Quiz App - deel 2

We zijn bezig met het maken van een Quiz app te maken in REACT. Dit is deel twee waarin we onze App verder gaan afmaken. We leren hier ook een nieuwe techniek waarmee we in JSX door een array heen kunnen loopen.

Daarnaast gaan we met de Ternary oparatore een soort if-then-else maken in JSX.

Zorg ervoor dat je Quiz App - deel 1 goed hebt doorlopen en dat alles goed werkt.

Stap 1 - een loop in JSX met de map functie

Onze render methode in de class App ziet er tot nu toe zo uit:

```
render() {
  var antwoorden=questions[this.state.vraagNr].answerOptions;
  return (
    <div className="app">
        <div>{questions[this.state.vraagNr].questionText}</div>
        <button onClick={() =>
this.handleAnswerOptionClick(antwoorden[0].isCorrect)}>{antwoorden[0].answerText}</button>
        <button onClick={() =>
this.handleAnswerOptionClick(antwoorden[1].isCorrect)}>{antwoorden[1].answerText}
{antwoorden[1].isCorrect}</button>
        <button onClick={() =>
this.handleAnswerOptionClick(antwoorden[2].isCorrect)}>{antwoorden[2].answerText}
{antwoorden[2].isCorrect}</button>
        <button onClick={() =>
this.handleAnswerOptionClick(antwoorden[3].isCorrect)}>{antwoorden[3].answerText}
{antwoorden[3].isCorrect}</button>
    </div>
  );
}
```

Regel 6,7, 8 en 9 lijken erg veel op elkaar en kunnen in een loop worden geplaatst. De code is JSX (uitleg zie eerste les) en binnen JSX maken we een loop met de map functie. Lees hiervoor de uitleg op:

https://www.telerik.com/blogs/beginners-guide-loops-in-react-jsx

Lees de uitleg goed en vervang de vier regels dan door de volgende code:

```
{ antwoorden.map((antwoord) => (
  <button onClick={() =>
this.handleAnswerOptionClick(antwoord.isCorrect)}>{antwoord.answerText}
{antwoord.isCorrect}</button>
))}
```

Je loopt dus door het array antwoorden heen en het element komt telkens in antwoord. Dus de eerste keer in de loop geldt:

`antwoord = antwoorden[0]`

daarna wordt

`antwoord = atwoorden[1]`

`antwoord = antwoorden[2]`

etc.

Stap 2 - Voortgang (vraagnummer)

In de state variabele *vraagNr* houden we bij bij welke vraag we zijn. Deze telt vanaf 0. We zijn dus bij vraag `vraagNr+1` en we hebben in totaal `questions.length` vragen. Dat hadden we in onze vorige les al gebruikt om te bepalen of we al bij de laatste vraag waren.

We willen op de regels afdrukken "Vraag X van Y. X is dus de state variabale vraagNr en y is de lengte van het questionsArray. We maken een aparte div zodat we de tekst straks apart kunnen stijlen.

```
<div>
  Vraag {this.state.vraagNr + 1} van {questions.length}
</div>
```

Plaats deze JSX code zodat deze regel boven aan wordt geplaatst.

Stap 3 - CSS stijlen

We maken gebruik van flex boxen; we maken één rij met twee kolommen. De hele app is één rij dus de volgende code kan bij de CSS-stijl .app worden geplaatst.

```
display: flex;
flex-direction: row;
justify-content: space-evenly;
```

We maken de twee kolommen voeg de volgende class toe in de CSS file:

```
.column {
  width: 100%;
  display: flex;
  flex-direction: column;
  flex: 1;
}
```

We maken nu de volgende structuur.

```
<div className="app">

  <div className="column">
    <span>Vraag {this.state.vraagNr + 1} van {questions.length}</span>
    <span className="question">{questions[this.state.vraagNr].questionText}</span>
  </div>

  <div className="column">
    { antwoorden.map((antwoord) => (
      <button onClick={() =>
this.handleAnswerOptionClick(antwoord.isCorrect)}>{antwoord.answerText}
{antwoord.isCorrect}</button>
    ))}
  </div>

</div>
```

Als je je code test dan zie je dat de knoppen aan de rechterkant niet mooi zijn uitgelijnd. Maak een nieuw class.

```
.uitlijnen {
  justify-content: space-between;
}
```

En zet deze class bij de tweede kolom.

```
 <div className="column uitlijnen">
```

De buttons stijlen we als volgt (plaats de code in de app.css file.

```
button {
  width: 100%;
  font-size: 16px;
  color: #ffffff;
  background-color: #252d4a;
  border-radius: 15px;
  display: flex;
  padding: 5px;
  justify-content: flex-start;
  align-items: center;
  border: 5px solid #234668;
  cursor: pointer;
}

button:hover {
  background-color: #555e7d;
}

button:focus {
  outline: none;
}
```

De app is nu al bijna af!

