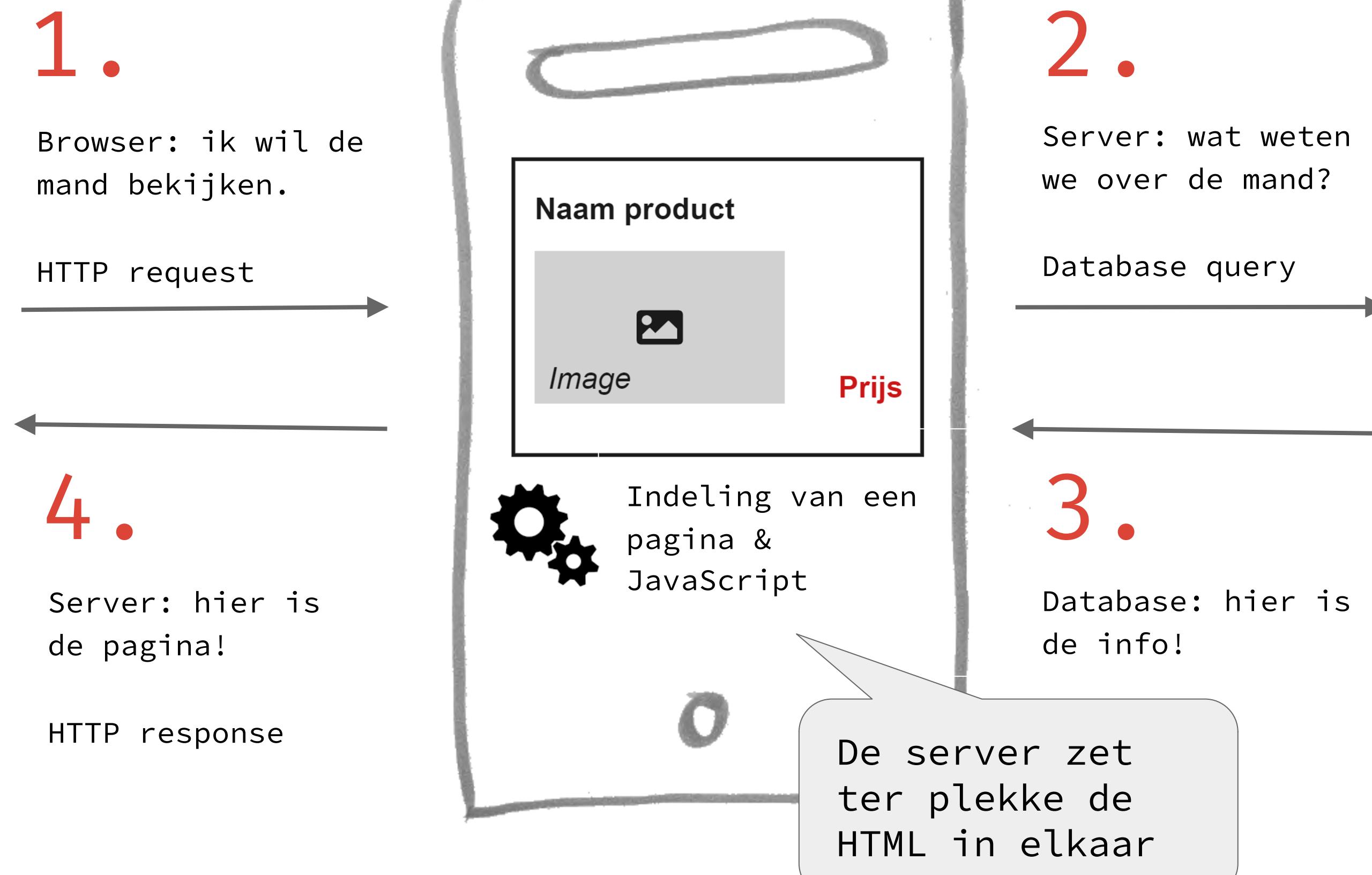
Optie 2

Functionaliteit in de backend

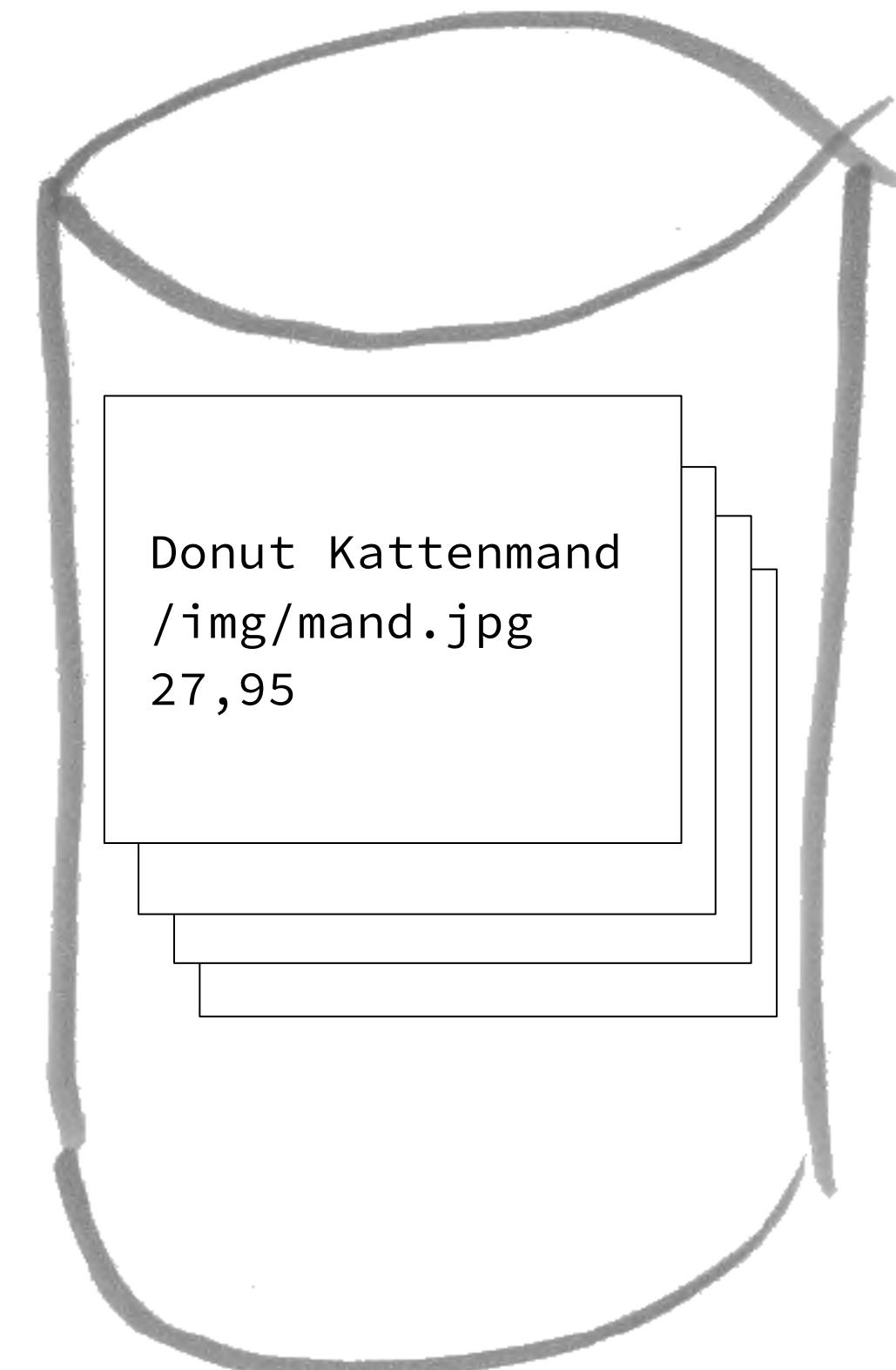
**Client-side / Front-end
(browser)**

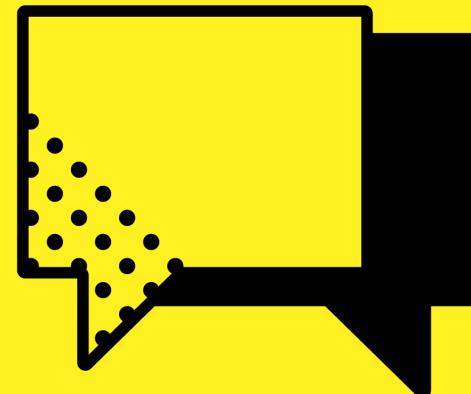


**Server-side / Back-end
(Webserver met Node.js)**



**Database
(MongoDB)**

