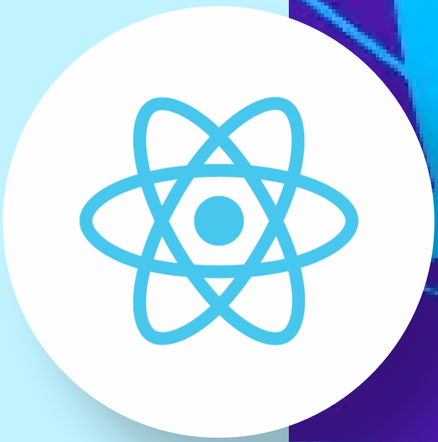
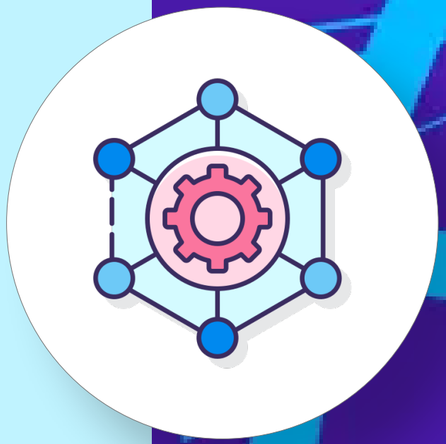
REACT basics

Les 14:

React intro, elements, components, properties





Onderwerpen les:

- React intro
- React gebruiken
- Elementen
- JSX
- Components
- Properties
- Map()
- Keys

Leerdoelen

- Aan het einde van de dag weet je wat REACT is en waar het voor wordt gebruikt
- Aan het einde van de dag kan je React components maken
- Aan het einde van de dag kun je props gebruiken
- Aan het einde van de dag kan je herbruikbare components maken en over een lijst itereren voor het aanmaken van components

React intro

Wat is React?

React is een zeer uitgebreide **JavaScript library** (bibliotheek) die developers helpt UI's (user interfaces) te maken aan de hand van kleine blokjes genoemd componenten.

React wordt vaak een JavaScript Framework genoemd zoals Angular, Vue en Ember, maar dit is niet juist. React is een library en niet een framework omdat integenstelling tot Angular of Vue je veel toegevoegde libraries nodig hebt om een grootschalige app te bouwen. React maakt het bouwen en onderhouden van userinterfaces van jouw applicatie sneller en makkelijker door ze op te breken in kleine **herbruikbare componenten.** Het helpt ook de complexiteit te versimpelen van het updaten van de DOM elements wanneer een user interactie heeft met jouw applicatie.

React intro

Wat is React?

JavaScript Library

Een JavaScript library is een bibliotheek met voorgeschreven JavaScript code dat developers helpt om makkelijker en sneller websites en applicaties te bouwen.

Met React en ook andere JavaScript frameworks kun je makkelijk **single page appliations** bouwen, waarbij alleen de inhoud wordt geladen en niet een hele nieuwe pagina.

Ontstaan React

React werd in 2011 ontwikkeld door het moederbedrijf van Facebook, Instagram en Whatsapp (Meta). Het was een intern project, maar is in 2013 publiek gemaakt. Grote namen als Netflix en Uber gebruiken React.

React intro

React gebruiken

Om gebruik te maken van de functionaliteiten die geschreven zijn in React moeten we React inladen in ons project. Dit kan op meerdere manieren.

> **https://reactjs.org/docs/add-react-to-a-website.html**

1 CDN

Via een content Delivery Network: met script tags die linken naar een locatie waar React staat.

```
<script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
```

React intro

React gebruiken

2 NPM

Via Node Package Manager. Download hiervoor Node.js en vervolgens via NPM voeg je React toe aan je project.

1. Je hebt nodig Node.js >= 14.0.0 en npm >= 5.6 op je laptop.
2. Run:

```
npx create-react-app my-app
cd my-app
npm start
```

3. Wanneer je je app wil runnen:

```
npm run build
```

React elements

Elements zijn de **kleinste bouwblokken** in React apps. Het geeft aan wat getoond moet worden in de User Interface.

