Quiz App - deel 1

In deze les gaan we ook nieuwe (REACT) techniek leren. We leren wat props zijn en hoe we daarmee kunnen werken.

Verder zullen we state variabelen gebruiken, we maken gebruik van een JSON-datastructuur.

We maken grote stappen en gaan een volledige Quiz app maken in React. De app komt er zo uit te zien:

image-1628456332603.png

Testen!

We gaan deze app stapje-voor-stapje maken en we proberen elke (kleine) stap te testen. Als je programmeert dan is het handig om elk klein stapje te testen, als het dan fout gaat dan weet je goed waar je naar moet kijken. Zeker met React werkt dit heel goed want op het moment dan je jouw aanpassing bewaard zie je gelijk of je ene fout hebt gemaakt.

Soms moet je om te kunnen testen even extra (tijdelijke) code maken. Bijvoorbeeld met `console.log()`. Dit lijkt soms veel werk, maar het is de *enige* manier om een goed werkende applicatie te krijgen. Bij test-driven development maak je zelfs vaak eerst de code om te testen, dan schrijf je de code en dan test je. We zullen dit in deze les bij sommige stappen ook toepassen.

image-1628549683281.png

Lees goed en ga pas door naar de volgende stap als je begrijpt wat je hebt gedaan.

In het kort

In deze les gaan we de volgende stappen doorlopen;

1. We beginnen met een 'lege' standaard app zoals we die in de eerste les hebben gemaakt.
2. We maken een paar state variabelen (uitleg zie vorige les).
3. We maken de vragen in JSON en zetten die in een constante (dat is een variabele die niet veranderd).
4. We laten de (eerste) vraag op het scherm zien en stijlen dit een beetje.
5. We laten antwoorden zien en zorgen dat door de vragen heen kunnen lopen.

Stap 1 tot en met 4 is allemaal set-up en het echte coderen begint eigenlijk pas echt bij stap 5.

Stap 1, We beginnen met een standaard App.js

```
import './App.css';

import React from 'react';

class App extends React.Component {

  constructor(props) {
    super(props);
    this.state = {

    }
    this.getNumbers();
  }



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

      </div>
    );
  }

}

export default App;
```

Stap 2, state variabelen

Tijdens het spelen van het spel moeten we bijhouden bij welke vraag we zijn en hoeveel goede antwoorden we hebben gegeven.

Maak hiervoor twee state variabelen aan; vraagNr om bij te houden bij welke vraag we zijn en score om de score bij te houden.

```
this.state = {
  vraagNr: 0,
  score: 0
}
```

Stap 3, we moeten vragen maken

We kunnen de vragen uit een database halen, maar om te beginnen zetten we de vragen gewoon even in de code als een constante. In de constante plaatsen we een JSON-data structuur met de vragen.

We gebruiken de JSON data structuur omdat we uiteindelijk de vragen uit een REST API gaan krijgen. Wat dat allemaam precies is dat wordt nog uitgelegd, maar een REST API geeft als resultaat ook een JSON-datastructuur terug. We beginnen nu dus alvast met JSON dan is de aanpassing later eenvoudiger.

Zet het volgende vlak boven de *class App* (regel 4).

```
const questions = [
    {
        questionText: 'Wat is de hoofdstad van Frankrijk?',
        answerOptions: [
            { answerText: 'New York', isCorrect: false },
            { answerText: 'London', isCorrect: false },
            { answerText: 'Paris', isCorrect: true },
            { answerText: 'Dublin', isCorrect: false },
        ],
    },
    {
        questionText: 'Wie is de oprichter van Tesla?',
        answerOptions: [
            { answerText: 'Jeff Bezos', isCorrect: false },
            { answerText: 'Elon Musk', isCorrect: true },
            { answerText: 'Bill Gates', isCorrect: false },
            { answerText: 'Tony Stark', isCorrect: false },
        ],
    },
    {
        questionText: 'Welk bedrij was eind jaren 90 bijna failliet?',
        answerOptions: [
```

```
            { answerText: 'Apple', isCorrect: true },
            { answerText: 'Intel', isCorrect: false },
            { answerText: 'Amazon', isCorrect: false },
            { answerText: 'Microsoft', isCorrect: false },
        ],
    },
    {
        questionText: 'Hoeveel Harry Potter boeken zijn er?',
        answerOptions: [
            { answerText: '1', isCorrect: false },
            { answerText: '4', isCorrect: false },
            { answerText: '6', isCorrect: false },
            { answerText: '7', isCorrect: true },
        ],
    },
    {
        questionText: 'Wat is GEEN Cryptomunt?',
        answerOptions: [
            { answerText: 'Bitcoin', isCorrect: false },
            { answerText: 'EOS', isCorrect: false },
            { answerText: 'Pancake Swap', isCorrect: false },
            { answerText: 'XOTA-XL', isCorrect: true },
        ],
    },
];
```

