Programmeren 6: Fullstack Webdevelopment (React & Node.js)

Versie: 2021-01-22

Knowledgebase: https://luukftf.github.io/knowledgebase
(code: https://github.com/LuukFTF/knowledgebase)
By: Lucas van der Vegt

Leerdoelen

Index

Fullstack < *server (webserver), back-end (logica en data) en front-end (HTML/JavaScript, de user interactie)* >

A. backend - Nodejs & Express & MongoDB

https://www.youtube.com/watch?v=ENrzD9HAZK4

let vs var

let is in de scope, var doet onverwachte dingen

Functions

(function via parameter)

functie in een functie meegeven

```
function helle() {
    console.log("Hello World")
}

function hello2(a) {
    a()
}

hello2(hello)

// Hello World
```

anonieme functie

```
let b = function() {
    console.log("Anonieme Functie")
}

b()

// Anonieme Functie
```

object functie

```
let j = {
    "abc" : 1,
    "xyx" : "asd",
    "f1" : hello,
    "f2" : b,
    "f3" : function() {
        console.log("Functie 3")
```

```
    }
}

j.f1()
j.f2()
j.f3()

// Hello World
// Anonieme Functie
// Functie 3
```

callback function

```
if (true) {
    j.f1()
} else {

}
```

nested functions

Arrow Function

```
let f = () => {
    console.log("Random Arrow Function")
}
```

IIFE old workaround for `var`

```
(function() {
:
:
})()
```

recursie regel 1. zorg dat het kan stoppen regel 2. zorg dat de recursie dichter bij de eindconditie kan komen

```
function recursion(n) {
    if (n == 1) {
        return 1
    }
    return n + recursion(n - 1)
}
```

beter alternatief op recursion

```
function count(n) {
    let total = 0;
    for (let i = 1; i <= n; i++ ) {
        total += 1
    }

    return total
}
```

NPM Packages

Express Mongoose Nodemon Dotenv

`/node_module` gitignore

Installing & Setup

installing software (windows)

```
winget install npm # install npm
winget install nodejs # install nodejs
winget install mongodb # install mondgodb

npm -v # check if npm is correctly installed
nodejs -v # check if nodojs is correctly installed
mongodb -v # check if mongodb is correctly installed
```

setup new project

```
git clone # clone corresponding git repo

npm install express mongoose dotenv # add express, mongoose & dotenv dependency to
project
npm install --save-dev nodemon # add nodemon dependency to project development

npm i # install repo packages
```

update

```
npm update # update all packages, respecting package versioning rules
```

run

```
cd 'C:\Program Files\MongoDB\Server\5.1\bin\'
mongod.exe

net start mongodb

npm run dev || npm start
```

.ENV

init

```
require('dotenv').config()
```

call

```
process.env.DATABASE_URL
```

`.env` file (.gitignore)

```
DATABASE_URL=mongodb://localhost/songs
PORT=8000
```

`.env.example` file

```
DATABASE_URL=mongodb://host/dbname
PORT=3000
```

Endpoint

Resources

Middleware

Models

Database

Mongoose Schema

Checker

http://checker.basboot.nl/

VPS: api hosted.hr: webservice.json

CORS

Acces-Allow-Origin

B. HTTP, RESTfull API & OAuth

https://www.youtube.com/watch?v=-MTSQjw5DrM

API < *an Application Programming Interface is an interface that defines interactions between multiple software applications or mixed hardware-software intermediaries.* >

idempotency < *Idempotence is the property of certain operations in mathematics and computer science whereby they can be applied multiple times without changing the result beyond the initial application.* >

Safe Method < *Deze method veranderd niks op de server* >

Internet < *een netwerk van netwerken* >

HTTP

Hypertext Transfer Protocol

Uniform Interface

stateless

Cacheable

Software

Postman Insomnia

VScode extension: REST Client

Methods

name	function	safe	idempotent	Status Code Response
POST	create		x	201
GET	read	x	x	200
PUT	create / update (geheel)		x	200
PATCH	update (deel)			200
DELETE	destroy		x	204
OPTIONS	get possible methods	x	x	200
HEAD	get without body	x	x	200
TRACE				200
CONNECT				200

POST overloading *< sending HTTP requests that doesnt exist can be done with a POST request. You send a method with the head that specifies what custom request you are using (document this correctly)>*

Custom Methods *< another way to use create new request methods, is to use custom http methods, just excange `post` for something else like `undelete` (document this correctly)>*

