State variabelen gebruiken

We gaan de kleur van de letters aanpassen. De kleur moeten we onthouden en dat doen we in een state variabele

Dynamische CSS in JSX

We gaan nu een knop maken waarmee we de kleur van de tekst (myString) kunnen aanpassen.

We volgen de volgende stappen:

1. Knop maken die functie aanroept.
2. state variabele maken waarin de kleur wordt vastgelegd.
3. In de functie de state variabele aanpassen.
4. We maken de stijlen in de css file.
5. in de render method de style/class aanpassen; is de state variabele red dan gebruiken we een stijl waarin de tekst in rood wordt afgedrukt en is de state variabele green dan gebruiken we een stijl die de tekst groen maakt.

Stap 1, Knop maken

Allereerst maken we een button. Bij de vorige opdracht heb je jouw naam ergens laten afdrukken. Dat is de plaats waar nu de button moet komen.

```
<br />
<button onClick={() => this.changeColor() } >Change Color</button>
```

image-1661525117373.png

Als je de code nu runt en op de knop drukt, dan gebeurt er nog weinig. Dat komt omdat de method changeColor (nog) niet bestaat.

We moeten nu eerst een state variabele waarin we de kleur gaan bewaren aanmaken.

Stap 2, State variabele maken

voeg myColor toe als string aan de state variabele, let op dat je wel een komma gebruikt tussen de state variabelen.

```
this.state = {
   myString: '',
   myColor: 'red'
}
```

Nu gaan we de functie maken die vanuit de knop wordt aangeroepen maken.

Stap 3, State variabele aanpassen in functie

Een state variabele mag en kan alleen worden aangepast met een (ingebouwde functie *setState()*. Dat werkt als volgt:

```
changeColor() {
  if ( this.state.myColor === "red") {
    this.setState({ 'myColor': 'green'} );
  } else {
    this.setState({ 'myColor': 'red'} );
  }
}
```

De setState() method is ingebouwd in React en bedoeld om de waarde van state variabelen te wijzigen. De state variabelen staan in JSON formaat.

Wat deze functie doet is de variabele myColor op green zetten als die rood is, en op rood zetten als die groen is.

En werkt de knop al? Nee? Waarom niet?

Juist we moeten nog zorgen dat de kleur die wordt bewaard ook werkelijk wordt toegepast.

Eerst de CSS maken.

Stap 4, css aanpassen

In de css maken we twee extra classes.

```
.red {
  color: red;
}
.green {
  color: green;
}
```

En nu de juisten class (*red* of *green*) aan de className van de Div toevoegen.

Stap 5, class dynamisch maken

In de app.js file passen we nu de stijl aan.

De class wordt dus de waarde van de state variabele *myColor*. Deze is *green* of *red*. Deze stijlen heb je in de CSS aangemaakt in stap 4.

De class App is nu weg, laten we die ook nog even toevoegen.

```
<div className={"App "+this.state.myColor}>
```

Stel *myColor* is "red"dan staat er dus className="App red". Dit betekent dat CSS classes App en red worden toegepast.

Je kunt in de CSS file bij de classes red en green natuurlijk veel meer aanpassen. Probeer zelf maar!

Opdracht

Verander de method getNumbers().

```
getNumbers() {
  for (var i=0; i < 10; i++) {
    this.state.myString = this.state.myString + "*";
  }
}
```

Nu zie je 10 sterretjes in plaats van de nummers.

Maak nu een tweede button. Telkens als je op de button drukt laat je een extra sterretje zien. De string *myString* wordt dus telkens iets langer.

Tip: Omdat je moet onthouden hoeveel sterretjes er al staan, moet je dus een state variabele aanmaken. In de state variabele kun je dan telkens als je op de knop drukt een sterretje toevoegen.

Inleveren

De file App.js.

--

Revision #4
Created 2022-08-26 13:07:54 UTC by Max
Updated 2022-08-27 11:35:23 UTC by Max