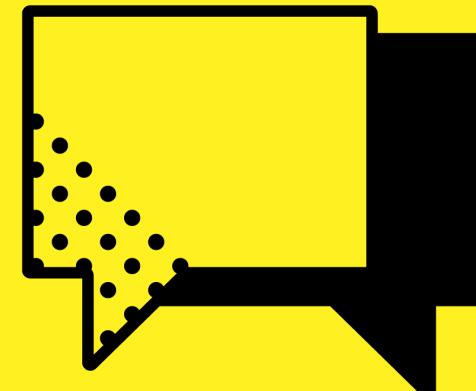




Wat is dan het beste?

Functionaliteit bouwen in de frontend of in de backend?

- Het zijn alletwee prima opties
- Frontend kan snel en prettig werken, want de DOM kan worden bijgewerkt zonder nieuwe pagina te laden.
- Backend is handig voor meer complexe logica en security, en meer controle over je data.
- Je kunt beide opties zelfs combineren

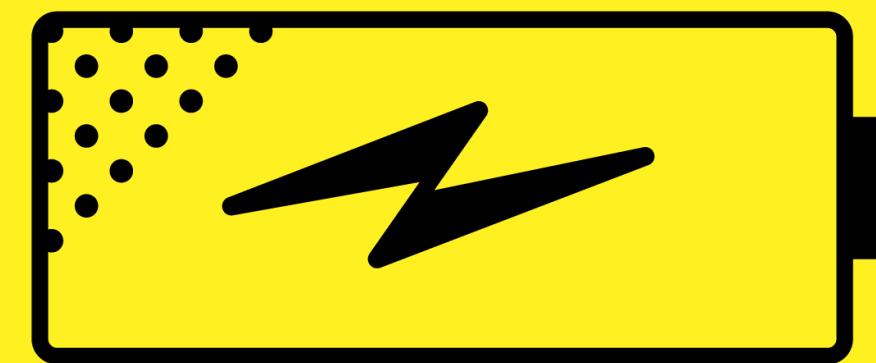


Frontend of backend?

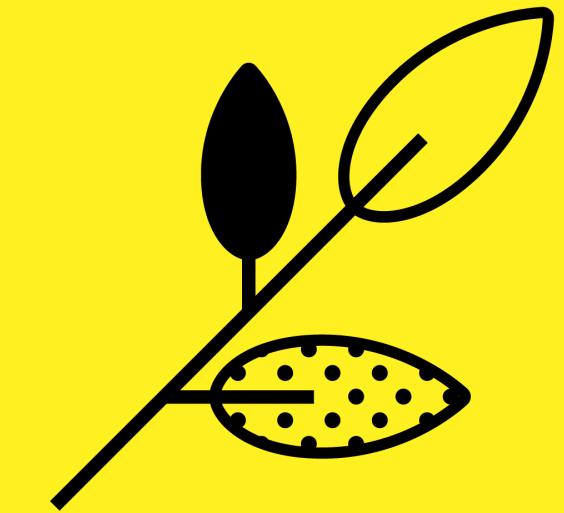
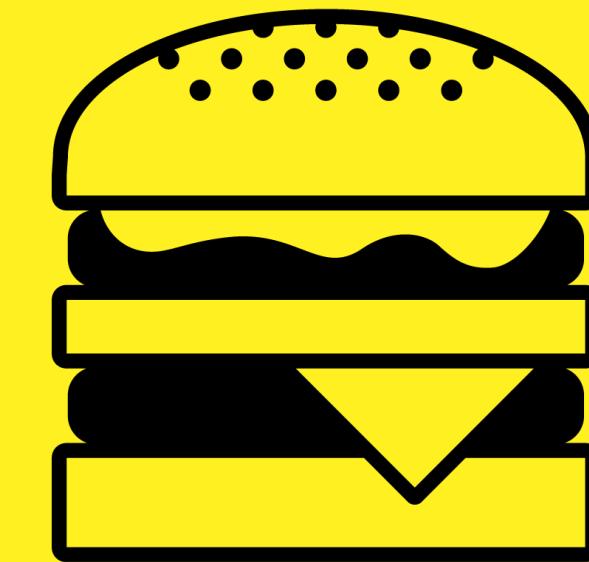
In dit project mag je kiezen je te verdiepen in de frontend of in de backend

