ToiletPaper #101

React Styled-Components

Autor: C. D.

✕ Problem

"Zwei CSS Properties gehen in eine Bar. Ein Barstuhl in einer komplett anderen Bar fällt um." – Jeder Entwickler, der schon mit **Cascading Style Sheets (CSS)** zu tun hatte, kennt die Thematik dieses Witzes: CSS haben globale Reichweite. Dies verursacht oft ungewollte Nebeneffekte und Bugs - kurzum, unnötigen Zusatzaufwand bei der Entwicklung. Insbesondere mit folgenden Problemen war ich im aktuellen Projekt schon konfrontiert:

- Kollisionen von Klassennamen
- Konfligierende CSS Regeln
- Schwierigkeiten bei der Eliminierung von „Dead Code“

✓ Lösung

Für React-Applikationen bieten **Styled-Components** einen eleganten Lösungsansatz für die oben genannten Probleme. Styled-Components definieren CSS-basiertes Styling auf modulare Weise für andere React-Komponenten. Sie werden **direkt im JavaScript-Code deklariert** und agieren als Wrapper für andere Komponenten im virtuellen DOM.

Die Zuordnung von Komponenten und Styles erfolgt nicht mehr manuell während der Entwicklung (z.B. via Klassennamen), sondern wird mithilfe generierter Klassennamen implizit durch die Bibliothek geregelt. Auf diese Weise kann der Kern der obigen Probleme angepackt werden – nämlich die globale Reichweite von CSS.

Indem die in einer Styled-Component spezifizierten CSS-Regeln nur den jeweiligen Kind-Komponenten zugewiesen werden, **beschränkt sich der Wirkungsbereich** auf die vom Programmierer intendierten Komponenten. Gleichzeitig bleibt das definierte Styling **wiederverwendbar,** ist klar von funktionalen und state-behafteten Komponenten getrennt und fügt sich intuitiv in die Codestruktur einer React-Applikation ein.

➔ Beispiel

Styled-Components Definitionen

```
const ContainerDiv = styled.div`
 width: 100%;
`;

const StyledButton = styled.button`
 color: red;
`;
```

Nutzung in React Komponente

```
return (
 <ContainerDiv>
    <StyledButton onclick={props.onclick}>
      Click me!
    </StyledButton>
  </ContainerDiv>
);
```

Resultierendes HTML und CSS

```
<div className="khQJAM">
 <button className="dEYsl">
    Click me!
  </button>
</div>
# Generierte CSS Klassen
.khQJAM { width: 100%; }
.dEYsl { color: red; }
```

✚ Weiterführende Aspekte

- Dokumentation von Styled-Components: https://www.styled-components.com/docs
- Blogpost zur Evolution von CSS: https://medium.com/@perezpriego7/css-evolution-from-css-sass-bem-css-modules-to-styled-components-d4c1da3a659b
- Beispiele für direkte Alternativen zu Styled-Components: Emotion (https://emotion.sh/) oder Glamorous (https://glamorous.rocks/)
- Andere Lösungsansätze für das Scoping-Problem: React Inline Styles (https://reactjs.org/docs/dom-elements.html#style) oder CSS Modules (https://github.com/css-modules/)