Stap 1: Laad react op de pagina

We hebben toegang tot React.js nodig. In het HTML bestand voor het sluiten van de body:

```
<script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
```

React elements

Stap 2 Maak een element

Met React.createElement() maak je een React node. node betekent unit/element. Deze React node wordt een DOM element na het renderen. In JavaScript:

```
const title = React.createElement (
    'h1', // HTML tag
    { id: 'main-title', title: 'This is a title.' }, // properties
    'My first REACT element!' // content
);
```

React elements

Stap 3 Renderen element

3.1 Met ReactDOM.render() creëer en update je de DOM. In JavaScript:

```
ReactDOM.render(
  title,
  document.getElementById('root') // root, app container populair
);
```

3.2 Maak een div met de gekozen id naam bv ‘root’ in de body van HTML:

3.3 Voeg een script tag aan het HTML bestand onderaan de body met een src naar het JavaScript bestand:

```
<body>
    <div id="root">Loading...</div>

    <script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>

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

React elements

Combineren React elementen

Stap 4 maak een 2e React element

We gaan nu een omschrijving maken voor bij de titel.

```
const desc = React.createElement (
      'p', // HTML tag
      null, // geeft aan dat er geen attributen zijn
      'I just learned how to create a React node and render it into the DOM'
       // React node: node betekent een unit een React node is een unit dat een DOM node kan worden
      // DOM: document object model = Naamgeving van HTML tags. de boom structuur van de website
);
```

React elements

Combineren React elementen

Stap 5 Maak een react node die beide elementen combineert

We hebben nu een title en een desc variabele met React nodes. Deze plaatsen we in de header

```
const header = React.createElement (
    'header', // HTML tag
    null, // geeft aan dat er geen attributen zijn
    title,
    desc
);
```

React elements

Combineren React elementen

Stap 6 Pas de render aan

Nu geven we de variabele header mee aan ons element met id ‘root’

```
ReactDOM.render(
  header,
  document.getElementById('root') // root, app container populair
);
```

React elements

Attributen

In voorgaande voorbeelden hebben we gezien hoe we een attribuut id of title kunnen toevoegen. We kunnen net als in HTML elk type attribuut toevoegen.

Bijvoorbeeld class zodat we deze elementen makkelijk kunnen selecteren en stylen in onze CSS.

Stap 7 class attribuut en styling

7.1 Voeg een class toe aan het titel element. ! dit is anders dan in HTML hier heet het className

```
const title = React.createElement (
    'h1', // HTML tag
    { id: 'main-title', title: 'This is a title.'  className: 'title' }, // properties
    'My first REACT element!' // content
);
```

React elements

Attributen

7.2 Style het element in de CSS.

```
.title {
    font-size: 60px;
    color: blue;
};
```

7.3 Link naar css bestand vanuit de head in het HTML bestand.

JSX

We kunnen natuurlijk onze gehele UI bouwen door elke keer opnieuw React.createElement te schrijven, maar dit is erg omslachtig! Gelukkig is er een makkelijkere manier.

JSX is een syntax extensie voor JavaScript dat gebruikt wordt met react om elementen in de UI te omschrijven. Het lijkt heel veel op gewonen HTML, wat het een stuk makkelijker en beter te lezen maakt.

JSX

Geen JSX

```
const title = React.createElement (
    'h1',
    { id: 'main-title', title: 'This is a title.' },
    'My first REACT element!'
);
const desc = React.createElement (
      'p',
      null,
      'I just learned how to create a React node and render it into the DOM'
);
```

```
const title = <h1 id='title' title='This is a title.' className='title'>My First React Element</h1>;
const desc = <p>I just learned how to create a React node and render it.</p>;
```

JSX

Babel

Om JSX werkend te krijgen hebben we een **transformer** nodig die onze JSX omzet in JavaScript zodat de browser onze code goed kan lezen.

1. in ons HTML bestand in de body voordat we onze JavaScript laden gaan we babel-standalone inladen
2. Het type van ons eigen JavaScript bestand veranderen we naar text/babel

