React

Let en var

var test = "voorbeeld";
var test = "voorbeeld2"; //verplaats gewoord de eerste test

let test = "voorbeeld";
let test = "voorbeeld2";   //SyntaxError: test is already declared

Default and namedexports

default => export default ...;
named => export const ExpresList ;

You can import default exports like this:
Default => import someNameOfYourChoice from './Components/ExpresList.js';
named => import { ExpresList } from './Components/ExpresList.js';

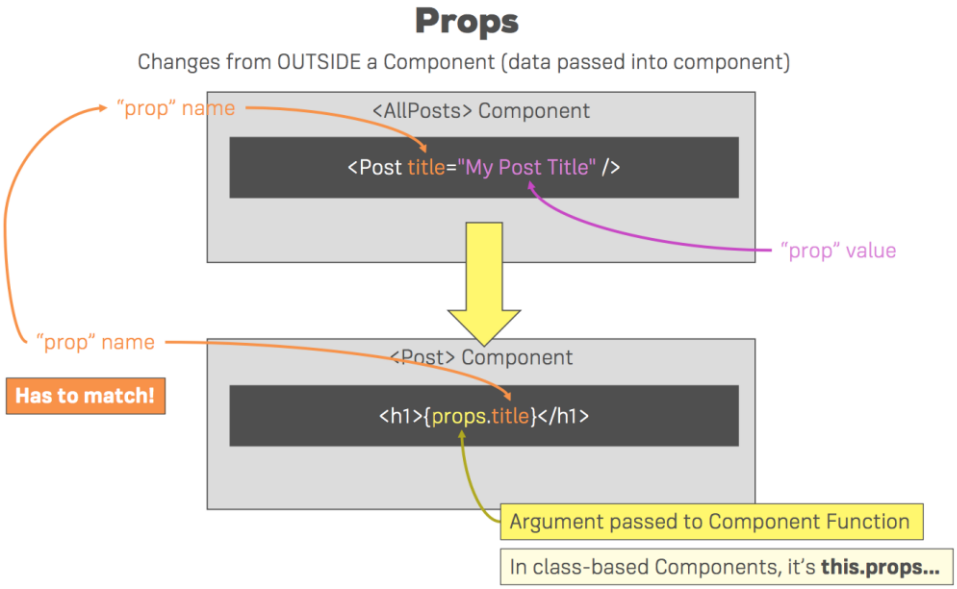
Spread Operator

const oudeArray = [1, 2, 3];
const newArray = [...oudeArray, 4, 5]; // Dit is nu[1, 2, 3, 4, 5];

Class-Components en Functional Components

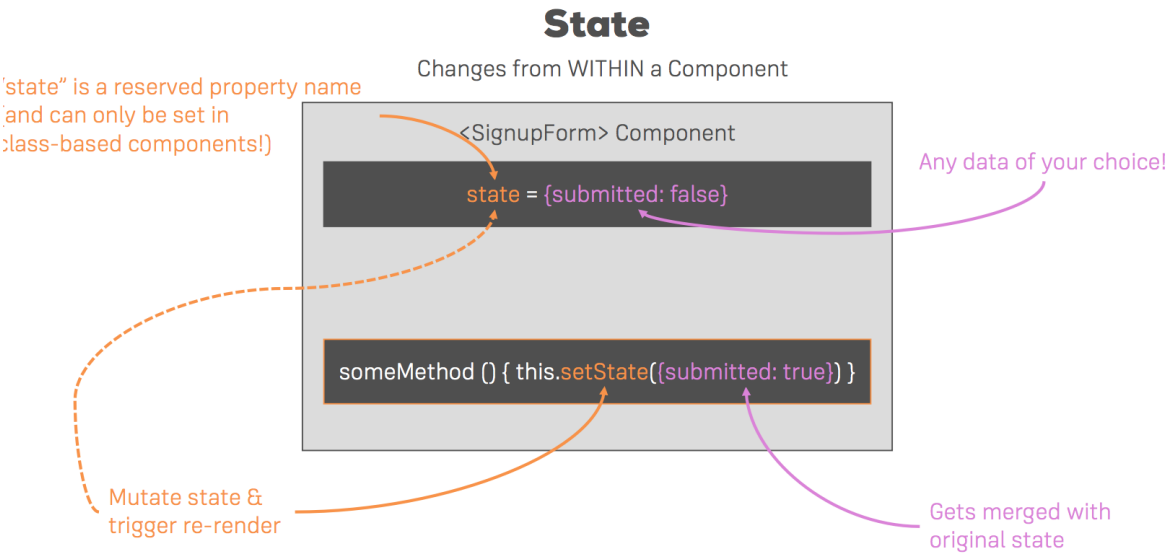
Functional components ("stateless") => const cmp = () => { return some JSX } (is sneller, omdat het niet in de Management Lifecycle zit"
Class-based components ("stateful") => class Cmp extends Component { render () { return some JSX } }