Stap 4a, Laat de vraag zien

De vragen staan in het JSON object en kunnen getoond worden met deze code:

```
<div>{questions[this.state.vraagNr].questionText}</div>
```

De code laat van het JSON array het eerste element (question[0]) zien. Van dat element wordt de questionText getoond.

Kijk maar eens wat er gebeurt als je deze code gebruikt:

```
<div>{questions[this.state.vraagNr].answerOptions[0].answerText}</div>
```

Kijk naar de code en naar het JSON-object en probeer te begrijpen hoe je gegevens uit het JSON-object kan laten zien.

Als je dat begrijpt, verander dan de code nu zodat je de vraag met daaronder de vier antwoorden krijgt.

image-1628538140614.png

Stap 4b, Tijd voor een beetje CSS!

Allereerst moet je weten dat de allereerste pagina die door REACT wordt geladen de file *public/index.html* is. Als je deze pagina opent dan zie je een normale HTML-layout en in de body wordt het root element aangeroepen. Deze staat in *src/index.js* en vanuit daar wordt de App.js aangeroepen.

Zoals je kunt zien in de public/index.html, wordt er een body tag gebruik. Laten we eerst de body stijlen. Voeg een App.css file toe en zorg ervoor dat de css file in de App.js file wordt geïmporteerd. In het voorbeeld van stap 1 gebeurde dat al op regel 1.

De body stijl (in App.css zetten).

```
body {
  background-color: #7cc6fe;
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  font-family: "Verdana", cursive, sans-serif;
  color: #ffffff;
}
```

Nu gaan we een mooie box om de vraag heen zetten. In de app stijl, dit is de eerste div in de *App.js* file.

Maak een tweede stijl voor de app en zet deze ook in de App.css.

```
.app {
  background-color: #252d4a;
  width: 450px;
  min-height: 200px;
  height: min-content;
  border-radius: 15px;
  padding: 20px;
  box-shadow: 10px 10px 42px 0px rgba(0, 0, 0, 0.75);
}
```

Dit zorgt voor een mooie box om onze vraag heen, onze app ziet er tot nu toe zo uit:

image-1628546817642.png

Stap 5, antwoord kunnen geven

Om een antwoord te kunnen geven, moeten je op de antwoorden kunnen klikken. We doen dat door van de antwoorden buttons te maken.

```
<button>{questions[this.state.vraagNr].answerOptions[0].answerText}</button>
<button>{questions[this.state.vraagNr].answerOptions[1].answerText}</button>
<button>{questions[this.state.vraagNr].answerOptions[2].answerText}</button>
<button>{questions[this.state.vraagNr].answerOptions[3].answerText}</button>
```

Dit gaan we iets vereenvoudigen. We maken vlak na de regel *render() {* een nieuwe variabele:

```
var antwoorden=questions[this.state.vraagNr].answerOptions;
```

Omdat de variabele antwoorden nu array met vier elementen is kunne we de buttons vereenvoudigen.

```
<button>{antwoorden[0].answerText}</button>
<button>{antwoorden[1].answerText}</button>
<button>{antwoorden[2].answerText}</button>
<button>{antwoorden[3].answerText}</button>
```

Als de button wordt ingedrukt moeten we een script uitvoeren. Dit script moet bepalen of het gegeven antwoord correct is.

Dat staat in `antwoorden[0].isCorrect`, `antwoorden[1].isCorrect`, etc.

Laten we dit even controleren/testen.
Verander hiervoor **tijdelijk** bij de eerste vraag *de false in* een 0 en *de true* een 1. Deze waarden kun je namelijk makkelijker afdrukken. We passen dus onze code tijdelijk aan om te kunnen testen!