URIs

< Uniform Resource Identifier > https://www.slideshare.net/landlessness/teach-a-dog-to-rest

```
protocol://userinfo@subdomain.domain.tld:port/path?query#fragment
```

```
https://api.com/v2/comet
```

network_location/resource

structuur / hierarchie (pad) nevenschikking (, 😉 leesbaarheid (- _) geen extensies (of extensies over de inhoud, liever application/json /xml (.json .xml))

routing verbergt techniek

Representatieformaten

Mensen: html

Mensen & Machines: html met microformats

Machines: json, xml, yaml, rss

JSON

```
{
    "widget": {
        "debug": "on",
        "window": {
            "title": "Sample Konfabulator Widget",
            "name": "main_window",
            "width": 500,
            "height": 500
        },
        "image": {
            "src": "Images/Sun.png",
            "name": "sun1",
            "hOffset": 250,
            "vOffset": 250,
            "alignment": "center"
        },
```

```
        "text": {
            "data": "Click Here",
            "size": 36,
            "style": "bold",
            "name": "text1",
            "hOffset": 250,
            "vOffset": 100,
            "alignment": "center",
            "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
        }
    }
}
```

XML

```
<widget>
    <debug>on</debug>
    <window title="Sample Konfabulator Widget">
        <name>main_window</name>
        <width>500</width>
        <height>500</height>
    </window>
    <image src="Images/Sun.png" name="sun1">
        <hOffset>250</hOffset>
        <vOffset>250</vOffset>
        <alignment>center</alignment>
    </image>
    <text data="Click Here" size="36" style="bold">
        <name>text1</name>
        <hOffset>250</hOffset>
        <vOffset>100</vOffset>
        <alignment>center</alignment>
        <onMouseUp>
            sun1.opacity = (sun1.opacity / 100) * 90;
        </onMouseUp>
    </text>
</widget>
```

YAML

```
widget:
  debug: 'on'
  window:
    title: Sample Konfabulator Widget
    name: main_window
    width: 500
    height: 500
  image:
```

```
    src: Images/Sun.png
    name: sun1
    hOffset: 250
    vOffset: 250
    alignment: center
  text:
    data: Click Here
    size: 36
    style: bold
    name: text1
    hOffset: 250
    vOffset: 100
    alignment: center
    onMouseUp: sun1.opacity = (sun1.opacity / 100) * 90;
```

Request

```
POST https://api.com/v2/comet HTTP/1.1
Accept: application/json
Authorization: <token>
Connection: keep-alive

{
    "body": "body"
}
```

VERB / resource uri / protocol

Response

```
HTTP/1.1 200 OK
Age: 2323
Connection: keep-alive

{
    "id": "2"
    "status": "3"
}
```

protocol / statuscode

Basic Networking

IP/TCP & OSI model

Packets (information)

7 layers

L7 - Application {L6 - Presentation} {L5 - Session}

L4 - Transport - protocol / ports (TCP & UDP + https:443, http:80, ssh:22, ftp) L3 - Network - ip adressen L2 - Data Link - mac addressen L1 - Physical - ethernet ports

Statuscodes

Uniform Interface kan errors afhandelen HTTP Errors

2XX good

200 - OK 201 - Created 204 - No Content

3XX recoverable error

300 - Multiple Choices 302 - Found (redirect) 304 - Not Modified

4XX client error

400 - Bad Request 401 - Unauthorized 403 - Forbidden 404 - Not found

5XX server error

500 - Internal Server Error 501 - Not Implemented 503 - Service Unavailable

RESTfull API

RESTful < *Representational State Transfer, invented by Roy Fielding in 2000* >

JSON

Restfull Contstraints

1. Client gescheiden van server < *cliënt weet niks van de interne server werking en andersom* >
2. Stateless < *Server houdt niks bij van de state van de cliënt bijv. inloggegevens* >
3. Cacheable < *Instellen wat wel/niet moet worden gecached om bandbreedte te besparen* >
4. Uniforme interface < *Een vaste manier waarop de communicatie verloopt* >

API Documentation

OpenAPI Specification https://swagger.io/specification/

Postman

Linking (HATEOAS)

< *Hypermedia as the Engine of Application State* > https://en.wikipedia.org/wiki/HATEOAS