```
<script crossorigin src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
<script type="text/babel" src="script.js"></script>
```

- **https://babeljs.io/en/setup/**
- **https://babeljs.io/repl**

JSX

JavaScript expressions in JSX

We kunnen nu elementen maken in React met JavaScript nodes en we kunnen elementen maken met JSX! Een volgende stap is het gebruiken van dynamische content in onze elementen.

Stap 1 Maak een nieuw JavaScript bestand

Zodat we elke stap goed terug kunnen bekijken gaan we werken in een nieuw js bestand zodat ons voorgaande werk bewaard blijft. Geef het de naam jsx-expressions.js

Stap 2 Variabelen

Maak een aantal variabelen aan: desc, myTitleID, first_name.

Stap 3 Maak een React element aan en gebruik variabelen

Plaats de variabelen in een React element.

Stap 4 Gebruik een expression

Gebruik een input veld met een expression.

JSX

JavaScript expressions in JSX

```
const desc = 'I just learned how to create a React node and render it into the DOM';
const first_name = 'Maurice’ ;

const title = (
    <header id='header'>{first_name}'s First React Element</h1>
    <p>{desc}</p>
);
```

Components

Bijna alles in React bestaat uit componenten! Veel React apps hebben veel kleine componenten die geladen worden in het root/app component.

In een component worden de logica en de informatie opgeslagen. Een component bevat **elementen die gereturned worden** door het component.

Maken React component

Wij maken een React component in 2 stappen:

1. Definieer een component als een function of class
2. Display een component in de UI met een JSX tag

https://www.w3schools.com/react/react_components.asp

Components

Stap 1 Definieer een react component

In deze stap definieer je een React component als een functie.

```
function Header() {
  return (
    <header>
      <h1>Scoreboard</h1>
      <span className="stats">Player: 1</span>
    </header>
  )
}
```

Components

Stap 2 Renderen component

In deze stap definieer je een React component als een functie.

```
ReactDOM.render(
  <Header />,
 document.getElementById('root')
);
```

Components

React components als arrow functions

https://www.w3schools.com/js/js_arrow_function.asp

```
const Header = ()  => {
   return (
      <header>
         <h1>Scoreboard</h1>
         <span className="stats">Player: 1</span>
      </header>
   )
}
```

Components

Composing components

Het is natuurlijk mogelijk om veel functies en content te stoppen in 1 component, maar dit maakt alles al snel onoverzichtelijk.

Composing components is het **logisch inrichten en het opbreken van components in kleinere components**. In React richten we components in aan de hand van doelen; bijvoorbeeld het tonen en updaten van de spelers lijst. Het tonen van de score, het tonen van de header etc.

Components

Composing components

Deze opgeknipte components met hun eigen doel kunnen weer worden gebruikt in andere components:

```
const App = ()  => {
   return (
      <div className="scoreboard">
         <Header />


         {/*Player list*/}
         <Player />
     <div>
   )
}
```

React dev tools

Bij het werken met HTML CSS en JavaScript heb je al kennis gemaakt met chrome dev tools, waarbij je HTML elementen kan inspecteren, CSS bekijken en aanpassen, JavaScript logs kunt lezen en errors kunt bekijken.

Voor React is er een extra extensie die je kunt downloaden om React Components te bekijken. Hier kun je de props en state bekijken (gaan we nog leren).

De extensie werkt in chrome en firefox. Downloaden chrome:

> **https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en**

React dev tools

Gebruiken dev tools

Extensie pinnen: klik op extensions -> klik op pin achter devtools.

Zien of een website react gebruikt:
Ga naar reactjs.org -> de extensie licht op -> hover en zie dat het de production build gebruikt.

Ga naar liveserver van jouw React pagina -> de extensie licht op -> hover en zie dat wij de development build gebruiken.

Componenten bekijken:
Ga naar liveserver van jouw React pagina -> inspecteren -> tabblad componenten -> bekijk een component

Properties (props)

Het gebruiken van properties (props) zijn een belangrijk concept in React. Ze worden gebruikt om datat te versturen naar een component.

Components zijn hierdoor aanpasbaar en herbruikbaar. Denk aan onze player component die straks aanpasbaar en herbruikbaar is voor elke speler.

Gebruiken props

Wij gebruiken van props gebeurt in 2 stappen:

1. Definieer de properties in een component haar JSX tag
2. Gebruik de props in een component

https://www.w3schools.com/react/react_props.asp

Properties (props)

Stap 1 Definieer props

- props geef je mee in de component tag waar de component wordt aangeroepen.
- De namen van props mag je zelf verzinnen (vaak camelCase)
- Een string geef je mee in dubbele quotes. Andere waarden in curly braces
- De properties zijn nu zichtbaar in React dev tools

