React State

Lernziele

- [ ] Wissen, wie man Events in React anfügt
- [ ] Das Konzept von "state" verstehen
- [ ] `useState()` verwenden, um state in React zu verwalten
- [ ] Den React Lifecycle verstehen

Was ist State?

State ist Daten, die sich im Laufe der Zeit ändern. Denk an die Lampe auf deinem Schreibtisch. Sie kann ein- oder ausgeschaltet sein. Die Lampe befindet sich zu einem bestimmten Zeitpunkt in einem bestimmten Zustand, und dieser Zustand kann sich im Laufe der Zeit ändern.

Ein weiteres Beispiel könnte die Menge an Geld in deinem Portemonnaie sein. Zu jedem Zeitpunkt hast du einen bestimmten Betrag an Geld in deinem Portemonnaie, aber die Menge kann sich ändern. Der Zustand deines Portemonnaies kann sich ändern. Wenn du zum Supermarkt gehst, verringert sich die Geldmenge, während sie sich erhöht, wenn du zum Geldautomaten gehst.

Dieses Konzept gilt auch für Software. Deine App kann Daten haben, die sich im Laufe der Zeit ändern.

Denk an einen Beitrag in einer Social-Media-App. Du könntest einen bestimmten Beitrag geliked haben oder nicht. Der "liked"-Zustand eines Beitrags kann ein- oder ausgeschaltet sein, wie die Lampe auf deinem Schreibtisch.

Die Website deiner Bank bezieht sich auf dein Portemonnaie in der analogen Welt. Zu jedem Zeitpunkt zeigt die Banking-Software den aktuellen Kontostand an, den aktuellen Zustand. Du kannst die Banking-Software verwenden, um diesen Zustand zu ändern. Zum Beispiel könntest du Geld auf ein anderes Konto überweisen, um den Betrag im "balance"-State zu verringern.

Oft ändern sich solche statusbehafteten Daten nach einer Benutzerinteraktion, wie einem Klick auf einen Button.

State in React

In React arbeiten wir mit state, indem wir den `useState` Hook verwenden.

Wir rufen die `useState` Funktion auf und übergeben den **Anfangszustand** als Argument. Dies ist der Wert, der in unserer App verwendet wird, bis sich etwas ändert.

Das Aufrufen der `useState` Funktion gibt uns zwei Dinge zurück:

- eine Variable mit dem **aktuellen Zustand** als Wert
- die `set` Funktion, um einen **neuen Zustand** festzulegen

```jsx
import { useState } from "react";

function SocialMediaPost() {
  const [liked, setLiked] = useState(false);

  function toggleLiked() {
    setLiked(!liked);
  }

  return (
    <article>
      <p>Liked: {liked ? "Yes" : "No"}</p>
      <button type="button" onClick={toggleLiked}>
        {liked ? "Remove like" : "Add like"}
      </button>
    </article>
  );
}
```

> 💡 Es gibt eine Namenskonvention für React-Apps, bei der die State-Variable und die Funktion immer dem Muster x und setX folgen.

> 📙 Lies mehr über das **Konzept von State** in den React Docs.

In React wird state pro Instanz einer Komponente gekapselt. Denk an einen Feed in einer Social-Media-App. Der Feed ist eine Liste von Beiträgen. Jeder Beitrag ist eine individuelle Instanz der `SocialMediaPost` Komponente, jeweils mit einem eigenen Zustand. Wenn du den "liked"-Zustand eines bestimmten Beitrags änderst, bleiben alle anderen Beiträge unverändert.

Eine React-Komponente kann mehrere Zustände haben. Du kannst die `useState` Funktion so oft verwenden, wie du sie benötigst.

Du kannst alle Arten von Daten im state speichern (wie Booleans, Zahlen, Strings, Objekte oder Arrays).

```´jsx import { useState } from "react";

function SocialMediaPost() { const [liked, setLiked] = useState(false); const [comments, setComments] = useState([]); const [views, setViews] = useState(0);

/_ … _/

return

{/_ … _/}
; }
```

Was passiert, wenn sich der State ändert?

```
Um state in React zu verwalten, können wir nicht einfach eine "normale" Variable verwenden und einen neuen Wert zuweisen. React muss informiert werden, dass sich die Daten geändert haben.

Dies steht im Zusammenhang mit dem Render-Zyklus von React-Komponenten.

Wenn React eine Komponente rendert, führt es die Komponentenfunktion aus, die JSX zurückgibt. Wenn das JSX eine State-Variable enthält, wird der Wert dieser Variable zu diesem Zeitpunkt in das JSX eingefügt. Das Aufrufen der `set`-Funktion mit einem neuen Wert informiert React, dass sich der Zustand geändert hat.

> 💡 Eine Änderung des States löst ein erneutes Rendern der Komponente aus.

Beim erneuten Rendern der Komponente führt React die Komponentenfunktion erneut von oben nach unten aus, die wiederum JSX zurückgibt. Diesmal hat die Variable jedoch einen neuen Wert - den Wert, der beim Aufruf der `set`-Funktion übergeben wurde. Das bedeutet, dass das zurückgegebene JSX den neuen Wert enthält.

> 📙 Lies mehr über [**State-Updates und erneutes Rendern** in den React Docs](https://react.dev/learn/render-and-commit).

---

## Resources

- [React Docs: Adding Interactivity](https://react.dev/learn/adding-interactivity)
- [React Docs: Responding to Events](https://react.dev/learn/responding-to-events)
- [React Docs: A simple variable is not enough](https://react.dev/learn/state-a-components-memory#when-a-regular-variable-isnt-enough)
- [React Docs: Render and commit](https://react.dev/learn/render-and-commit)
- [MDN: react events and state](https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_interactivity_events_state)
```