HAL

https://en.wikipedia.org/wiki/Hypertext_Application_Language

```
{
    "_links": {
        "self": { "href" : "http://api....." },
        "collection": { "href" : "http://api......" }
    }
}
```

link relation types

self collection alternate edit related previous & next first & last

Pagination

start (begin bij 1) limit (aantal)

```
GET /items?start=6&limit=5
Accept: application/json
```

pagina 6 tot en met 10

```
{
    "pagination": {
        "currentPage": 2,
        "currentItems": 5,
        "totalPages": 2,
        "totalItems": 10,
        "links": {
            "first": {
                "page": 1,
                "href":"/items?start=1&limit=5"
            },
            "last": {
                "page": 2,
                "href":"/items?start=6&limit=5"
            },
            "previous": {
                "page": 1,
                "href":"/items?start=1&limit=5"
            },
            "next": {
                "page": 2,
                "href":"/items?start=6&limit=5"
            },
        }
    }
}
```

Response Categories

items links pagination

```
GET /items/
Accept: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "items": [
        {
            "id": "200",
            "title": "test",
            "links": {
                "self": "/items/200"
            }
        },
        {
            "id": "201",
            "title": "test",
            "links": {
                "self": "/items/201"
        },
    ],
    "links": {
        "self": "/items/"
    },
    "pagination": {
        "currentPage": 1,
        "currentItems": 4,
        "totalPages": 1,
        "totalItems": 4,
        "links": {
            "first": {
                "page": 1,
                "href":"/items/"
            },
            "last": {
                "page": 1,
                "href":"/items/"
            },
            "previous": {
                "page": 1,
                "href":"/items/"
            },
            "next": {
```

```
                "page": 1,
                "href":"/items/"
            },
        }
    }
}
```

Type RESTFULL Resources

REST Resource < *Het type resource wat in de body teruggestuurd word* >

Detail < *all data of specific item* >

```
GET https://pokeapi.co/api/v2/pokemon/4
Accept: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{

    "id": 4,
    "name": "charmander",
    "base_experience": 62,
    "height": 6,
    "weight": 85,
    "location_area_encounters": "https://pokeapi.co/api/v2/pokemon/4/encounters",
    "stats": [
        {
            "base_stat": 39,
            "stat": {
                "name": "hp",
                "url": "https://pokeapi.co/api/v2/stat/1/"
            }
        },
        {
            "base_stat": 52,
            "stat": {
                "name": "attack",
                "url": "https://pokeapi.co/api/v2/stat/2/"
            }
        },
    ],
}
```

GET, PUT/PATCH, DELETE, OPTIONS

Collection < *list of items with indexable information*>

```
GET https://pokeapi.co/api/v2/pokemon
Accept: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "results": [
        {
            "name": "bulbasaur",
            "url": "https://pokeapi.co/api/v2/pokemon/1/"
        },
        {
            "name": "ivysaur",
            "url": "https://pokeapi.co/api/v2/pokemon/2/"
        },
        {
            "name": "venusaur",
            "url": "https://pokeapi.co/api/v2/pokemon/3/"
        },
        {
            "name": "charmander",
            "url": "https://pokeapi.co/api/v2/pokemon/4/"
        },
        {
            "name": "charmeleon",
            "url": "https://pokeapi.co/api/v2/pokemon/5/"
        },
        {
            "name": "charizard",
            "url": "https://pokeapi.co/api/v2/pokemon/6/"
        },
        {
            "name": "squirtle",
            "url": "https://pokeapi.co/api/v2/pokemon/7/"
        }
    ]
    "pagination": {
        "currentPage": 1,
        "currentItems": 7,
        "totalPages": 160,
        "count": 1118,
    }
}
```

GET, POST, OPTIONS

Composition < *a combination of diffrent types of resources*>

```
GET https://pokeapi.co/api/v2/location/2/
Accept: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 2,
    "name": "eterna-city",
    "region": {
        "name": "sinnoh",
        "url": "https://pokeapi.co/api/v2/region/4/"
    }
    "areas": [
        {
            "name": "eterna-city-area",
            "url": "https://pokeapi.co/api/v2/location-area/2/"
        },
        {
            "name": "eterna-city-west-gate",
            "url": "https://pokeapi.co/api/v2/location-area/788/"
        }
    ],
    "game_indices": [
        {
            "game_index": 9,
            "generation": {
                "name": "generation-iv",
                "url": "https://pokeapi.co/api/v2/generation/4/"
            }
        }
    ],

}
```