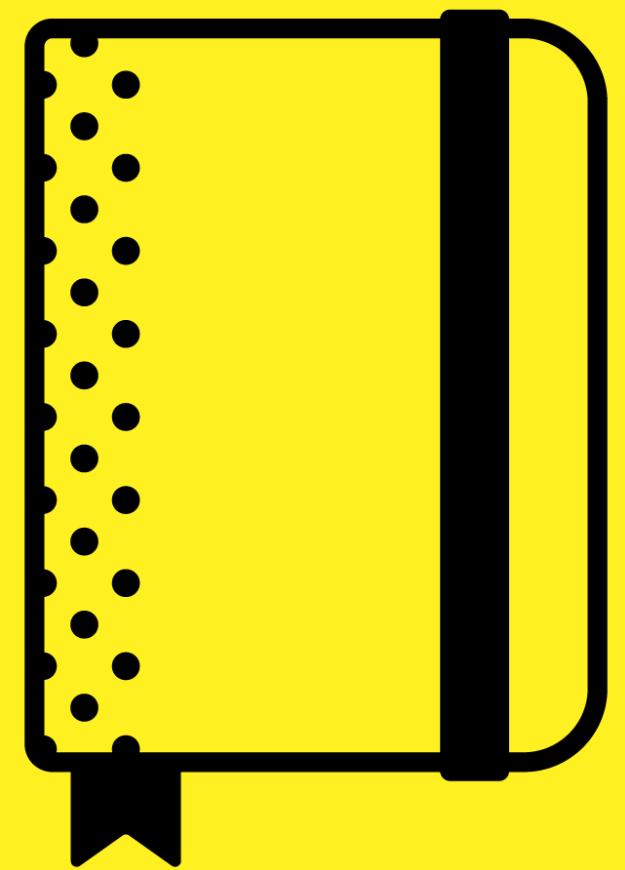
- Frontend is een goede optie voor features als filteren en sorteren
- Backend is een goede optie voor features als accounts aanmaken of inloggen





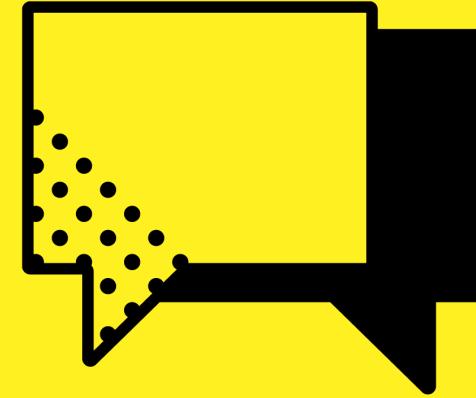
Pauze





Theorie

Node & NPM

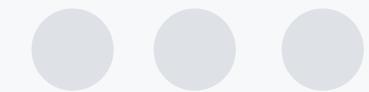


Node.js



- Is een open-source platform om JavaScript te draaien op een webserver
- Door het schrijven van JavaScript code kun je bepalen hoe je webserver reageert op verschillende HTTP requests
- En kunnen HTML pagina's dynamisch worden samengesteld, voordat ze in de response naar de browser worden gestuurd