Props



laten je gegevens doorgeven van een parent component naar een child component

State



Destructued statement

```
const person = {
    name: 'Cansu',
    age: 21,
    location:{
        city:'Gent',
        temp:2
    }
};

const { name: firstname = 'Anonymous', age} = person; //destructered statement
console.log(`${firstname} is ${age}`);

const { city, temp: temperature} = person.location
if(city && temperature){
    console.log(`${city} is ${temperature} graden`);
}


const book ={
    title: 'Boek1',
    author:'Een auteur',
    publisher:{
        name: 'Maan'
    }
}

const {title, author, publisher} = book;
const {name: publisherName='Zon'} = book.publisher;

if(publisherName){
    console.log(`Title: ${title} Author: ${author} Publisher: ${publisherName}`);
}
```

Hooks

```
const App = (props) => {
  const [count, setCount] = useState(props.count);
  const [text,setText] =  useState('');

  useEffect(()=>{
    console.log('this zal 1 keer runnen');
  },[]) //zal maar 1 keer runnen, is gebruikerbaar voor bv fetch data

  useEffect(() =>{
    console.log('useffect');
    document.title = count
  },[count]) //only run this effect when count changes zal niet gebeuren dus als tekst veranderd

  return (
    <div>
      <p>The current {text || 'count'} is {count}</p>
      <button onClick={() => setCount(count + 1)}>+1</button>
      <button onClick={() => setCount(count - 1)}>-1</button>
      <button onClick={() => setCount(props.count)}>reset</button>
      <input type='text' value={text} onChange={(e) => setText(e.target.value)}/>
    </div>
  );
};

ReactDOM.render(<App count={0} />, document.getElementById("root"));
```

ComponentDidMount
Wordt uitgevoerd wanneer een class word gerenderd.

ComponentDidUpdate
Wanneer eeen update wordt uitgevoerd bij props en states.

ComponentWillUnmount
Als er een component wordt verwijdered

Webpack
Organiseerd het react applicatie en gebruikt container dependencies.

Link en NavLink
Het blijkt dat <NavLink> wordt geleverd met extra stijlkenmerken waarmee we de activeClassName link kunnen stileren en onderscheiden.

Provider en Connect voor Redux
<Provider /> stelt de store beschikbaar voor alle geneste componenten.
Connect(store)(component) functie verbindt een React-component met een Redux-store.

JEST
.toBe => Boolean, String, Number
.toEqual => Object, Array
Een fake store voor testen: npm install redux-mock-store@1.2.3

Heroku
Heroku login
Heroku create react-pop
Heroku git:remote -a react-pop
git init
git add .
git commit -m “message”
git push Heroku master
Heroku open

Firebase
Npm install firebase@4.2.0
Voor te createn naar DB. Bij actions gaan we returns functions doen, maar redux laat dit niet toe daarom gaan we een module opzetten. Npm install redux-thunk@2.0.0

Hooks

Door hooks te gebruiken kan je state in functional components gebruiken.
Usestate geeft een current en een update van de state terug.
UseEffect is het zelfde als componentDidMount en componentDidUpdate

Context

Dit is een manier dat je geen props moet door geven. Context zorgt ervoor dat de props aan elke level kan gegeven worden.

Fragments

JSX moet niet meer in zitten maar kan in lege tags <> <>

Installaties
Npm install live-server
Npm install babel-cli@6.24.1 //babel downloaden zodat compiler JSX wordt aanvaard
Npm init
Npm install babel-presets-react@6.24.1 babel-preset@1.5.2
Npm run serve
→ in src/app.js –out-file= public/scripts/app.js –presets=env,react

Installeren Webpack
Npm install webpack@3.1.0
→ aanpassen in package.json

ES6 import
Npm install validator@12.0.0
Nom install react@16.11.0 react-dom@16.11.0
Npm install babel-core@6.25.0 babel-loader@7.1.1
Npm install webpack-dev-server@2.5.1
Npm install babel-plugin-transform-class-properties@6.24.1

Installeren Model
Npm install react-model@2.2.2

SCSS
Npm install style-loader@0.18.2
Npm install css-loader@0.28.2
Npm install sass-loader@6.0.6
Npm install node-sass
Npm install normalice.ss@7.0.0

ROUTER
Npm install react-router-dom@4.2.2

REDUX
Npm install redux@3.7.2

UUID
Npm install uuid@3.1.0

HOOK
Npm install react-redux@5.0.5

MOMENT
Npm install moment@2.18.1

DATES
Npm install react-dates@12.7.0

SNAPCHOT FOR TESTS
Npm install react-addons-shallow-compare@15.6.0

JEST
Npm install jest@20.0.4
Npm test -- --watch

COMPENTNS TEST
Npm install react-test-renderer@16.0.0
Npm install enzyme@3.0.0
Npm install enzyme-adapter-react-16@1.0.0
Npm install raf@3.3.2

SECTIE19 new features
Npm install -g create-react-app@2.1.8
Create-react-app naamapp