GET, OPTIONS, (PUT/PATCH)

Function < *functional resource, custom input generates custom output* `/distance/rdam;adam` >

```
GET https://pokeapi.co/api/v2/battle/4;6
Accept: application/json
```

```
// calcutation who wins the battle with stats and winner
```

GET, OPTIONS

Controller < *special function* >

```
GET https://pokeapi.co/api/v2/hit
Accept: application/json
```

```
// calcutation if a pokemon is still alive after a hit, amount left & stats
```

POST, OPTIONS

OPTIONS < *see what html methods are pssible on this link*>

```
OPTIONS https://pokeapi.co/api/v2/pokemon/4/
Accept: application/json
```

```
// insert options response
GET, PUT/PATCH, DELETE, OPTIONS
```

Queries

Filter

/pokemon?type=fire

CORS headers

Geen acces control voor "normale", cross origin browser requests:

- GET of POST
- Geen bijzondere headers (zoals Authentication)
- Geen custom headers (zoals x-requested-with)
- Geen custom header values (zoals Accept: application/json)
- Content-type van request één van
- application/x-www-form-urlencoded
- multipart/form-data
- text/plain

General

vb. res.header("Acces-Control-Allow-Origin", "*"); res.header("Acces-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept")

Preflight Request

Options

Json Web Token

Base64

OAuth

Toegang tot de gegevens, maar niet tot het wachtwoord Gebruiker logt in bij andere partij, en geeft een applicatie toegang Gebruiker kan toegang te allen tijde intrekken

OAuth 2.0

De gebruiker vraagt om in te loggen bij de applicatie. Je gaat dan meteen door naar de authorisatie server en die valideert of dat mag. Als het mag ga je terug naar de gebruiker en die zet z'n gegevens neer om in te loggen. De authorisatie server kijkt dan of die gegevens correct zijn en valideert ze. Daarna stuurt de autorisatie server een tijdelijke code naar de applicatie zodat de applicatie een access token kan aanvragen. Je gaat dan naar de resource server voor het aanvragen van de access token. Zowel de autorisatie server en de resource server moeten dan nog een keer het access token valideren. En als die klopt dan laat de applicatie de pagina zien aan de gebruiker

Resource owner dat is de gebruiker **Client** dat is je applicatie **Resource server** die heeft de resource gegevens (twitter, facebook,...) **Authorization server** die beheert OAuth voor de resource (twitter, facebook,...)

Grant Types

- Authorization code (back-end)
- Implicit (deels front-end)
- Client credential (app is resource owner)
- Resource owner password credential

https://www.c-sharpcorner.com/article/understanding-workflow-of-oauth2-0-authorization-grant-types/

Documentatie

- Resources
    - Beschrijving van de resource
    - URI
    - Volledige beschrijving van alle velden
- Representatie-formaten (xml, json)
- Welke methoden (PUT, POST etc.)
    - Welke filters zijn mogelijk (?)
    - Bij POST: beschrijf ook de resource die je stuurt
    - Welke headers (request en response)
    - Welke statusmeldingen, en wat betekenen ze
    - Is authenticatie nodig?
        - Beschrijf hoe er geauthenticeerd kan worden
        - Hoe kan je credentials krijgen

Versiebeheer

C. operations - VPS & Linux

Virtual Private Server (VPS)

HTTP en HTTPS samen

kan niet

Basic Networking / VPS commands

```
ssh username@ip
```

Linux

https://cheatography.com/davechild/cheat-sheets/linux-command-line/

Basic BASH commands

```
pwd # Show current directory
mkdir [dir] # Make directory
cd [dir] # Change directory to dir
cd .. # Go up a directory
cd / # go to root dir
cd ~ # go to home dir
cd - # go to previous dir
ls # List files
df -h # show disks
du -h # show disk usage for a dir
-a # Show all (including hidden)
-t # Sort by last modified
-S # Sort by file size
cp [dir || file] [new dir || file] # copy
mv [dir || file] [new dir || file] # move
rm [dir || file] # remove
touch [name] # new file


top # show live processes
ps # process snapshot
kill [pid] # kill process
uptime # Show uptime
uname -a # Show system and kernel
whoami # Show your username
[tool] -v # show if tool is installen and which version
[tool] help || -h || --help || man # manuals and information
whereis || where [tool] # find location of installed tool
clear # clear screen
```

```
CTRL-C # Stop dcurrent running command
```