De enige stap is nog dat we de score moeten bijhouden en aan het eind moeten stoppen en de score laten zien.

Stap 4, Score

We hadden al een score state variabele gemaakt. We hoeven alleen nog maar een punt bij de score op te tellen als de vraag goed is beantwoord. We weten ook al wanneer een vraag goed is beantwoord want daar hebben we nog een console.log staan.

Achter de console.log waar we afdrukken dat het antwoord goed is plaatsen we dus de volgende code.

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

Stap 5, naar het einde toe

In de method *handleAnswerOptionClick* testen we of de vraagnummer kunnen ophogen.

`if ( this.state.vraagNr < questions.length-1) {`

Als dit niet mag dan zijn bij de laatste vraag en dus bij het einde van de quiz. Als we dus een *else* maken dan komen we alleen in deze else als we de laatste vraag hebben beantwoord.

We maken een else en zetten een (nieuwe) state variabele stop op true.

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

Maak zelf de state variabele aan in de *construtor* en zet de waarde op `false`.

Als de state variabele stop true is dan moeten we stoppen met de quiz en moeten we geen vraag maar het eindresultaat laten zien.

We gaan de Ternary operator gebruiken in onze JSX code.

De opzet wordt als volgt:

(einde ? laat_einde_zien : laat_de_vraag_zien)

In code ziet dat er als volgt uit

```
render() {
  var antwoorden=questions[this.state.vraagNr].answerOptions;
  console.log( this.state.vraagNr, questions.length);
  return (
    <div className="app">
      { this.state.stop  ?  // hier testen we of we moeten stoppen, als we moeten stoppen laat
dan het einde zien.
        (
          <div>einde!</div>
        )
        : // dit is de else, we stoppen dus niet en laten de volgende vraag zien
        (
          <>
```

```
            <div className="column">
              <span>Vraag {this.state.vraagNr + 1} van {questions.length}</span>
              <span className="question">{questions[this.state.vraagNr].questionText}</span>
            </div>

            <div className="column uitlijnen">
              { antwoorden.map((antwoord) => (
                <button onClick={() =>
this.handleAnswerOptionClick(antwoord.isCorrect)}>{antwoord.answerText}
{antwoord.isCorrect}</button>
              ))}
            </div>
          </>
        )
      }
    </div>
  );
}
```

Kijk goed of je alles begrijpt, herken je de if - then -else? Test de code uit.

<> en </>. wtf?

Je ziet op regel 12 en 23 de tags <> en </>, dat zijn speciale JSX tags. een code blok moet namelijk altijd met één tag openen en sluiten. Als je de <> en </> weglaat dan heb je een op regel 13 en die sluit op regel 16 regel 16 moet dan het einde van het blok zijn. Je mag de <> en </> ook veranderen in een en . Het gaat erom dat de twee buitenste tags in een blok bij elkaar horen.

Stap 6, het eindresultaat

Nu laten we het einde zien. We willen natuurlijk onze score bekijken.

```
<div className='score'>
  Jouw score is {this.state.score} van de {questions.length}
</div>
```

De stijl die je kunt toepassen:

```
.score {
  display: flex;
```

```
  font-size: 24px;
  align-items: center;
}
```

Het eindscherm ziet er dan zo uit, maar je kunt natuurlijk zelf een andere stijl maken.

image-1661594536349.png

Stap 7, opruimen

Vergeet niet om de console.log statements uit jouw code te halen. Als alles klaar zijn zijn die niet meer nodig. Vanuit security is het aan te bevelen om alles wat niet nodig is weg te helen, zeker als het code is die inzicht geeft over hoe je iets hebt gecodeerd.

Inleveren

1. Lever een screendump in van jouw gehele laptop scherm met de browser waarin de uitslag van je quiz te zien is.
2. Lever de file App.js in.

--

Revision #18
Created 2021-08-10 08:43:23 UTC by Max
Updated 2022-08-27 11:35:23 UTC by Max