Vervang dan de buttons door de volgende code

```
<button>{antwoorden[0].answerText} {antwoorden[0].isCorrect}</button>
<button>{antwoorden[1].answerText} {antwoorden[1].isCorrect}</button>
<button>{antwoorden[2].answerText} {antwoorden[2].isCorrect}</button>
<button>{antwoorden[3].answerText} {antwoorden[3].isCorrect}</button>
```

Je ziet nu het volgende:image-1628548398163.png

De 1 achter Parijs geeft dus aan dat dat het juiste antwoord is. Natuurlijk moeten we dit weer weg halen.

Vergeet ook niet de 1-en en 0-en in de eerste vraag weer te vervangen in `true` en `false`.

We roepen nu een script aan als je op de button drukt:

```
<button onClick={() =>
this.handleAnswerOptionClick(antwoorden[0].isCorrect)}>{antwoorden[0].answerText}</button>
```

Hierboven staat één button, maak zelf de andere drie.

Als de button wordt ingedrukt dan wordt de methode (functie) this.*handelAnswerOptionClick* uitgevoerd.

Het woord *this* geeft aan dat de methode in deze Class staat.

We schrijven nu een nieuwe methode om te testen of alles goed werkt.

```
handleAnswerOptionClick(isCorrect) {
  console.log("Button clicked");
  if (isCorrect) {
    console.log("Antwoord is goed!");
  }else{
    console.log("Antwoord is fout!");
  }
}
```

Kijk naar je console log (meeste browsers CTRL-Shift-i) en test je code. Druk op de verschillende knoppen en controleer of het juiste antwoord herkend wordt.

Als alles goed is hoef je alleen nog maar naar de volgende vraag te springen als je het antwoord hebt gegeven. De huidige vraag wordt bijgehouden in de state variabele vraagNr, dus die moeten we één ophogen nadat het antwoord is gegeven.

Laten we dit in regel 7.5 (dus tussen regel 7 en 8) van de methode *handleAnswerOptionClick* doen.In de vorige les is uitgelegd dat je een state variabele alleen mag veranderen door gebruik te maken van de functie *this.setstate()*

```
this.setState( {'vraagNr': this.state.vraagNr+1} );
```

Test je code!

Oops het gaat goed tot de laatste vraag. Dat lijkt logisch want als de state variabele wordt opgehoogd en die wordt hoger dan het aantal vragen dan wordt er een vraag aangeroepen die niet

bestaat.

Om te weten wanneer we niet meer naar de volgende vraag kunnen, moeten we weten hoeveel vragen er zijn. DIt is gelijk aan de lengte van de constante *questions.* De constante *questions* is een array met 5 elementen, elk element heeft weer twee elementen en deze elementen hebben ook weer elementen. Het gaat echter om het aantal vragen en dat is het aantal elementen in de constante *questions*, 5 dus.

Laten we het even testen, in de methode (functie) *handleAnswerOptionClick* zetten we:

```
console.log("Aantal vragen: "+questions.length);
```

Voer de test uit en kijk in je console log. Zoals je kunt zien in de test is het aantal vragen 5, de vragen lopen dus van 0,1,2,3 tot en met 4. Dus in dit geval mag 3 nog worden opgehoogd en bij 4 moeten we stoppen. We zetten het ophogen van de *vraagNr* dus in een *if*.

```
if ( this.state.vraagNr < questions.length - 1) {
  this.setState( {'vraagNr': this.state.vraagNr+1} );
}
```

Test alles goed en als alles goed werkt.

We zien dus alle vragen en in de console log zien we of we het antwoord goed of fout hebben beantwoord. Telkens als we een antwoord geven springen we naar de volgende vraag en bij de laatste vraag wordt er niet meer naar de volgende vraag gesprongen. Werk dat allemaal? Top! In het volgende deel gaan we iets met punten doen.

Opdracht

Zorg dat allen goed werkt en bedenk zelf één extra vraag en voeg deze vraag toe. Bijvoorbeeld:

image-1661593315876.png

Inleveren

Als alle klaar is dan maak je een screenshot van je *gehele* scherm waarin de browser is te zien met de laatste (zelfbedachte vraag) vraag van de quiz.

Deze les is een aangepaste versie van: https://www.freecodecamp.org/news/how-to-build-a-quiz-app-using-react/

Revision #19
Created 2021-08-08 20:58:15 UTC by Max
Updated 2022-08-27 11:35:23 UTC by Max