?how to clean cache & logs

Installing Backend MERN

installing software (linux)

```
sudo apt update # update apt packages
sudo apt install npm # install npm
sudo apt install nodejs # install nodejs
sudo apt install mongodb # install mondgodb

npm -v # check if npm is correctly installed
nodejs -v # check if nodojs is correctly installed
mongodb -v # check if mongodb is correctly installed
```

installing project on server / locally

```
git clone / pull [repo] # clone or pull repository

cd [dir] # change directory to repo dir

npm i # install repo packages
node . # start node index.js

sudo systemctl start mongodb # start mongodb server

sudo systemctl status mongodb # check status of mongodb server
```

configuration

```
mongo --eval 'db.runCommand({ connectionStatus: 1 })' # diagnostic mongo command

sudo systemctl stop mongodb # stop mongodb server
sudo systemctl restart mongodb # restand mongodb server

sudo ufw status # check firewall status

sudo nano /etc/mongodb.conf # edit mongodb config
```

Screen

```

```

```
sudo apt install screen

screen
screen -r
```

`ctrl+a d`

Installing Frontend MERN

File Rights

https://www.linux.com/training-tutorials/understanding-linux-file-permissions/

Als je geen rechten hebt kun je met “is l’ kijken wie welke rechten heeft. Dan kun je met het command “chmod” deze rechten aanpassen. Je hebt de opties “r, w, x”. Read, Write, Excecute. Je moet dan zorgen dat je of owner wordt, of dat je alle rechten krijgt

Read Write eXecute RWX

list with rights `ls -l`

```
drwxrwxr-x 3 ubuntu-user ubuntu-group   4096 Nev 23 10:59 helloworld
drwx------ 2 root        dialout        4096 Dec 3  13:54 test
```

Rights Owner Group Other

Right codes Owner, Group, Other

-=======---===

d: directory r: read w: write x: execute

change right modus `chmod` change owner `chown`

Directories

var/www/

D. frontend - React

javascript framework

reactjs.org

1. General

1.1 History

Facebook

De facebook website werd te complex om met traditionele webdesign technieken te bouwen.

React

Facebook bedacht React in 2013 om beter om te gaan met grote hoeveelheid data die door de app "stroomt".

FLOW / Typescript

Facebook bedacht "FLOW" om een betere ontwikkelomgeving voor Javascript te bouwen.

1.2 Frontend Frameworks

```
React
Angular
Vue

Svelte
Gatsby (CMS)
Stencil
Preact
React Native
```

1.3 Wanneer gebruik je react en wanneer niet?

Statische Website (Onepager / Papier) Statische tekst en afbeeldingen in een html pagina. (Geen react nodig)

Web Applicatie

- Complexe logica
- Complexe interactie
- Veel gebruikersdata

1.4 React Native

React native voor native (mobile) apps

1.5 The 3 Modern Frontend Framework Concepts

Single Page Application < *Een React app bestaat uit 1 enkele HTML pagina. De pagina bevat een Javascript Applicatie, geschreven in React.* >

Components < *Geïsoleerde componenten* >

Databinding < *Data oriented, React kan automatisch de DOM updaten zodra je een variabele aanpast. (Reactive)* >

1.6 Native In Web

Webcomponents

Modules

1.7 React Example (pseudo code):

app.js

```
<body>
    <Navigation />
    <Shop />
<body/>
```

Navigation.js

```
<div>
    <button>home</button>
    <button>about us</button>
    <button>shop</button>
</div>
```

shop.js

```
<div>
    <Product />
    <Product />
    <Product />
</div>
```

2. PRG04 vs PRG06

Flashback naar OOP in PRG04

```
class Car extends Vehicle {
    constructor() {
        super()
    }
    public drive() {
        console.log("Vrooom")
    }
}
```

No DOM Manipulation

In je React code staat geen rechtstreekse DOM manipulation meer!