Node programma



~/index.js

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

Draaien via Terminal

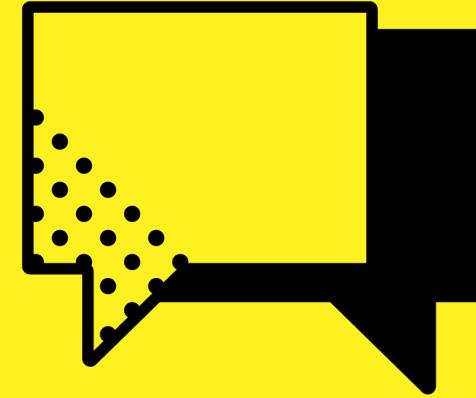


bash

```
[tilde] $ node index.js
```

```
Hello world!
```

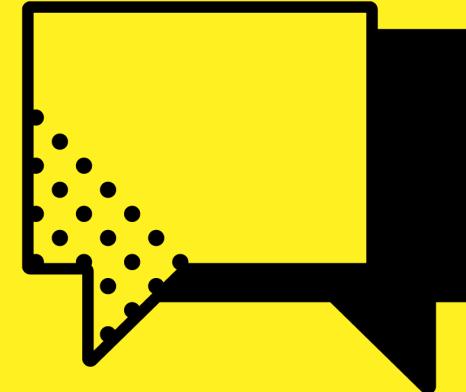
```
[tilde] $
```



Waarom node leren?



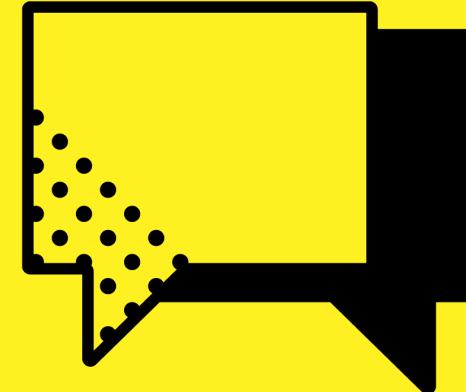
- Als je de basis kent, werk je makkelijker samen met de backenders in je team
- Je vergroot je begrip van webtechnologie
- Ook als frontender werk je vaak met tools op basis van node.js en npm
- En het is een opstapje om wel richting backend te groeien



NPM



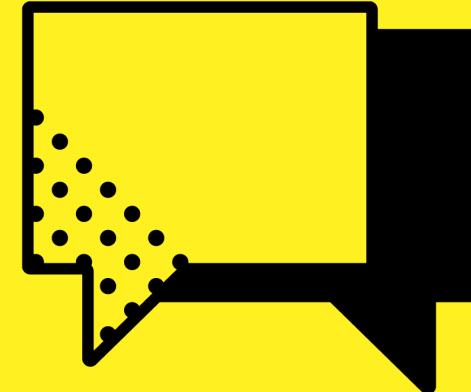
- NPM is de Node.js package manager
- Enorm veel nuttige functionaliteit is al door anderen gebouwd en als module beschikbaar gesteld
- Met NPM kun je deze modules installeren en beheren
- NPM werkt vanaf de command line



Een paar NPM commando's



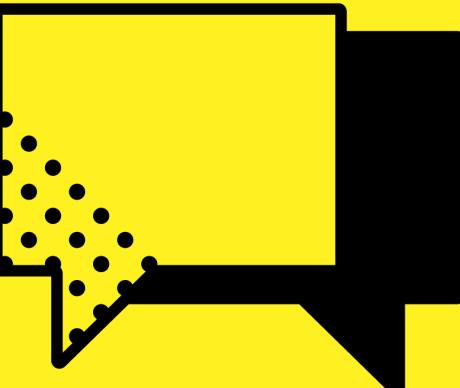
- npm `init` – maakt een nieuw node.js project aan
- npm `install modulenaam` – installeert deze module in je project
- npm `update` – update alle gebruikte modules naar de nieuwste versie
- npm `audit` – check gebruikte modules op security issues



