React Props

Lernziele

- [ ] Verstehen, was Props sind
- [ ] Verstehen, wie man Props an eine Komponente übergibt
- [ ] Verstehen, wie man Props in einer Komponente verwendet
- [ ] Verstehen, wie man bedingt rendert

Verwendung von Props

Props ist die Abkürzung für properties (Eigenschaften). Sie sind ein Weg, um Daten an eine Kindkomponente zu übergeben. Eine Komponente erhält ein Props-Objekt als ersten Funktionsparameter.

Props werden als Attribute an eine Komponente übergeben.

```
function UserCard(props) {
  return <div>{props.name}</div>;
}
```

Der Einfachheit halber wird das Props-Objekt oft im Funktionsparameter destrukturiert.

```
function UserCard({ name }) {
  return <div>{name}</div>;
}
```

Du kannst beliebige Namen für deine Props wählen.

> 💡 Es gibt jedoch einige Namenskonventionen. Boolean-Props werden oft mit `is`, `has` oder `should` präfixiert. Zum Beispiel `isDisabled`, `hasError` oder `shouldShow`. Props, die Funktionen enthalten, werden oft mit `on` präfixiert. Zum Beispiel `onClick`, `onSubmit` oder `onHover`. Das Befolgen dieser Konventionen macht es einfacher, den Zweck des Props zu verstehen.

Props können jeden Datentyp haben (string, number, array, object, function, ...).

Du solltest das Props-Objekt als unveränderlich und schreibgeschützt behandeln.

Übergabe von Props an eine Komponente

Props werden als Attribute an eine Komponente übergeben.

```
<UserCard name="Alex" />
```

Du kannst Daten jeglichen Typs als Prop übergeben.

```
<UserCard
  name="Alex"
  age={25}
  onContact={() => console.log("let's chat!")}
  isFavorite={true}
  favoriteFoods={["Pasta", "Salad"]}
  contactDetails={{ email: "alex@spiced.com", phone: "123456789" }}
/>
```

String-Props können mit Anführungszeichen übergeben werden. Alle anderen Props müssen mit geschweiften Klammern übergeben werden.

> 💡 Beachte die doppelten geschweiften Klammern für das Objekt. Das liegt daran, dass die äußeren geschweiften Klammern verwendet werden, um einen JavaScript-Ausdruck zu kennzeichnen. Die inneren geschweiften Klammern werden verwendet, um ein Objekt zu definieren.

> 💡 Es gibt eine Kurzschreibweise für Boolean-Props. Wenn der Wert `true` sein soll, kannst du den Wert weglassen.

```
<UserCard isFavorite />
```

Das Weglassen eines Attributs führt dazu, dass der Wert für dieses Prop `undefined` ist.

> 📙 Lies mehr über **Passing props to a Component** in den React Docs.

Bedingtes Rendern

Du kannst Props verwenden, um Teile einer Komponente bedingt zu rendern.

```
function UserCard({ name, isFavorite }) {
  return (
    <div>
      {name}
      {isFavorite ? <span>🌟 </span> : null}
    </div>
  );
}
```

> 💡 In JSX ist `null` eine Möglichkeit, nichts zu rendern.

Du kannst keine `if`-Anweisung innerhalb von JSX verwenden, da nur Ausdrücke erlaubt sind. Du kannst jedoch eine `if`-Anweisung außerhalb des JSX verwenden.

```

```

```
function UserCard({ name, isFavorite }) {
  let favoriteStar = null;
  if (isFavorite) {
    favoriteStar = <span>🌟 </span>;
  }

  return (
    <div>
      {name}
      {favoriteStar}
    </div>
  );
}
```

> 📙 Lies mehr über **Conditional Rendering** in den React Docs.

Resources

- Passing props to a Component in the React Docs
- Conditional Rendering in the React Docs