Oude Methode:

shop.html

```
<div>
    <p>winkelwagen</p>
    <div id="items">1</div>
    <button id="button">Buy Item</button>
</div>

<script src="shop.js"></script>
```

shop.js

```
let cart = document.querySelector("#items")
let btn = document.querySelector("#button")
btn.addEventListener("click", ()=>buyItem())

let items = 1

function buyItem(){
    items++
    cart.innerHTML = `Winkelwagen: ${items}`
}
```

Functional vs OOP programming

function

hooks function

functions zijn kleiner

class

standaard

uitgebreider (constructor)

geen public of private > alles is private

3. Setup

Installeer NodeJS. Maak een project met `npm init -y` Installeer parcel. Installeer react

```
npm init -y
npm install --save-dev parcel

npm install react react-dom
```

Typescript & SASS

Typescript Hernoem je .js modules naar .ts modules. Installeer type information.

```
npm install @types/react @types/react-dom --dev
```

Sass

```
npm install node-sass
```

Directory Structure

- /public || /docs
- /src
    - App.js
    - index.js
    - style.css
- /node_modules [GITIGNORE]
- package.json

NPM Script

Het is handig om Parcel's development en build commandos in je package.json te plaatsen

```
{
    :
    "scripts": {
        "start": "parcel src/index.html",
        "build": "parcel build src/index.html --dist-dir docs --public-url ./"
```

```
    },
    :
}
```

Start live dev mode

```
npm run start
```

Build

```
npm run build
```

Package.json

package.json bevat informatie over alle libraries die je gebruikt. Deze libraries staan in de `node_modules` map.

`package.json`

```
{
    "name": "react app",
    "version": "1.0.0",
    "description": "",
    "scripts": {
        "start": "parcel src/index.html",
        "build": "parcel build src/index.html --dist-dir docs --public-url ./"
    },
    "repository": {
    "type": "git",
    "url": "git+https://github.com/..."
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "bugs": {
        "url": "https://github.com/LuukFTF/prg06-frontend/issues"
    },
    "devDependencies": {
    "parcel": "^2.0.1"
    },
    "dependencies": {
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "node-sass": "^7.0.0"
    }
}
```

Github

Gitignore

Het heeft geen nut om alle libraries naar je GitHub repo te uploaden. Maak daarom een .gitignore file.

`.gitignore`

```
.env

node_modules

.parcel-cache

.DS_Store
.vscode
```

Als iemand jouw GitHub repo uit checkt, kan hij/zij zelf de node_modules map aanmaken.

```
npm install
```

Github Pages

Verander de output map van je project naar docs via package.json

Na het build commando heb je nu een docs map waarin je project staat.

Je kan nu je hele project pushen naar GitHub. Activeer Github Pages en kies de main branch, docsfolder voor de live output!

```
    ⋮
    "start": "parcel src/index.html",
    "build": "parcel build src/index.html --dist-dir docs --public-url ./"
    ⋮
```

Templates

React Template vanilla Functionl

https://parceljs.org/recipes/react/

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
```

```
    <title>PRG06-Frontend</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="index.js"></script>
  </body>
</html>
```

index.js

```
import React from "react";
import ReactDOM from "react-dom";

import { App } from "./App";

ReactDOM.render(<App />, document.getElementById("root"))
```

App.js

```
import React from "react";
import "./style.css";

export function App() {
    render() {
        return(
            <div className="app">

            </div>
        );
    }
}
```

React Template vanilla Object Oriented

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>PRG06-Frontend</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="index.js"></script>
  </body>
</html>
```

index.js

```
import React from "react";
import ReactDOM from "react-dom";

import { App } from "./App";

ReactDOM.render(<App />, document.getElementById("root"))
```

App.js

```
import React from "react";
import "./style.css";

export class App extends React.Component {
    render() {
        return(
            <div className="app">

            </div>
        );
    }
}
```

Strictmode

4. Tools

Module Bundler

Parcel rollopjs webpack Create React App

Parcel

Parcel: https://parceljs.org

Start de live development server in watch mode. Openhttp://localhost:1234

```
npx parcel src/index.html
```

Als je project af is maak je de final build. Open deze in localhost of upload naar je server.

```
parcel build src/index.html
```

Broswer Addon

React Developer Tools

5. Modules

CommonJS vs ES6 modules

In NodeJS heb je met CommonJS modules gewerkt

CommonJS

```
const express = require('express');
const myapp = require('./app.js');

:
```

In React (en Node 17) werk je met ES6 modules

ES6

```
export default function App() {
}
```

```
import App from "./App.js"

:
```

or

```
export function App() {
}
```

```
import { App } from "./App.js"

:
```

6. Components

Een React App is opgebouwd uit geïsoleerde components.

Een component bevat Javascript en HTML (JSX)

Composition App HAS a shop Shop HAS products

Inheritance (fixed structure) Shop EXTENDS app

Components Updates (functional)

Effect Hook

https://reactjs.org/docs/hooks-effect.html

7. Databinding

Een component haalt JSON data van een API.