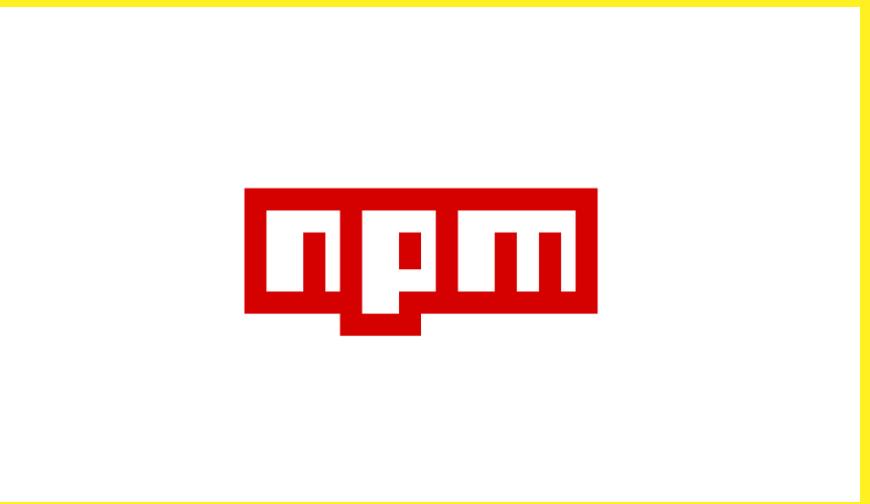
package.json



- NPM houdt info over je project, inclusief de gebruikte modules, bij in het bestand package.json
- Als je een project download vanaf GitHub, kun je daardoor met npm install in 1x alle nodige modules installeren.
- Zet zelf ook je modules niet in je repository, daarvoor hebben we al NPM! Voeg de directory node_modules toe aan je .gitignore



dependencies



npm



bash

```
$ npm install repeat-string
```

Dependencies are used in the project itself

\$

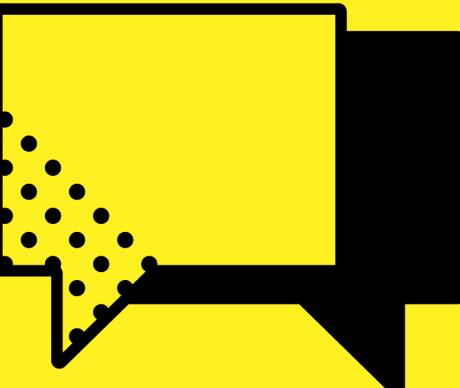


package.json

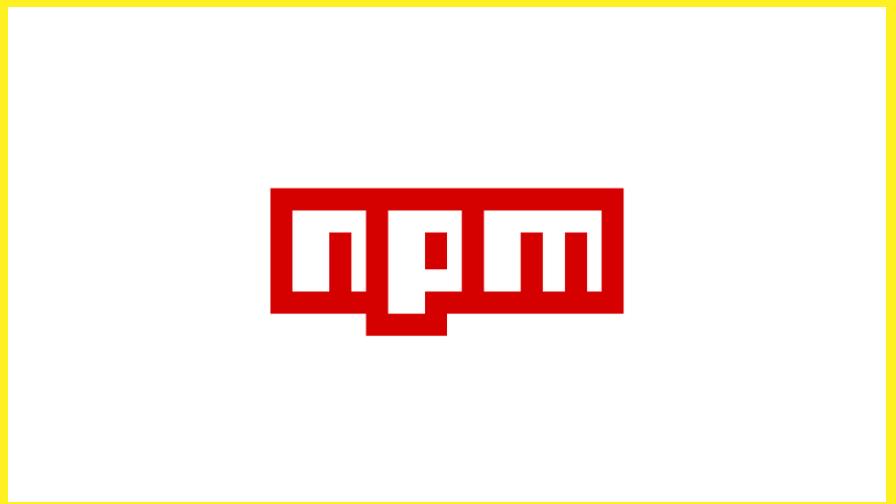
{

...