```
const App = ()  => {
    return (
        <div className="scoreboard">
            <Header title="Scoreboard" totalPlayers={1} />
        </div>
  )
}
```

Properties (props)

Stap 1 Definieer props

- In onze component function kunnen we nu props ontvangen. In de ronde haken van onze component typen we props
- via een console.log(props) kun je nu de properties zien.
- De values van deze props kun je gebruiken via een dot notation bv: { props.title }

```
const Header = (props)  => {
   return (
      <header>
         <h1>{props.title}/h1>
         <span className="stats">Player: {props.totalPlayers}</span>
      </header>
   )
}
```

Properties (props)

Props zijn read only (immutable)

- Belangrijk is om te onthouden dat props ‘read only’ zijn (immutable). Dit betekent dat een component aleen de properties kan lezen en niet veranderen.

```
const Header = (props)  => {
   return (
      <header>
         <h1>{props.title = "Fun Game"}/h1>
      </header>
   )
}
```

Dit kan dus niet!

Properties (props)

Reuseable components met props

Doordat de content van onze components niet meer vast staan, maar meegegeven worden via de properties kunnen wij onze components hergebruiken en elke keer andere waarden meegeven. Denk aan de player component. Elke speler heeft een andere naam en een andere score.

- Indien components in components zitten dan geef je in de meeste situaties de waarden van de properties mee aan de parent component.

```
<Player name="Klaas" score={12} />
<Player name="Piet" score={22} />
<Player name="Clara" score={55} />
```

Properties (props)

Reuseable components met props

```
const Player = (props)  => {
    return (
        <div className="player">
            <span className="player-name">{props.name}</span>
            <Counter score={props.score} />
        </div>
    )
}
```

```
const Counter = (props)  => {
    return (
        <div className="counter">
            <span className="counter-score">{props.score}</span>
        </div>
    )
}
```

Itereren en renderen met map()

Met map itereer (loop) je over een array en maakt een nieuwe array met een function call voor elk element in de array. Deze functie returnt een waarde die in de nieuwe array geplaatst wordt.

Voorbeeld array met getallen en je wil het kwadrant weten:

```
const numbers = [1,2,3,4,5];
const newArr = numbers.map(myFunction);

function myFunction(num) {
    return num * num;
}

console.log(newArr);
```

https://www.w3schools.com/jsref/jsref_map.asp

Itereren en renderen met map()

Toepassen map() op players

Stap 1 Array

```
const players = [
    {
        name: "maurice",
        score: 200
    },
    {
        name: "james",
        score: 100
    }
];
```

Stap 2 players array meegeven via props

```
ReactDOM.render(
  <App players={players}/>,
 document.getElementById('root')
);
```

Itereren en renderen met map()

Toepassen map() op players

Stap 3 Gebruik de props players in een map() in App

```
const App = (props)  => {
    return (
        <div className="scoreboard">
            {props.players.map( player =>
                <Player name={player.name} score={player.score}/>
            )}
        </div>
    )
}
```

Keys

Gebruik keys voor het bijhouden van elements

Een unieke identifier dat React een snelle en makkelijke manier geeft om een element uit een lijst te identificeren. In ons voorbeeld hebben we meerdere players, om deze makkelijk te identificeren kunnen we alle players een unieke key geven.

- Een key is een propertie
- Een key value behoort een string te zijn. Gebruik .toString()
- Een key value behoort uniek te zijn

https://www.w3schools.com/react/react_lists.asp

Keys

Toepassen keys op players

1 geef de array een nieuwe key en unieke value. Bijvoorbeeld id (mag ook een andere naam)
2 Geef het component een nieuwe prop genaamd key. geef de waarde de aangemaakte id als string

```
const players = [
    {
        name: "maurice",
        score: 200,
        id: 1
    },
    {
        name: "james",
        score: 100,
        id: 2
    }
];
```

```
{props.players.map( player =>
    <Player
        name={player.name}
        score={player.score}
        id={player.id.toString()}
    />
)}
```

Opdracht

Top 5 lijst

Gebruik React om een pagina te maken van een top 5 (bv: steden, voetbalspelers, games). Deze top 5 heeft iig een afbeelding, daarnaast iets als aantal inwoners, salaris etc. Maak componenten en gebruik map().