De HTML wordt niet herladen. Alleen de DOM elementen die de data tonen worden aangepast.

Variabelen in een component zijn verbonden aan de view van het component. Als de variabele verandert, verandert de view automatisch mee.

```
let items = 1

function buyItem(){
    items++
}

function render() {
    <div>
        <p>winkelwagen</p>
        <p>{ items }</p>
        <button onClick={ buyItem() }>Buy Item</button>
    </div>
}
```

psuedocode!

State

Reactive data maak je aan middels een state variabele

State variabelen mogen alleen door de eigenaar aangepast worden.

```
:
export class Product extends React.Component {
    constructor {
        super()

        this.state = {
            name: "Canon 200D",
            description: "Een mooie canon camera",
```

```
            price: 499
        }
    }

    updatePrice() {
        this.setState((oldState) => {
            name: "6D",
            price: oldState.price + 500
        })
    }

    render() {
        return(
            <div className="product">
                <h1>{ this.state.title }</h1>
                <p>{ this.state.description }</p>
                <h2>Price: { this.state.price }</h2>
                <button onClick={ () => this.updatePrice() }>Aanbieding</button>
            </div>
        );
    }
}
```

Prop

Met Props kan je reactive data aan een childcomponent doorgeven.

Het child component kan props data tonen maar niet bewerken.

```
:
export class Product extends React.Component {

    :
    render() {
        return(
            <div className="product">
                <h1>{ this.props.title }</h1>
                <p>{ this.props.description }</p>
            </div>
        );
    }
}
```

```
:
export class Shop extends React.Component {

    :
```

```
    render() {
        return(
            <div className="shop">
                :

                <Product name="Google Chromecast"/>
                <Product name="Skullcandy Crusher"/>
                <Product name="Duracell AA Batterijen"/>

                :
            </div>
        );
    }
}
```

Een component kan de waarde van zijn props niet aanpassen!

Prop + State

```
:

export class Shop extends React.Component {
    constructor {
        super()

        this.state = {
            products: ["Canon 200D", "Google Chromecast", "Skullcandy Crusher",
"Duracell AA Batterijen"]
        }
    }
    :

    :
    render() {
        return(
            <div className="shop">

                :
                <Product name={ this.state.products[0] }/>
                <Product name={ this.state.products[1] }/>
                <Product name={ this.state.products[2] }/>
                <Product name={ this.state.products[3] }/>
                :

            </div>
        );
    }
}
```

Lifting state up

Data die in je hele app relevant is plaats je vaak in de main app.

Data Store

Gebruik bij complexe / nested flow (big scale, coolblue)

8. Event Handlers

Een child component kan event handlers in een parent aanroepen.

Dit is de manier om de state van een parent te veranderen vanuit een child.

```
export class Product extends React.Component {

        :
        render() {

            :
            <button onClick={ Shop.doSomething() }>Button</button>
            :

        }
}
```

9. Arrays

Map (Array)

loop over array ("foreach")

```
:
export class Shop extends React.Component {
    constructor {
        super()

        this.state = {
            products: ["Canon 200D", "Google Chromecast", "Skullcandy Crusher"]
        }
    }
    :

    :
    render() {
        const allProducts = this.state.products.map((prod, index) => (
            <Product key={ index } name={ prod }/>
        ))

        return(
            <div className="shop">
```

```
                :
                <div>
                    { allProducts }
                </div>
                :

            </div>
        );
    }
}
```

Get Array length

```
    :
    <h2>Total Products: { this.state.products.length } </h2>
    :
```

Add to Array

```
    :
    addProduct {
        this.setState((oldState) => {
            products: [...oldState.products, "Duracell AA Batterijen"]
        })
    }
    :
```

10. Input

Een input element heeft een `onChange` handler nodig. Deze verandert de `inputValue` zodra iemand in het form field typt.