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



dependencies



npm



bash

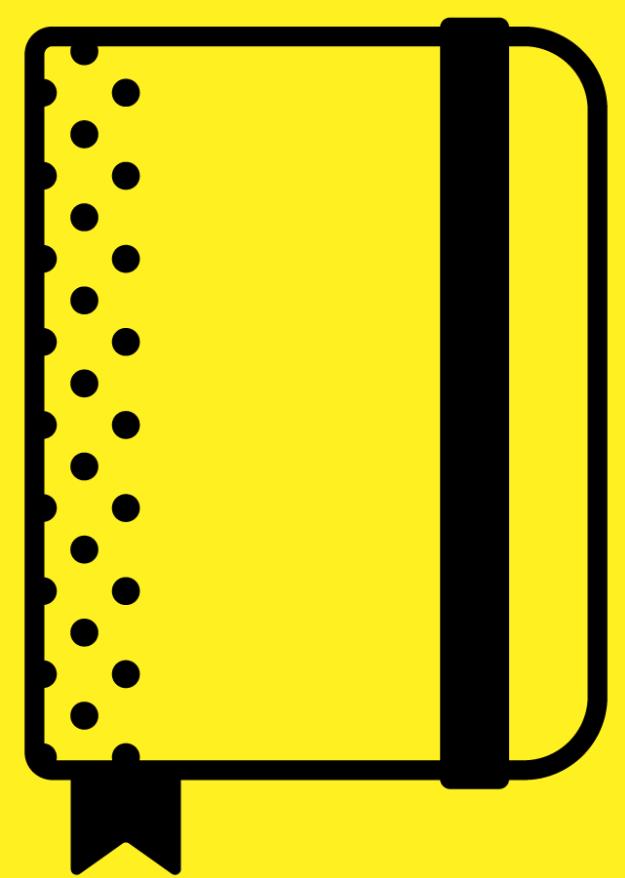
```
$ npm install tape --save-dev  
+ tape@4.8.0  
updated 1 package in 1.44s
```

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

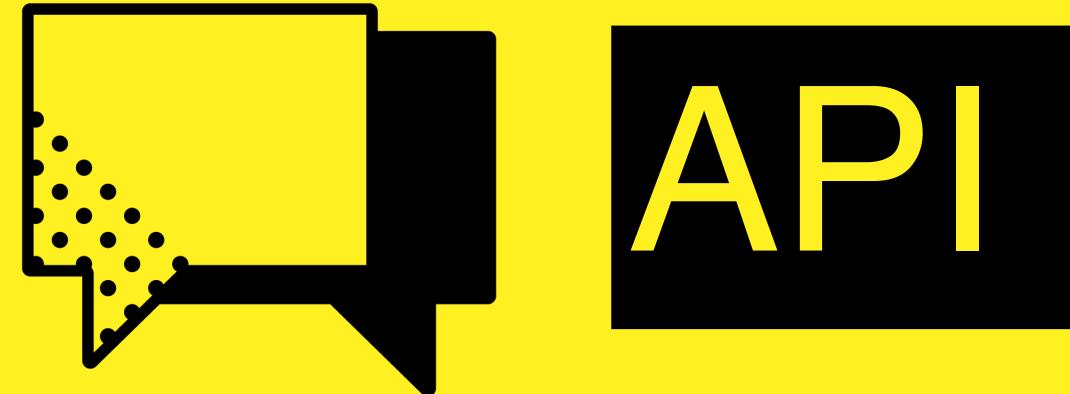


package.json

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

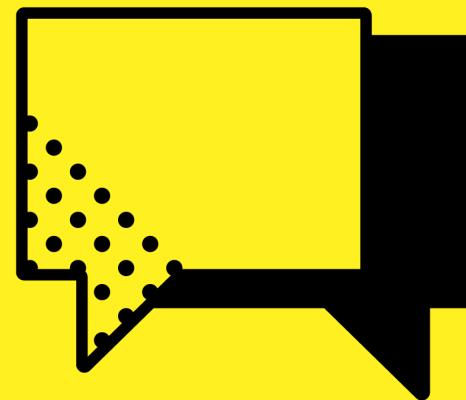


Theorie
API



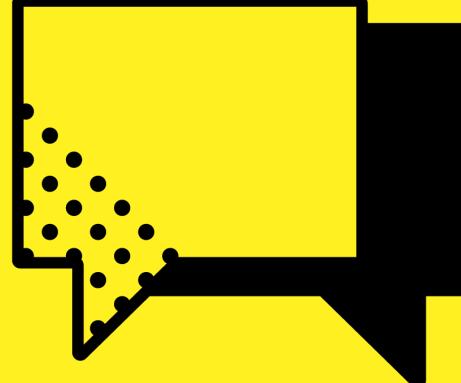
Een API – of *application programming interface* – maakt het leven van een programmeur makkelijker:

