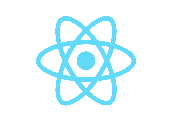
React

# React app aanmaken

C:\Users\Joren> create-react-app naam-van-app

# App runnen

C:\Users\Joren\naam-van-app> npm start

# Bootstrap in applicatie instalen

C:\Users\Joren\naam-van-app> npm i bootstrap

## In index.js toevoegen:

import ‘bootstrap/dist/css/bootstrap.css’;

# Component toevoegen

1. Onders src nieuwe map “components” maken
2. Onder src/components nieuw bestand maken: naamVanComponent.jsx

## In naamVanComponent.jsx

1. imrc tap (import react / component)
2. cc tap (create class)
3. geef naam aan class (bijvoorbeeld NaamVanComponent)

## Voeg toe aan index.js

import NaamVanComponent from ‘./components/naamVanComponent’;

ReactDom.render(<NaamVanComponent />, document.getElementById(“root”));

# 2 elementen meegeven aan component

class Counter extends Component{

render(){

return <React.Fragment><h1>Hey</h1><button>Knop</button> </React.Fragment>

}

}

!!! ALS WORDT GESAVED, WORDT TEKST HERVORMT TOT MEERDERE LIJNEN EN MOOIERE TEKST!!!

# Dynamisch tonen van value (in component)

Tussen { } kunnen alle expressies worden weergegeven

## Voorbeeld met statische tekst

class Counter extends Component{

state = {

count: 0

};

render(){

Return (

<React.Fragment>

<span>{this.state.count}</span>

<button>Knop</button>

</React.Fragment>

);

}

}

## Voorbeeld met wiskundige formule

class Counter extends Component{

render(){

Return (

<React.Fragment>

<span>{1 + 1}</span>

<button>Knop</button>

</React.Fragment>

);

}

}

## Voorbeeld met oproepen van functie die waarde terug geeft

class Counter extends Component{

state = {

count: 0

};

render(){

Return (

<React.Fragment>

<span>{this.formatCount()}</span>

<button>Knop</button>

</React.Fragment>

);

}

formatCount() {

const { count } = this.state;

return count === 0 ? ‘Zero’ : count;

}

# Setting Attributes

## Foto dynamisch instellen

class Counter extends Component{

state = {

imageUrl: ‘https://picsum.photos/200’

};

render(){

Return (

<div>

<img src={this.state.imageUrl} alt=””/>

</div>

);

}

}

## Bootstrap styling gebruiken/class gebruiken

class Counter extends Component{

render(){

Return (

<div>

<span className=”badge badge-primary”>zero</span>

</div>

);

}

}

***Tip: className=”badge badge-primary m-2” 🡺 geeft een margin van 2***

## Styles aanpassen optie 1

class Counter extends Component{

styles = {

fontSize: 50,

fontWeight: “bold”

};

render(){

Return (

<div>

<span style={this.styles} className=”badge badge-

primary”>zero</span>

</div>

);

}

}

## Styles aanpassen optie 2

class Counter extends Component{

render(){

Return (

<div>

<span style={{ fontSize: 30 }} className=”badge badge-

primary”>zero</span>

</div>

);

}

}

<https://www.youtube.com/watch?v=Ke90Tje7VS0>

# rendering classes dynamically

class Counter extends Component{

state = {

count: 0

}

render(){

Return (

<div>

<span style={{ fontSize: 30 }} className=”this.getBadgeClasses”> {count} </span>

</div>

);

}

getBadgeClasses() {

let classes = “badge m-2 badge-”;

classes += (this.state.count ===0) ? “warning”: “primary”;

return classes;

}

}

# Rendering lists

class Counter extends Component{

state = {

tags: [‘tag1’, ‘tag2’, ‘tag3’]

}

render(){

Return (

<div>

<ul>

{ this.state.tags.map(tag => <li key={tag}>

{tag} </li>) }

</ul

</div>

);

}

}

# Conditional rendering

## Optie 1 (if else)

class Counter extends Component{

state = {

tags: [‘tag1’, ‘tag2’, ‘tag3’]

}

renderTags(){

if(this.state.tags.length === 0) return <p>There are no tags!</p>

return <ul> { this.state.tags.map(tag => <li key={tag}> {tag}

</li>) } </ul>

}

render(){

Return (

<div>

{ this.renderTags() }

</div>

);

}

}

## Optie 2 (enkel if)

class Counter extends Component{

state = {

tags: [‘tag1’, ‘tag2’, ‘tag3’]

}

render(){

Return (

<div>

{ this.state.tags.length === 0 && “please create a

new tag!”}

</div>

);

}

}

# Handling events

class Counter extends Component{

handleIncrement(){

console.log(‘Increment Clicked’);

}

render(){

Return (

<div>

<button onClick={this.handleIncrement}> Increment </button>

</div>

);

}

}

# Binding event handlers

class Counter extends Component{

stage = {

count:0

};

handleIncrement = () => {

this.setState({ count: this.state.count + 1 });

}

render(){

Return (

<div>

<button onClick={this.handleIncrement}> Increment </button>

</div>

);

}

}

# Passing event arguments