In je overige code kan je de `inputValue` variabele gebruiken om te weten wat er is ingevuld

```
import "./styles.css";
import { useState } from "react";

export function Pokedex() {

    const [inputValue, setInputValue] = useState("");
```

```
    const onChangeHandler = (event) => {
        setInputValue(event.target.value);
    };

    return (
        <div className="Pokedex">
            <input type="text" onChange={onChangeHandler} value={inputValue} />
        </div>
    );
}
```

11. Conditional

If

Je kan && gebruiken om een element alleen te tonen als een variabele TRUE is

```
    function Pokemon(props) {

        return (
        <div>
            <h1>{ props.name }</h1>
            { props.liked && <h2>♥</h2> }
        </div>
        );
    }
```

```
function Pokemon(props) {

    return (
    <div>
        <h1>{props.name}</h1>
        { props.likes > 100 && <h2>♥♥♥</h2> }
    </div>
    );
}
```

If Else

Class

Toon een verschillende CSS style afhankelijk van of LIKED true of false is.

```
export function Pokemon() {

    const [liked, setLiked] = useState(false);
```

```
    return (
        <div className={ liked ? 'heart' : 'noheart' }>
            Poliwhirl
        </div>
    );
}
```

Component

Toon een verschillend component afhankelijk van of LOADING true of false is.

```
export function Pokemon() {

    const [loading, setLoading] = useState(true);

    return (
        <div>
            { loading ? <Spinner /> : <Details /> }
        </div>
    );
}
```

12. API

fetch (JSON laden)

Gebruik `async await` of `promises.then()` syntax om JSON te laden. Als de JSON geladen is, kan je de state aanpassen.

```
:
export function ApiDemo() {
    const [pokemon, setPokemon] = useState([]);

    const loadJson = () => {
        fetch("https://pokeapi.co/api/v2/pokemon")
            .then((response) => response.json())
            .then((data) => dataWasLoaded(data))
            .catch((error) => console.error("API fetch error"))
    }

    const dataWasLoaded = (data) => {
        console.log(data.results)
        setPokemon(data.results)
    }

    return (
        <div className="ApiDemo">
```

```
            <h4>JSON Pokemon loaded: { pokemon.length}</h4>
            <button onClick={ loadJson }>Load JSON</button>
        </div>
    )
}
```

13. Images

Gebruik import om images te laden.

```
import logoImage from './images/logo.png';
```

```
<img src={ logoImage } alt="logo"/>
```

14. Styling

https://reactjs.org/docs/faq-styling.html

https://parceljs.org/recipes/react/#styling

XX. React Full Basic Example Vanilla

https://reactjs.org/docs/hooks-effect.html

index.js

```
import React from "react";
import ReactDOM from "react-dom";

import { App } from "./App";

ReactDOM.render(<App />, document.getElementById("root"))
```

App.js

```
import React from "react";
import "./style.css";
import { Shop } from "./Shop.js"

export class App extends React.Component {

    constructor() {
        super()
        console.log("Created the app")
        this.doSomethig()
```

```
    }

    doSomething() {
        console.log("Doing something!")
    }

    render() {
        return(
            <div className="app">
                <h1>Title</h1>
                <p>Hello World!</p>

                <Shop />
                <Shop />
            </div>
        );
    }
}
```

Shop.js

```
import React from "react";
import "./style.css";

export class Shop extends React.Component {
    render() {
        return(
            <div className="shop">
                <h1>SHOP</h1>
                <p>This is a shop</p>

            </div>
        );
    }

}
```

style.css

```
.body {
    background-color: lightgrey;
}

.shop {
    background-color: white;
    margin: 20px;
    padding: 20px
}
```

Lifecycle

Alle statussen die je doorloopt. Deze breid je normaal uit door je classes uit te breiden en 'hook-methods' te implementeren. Daar plaats je code in waarvan je wilt dat hij op dat moment uitgevoerd wordt. useEffect en useState zijn hier goede voorbeelden van. Bijvoorbeeld de functie `useEffect () => {}` hier zet je alles in waarvan je wilt dat het wordt uitgevoerd aan het begin van de pagina. Dus zodra de pagina geladen wordt.

E. frontend - Sass

styleguides http://styleguides.io/
https://web.archive.org/web/20170523012226/http://codepen.io/guide/#one

CSS

Grid

```
.shop {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    grid-auto-rows: max-content;
    grid-column-gap: 10px;
    grid-row-gap: 10px;
}
```

```
<div className="shop">
    <div>Item</div>
    <div>Item</div>
</div>
```

Flex

```
.shop {
    display: flex;
}
```

```
<div className="shop">
    <div>Item</div>
    <div>Item</div>
</div>
```

Installing

Installing To React

Installing Generally NPM

Links

https://www.youtube.com/watch?v=fgTGADljAeg

`end of file`
modified date: 2022-01-22
publish date: 2022-01-22