- Een API biedt toegang tot complexe systeemfuncties
- Maar wij hoeven niet te weten hoe dat precies werkt
- Wij hoeven alleen te weten hoe we met de API kunnen communiceren – de API regelt de rest
- Oftewel, via een API kunnen twee stukken software met elkaar communiceren



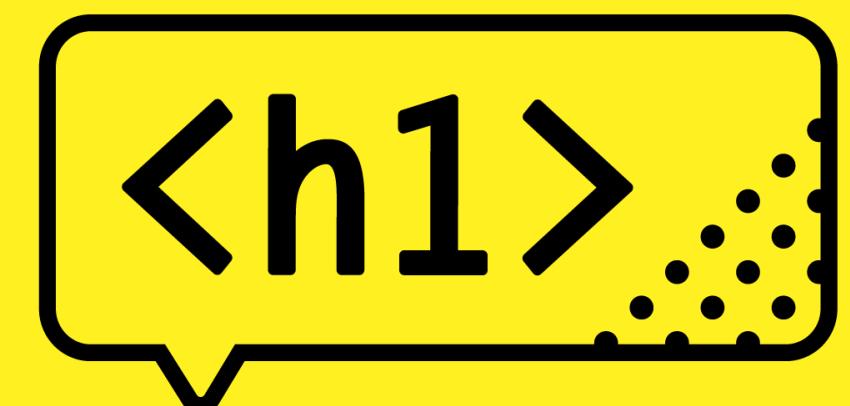
Soorten API's

- **Operating Systems API** – stelt bv functies van Windows of MacOS beschikbaar aan applicaties. (*gaan wij in dit blok niet gebruiken*)
- Browser **Web API** – zit in de browser en kan worden aangesproken door front-end code. Je kent bv de Fetch API, maar ook de DOM is een API.
- **Online Web API** – een API die draait op een webserver die online benaderbaar is via HTTP requests.
Die requests kun je zowel vanuit front-end als back-end code doen.



De Project Tech “Data API”

- Is een online Web API
- Die je als aparte webserver kunt draaien op je eigen computer
- Stelt je in staat (relatief) eenvoudig data te lezen uit een database en op te slaan middels HTTP requests
- Maar heeft minder functionaliteit dan als je zelf back-end code schrijft die rechtstreeks communiceert met de database.



Lesopdracht

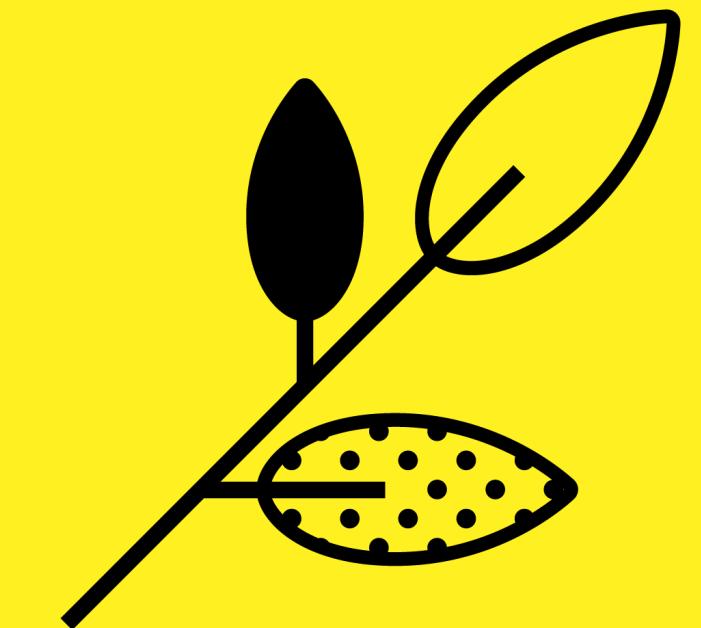
Installeren en uitproberen “Data API”

Maak de oefening op GitHub



Tip: zorg dat je wel eerst Node.js
en NPM hebt geïnstalleerd...





Einde van de les

Tot de volgende keer!

