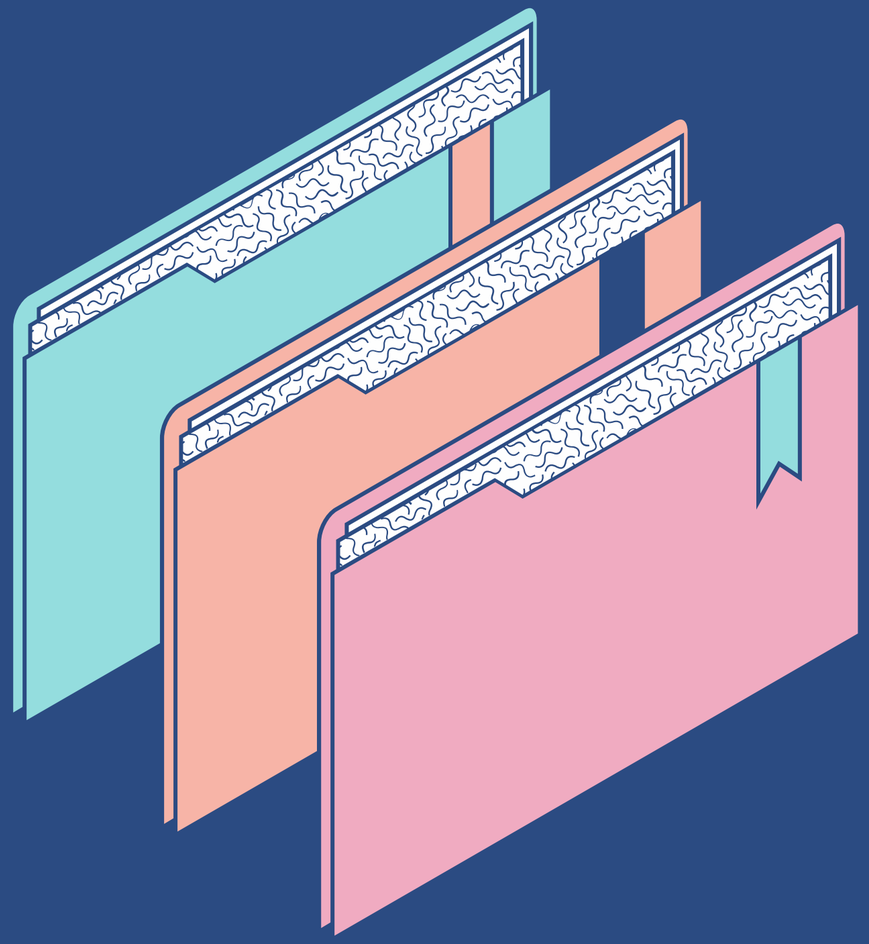
TEMAT 11

Ikony w Reacie

Alicja Dąbrowska
Jakub Bartosiewicz



Biblioteka React Icons

POZWALA DOŁĄCZAĆ DO PROJEKTU IKONY JAKO KOMPONENTY SVG

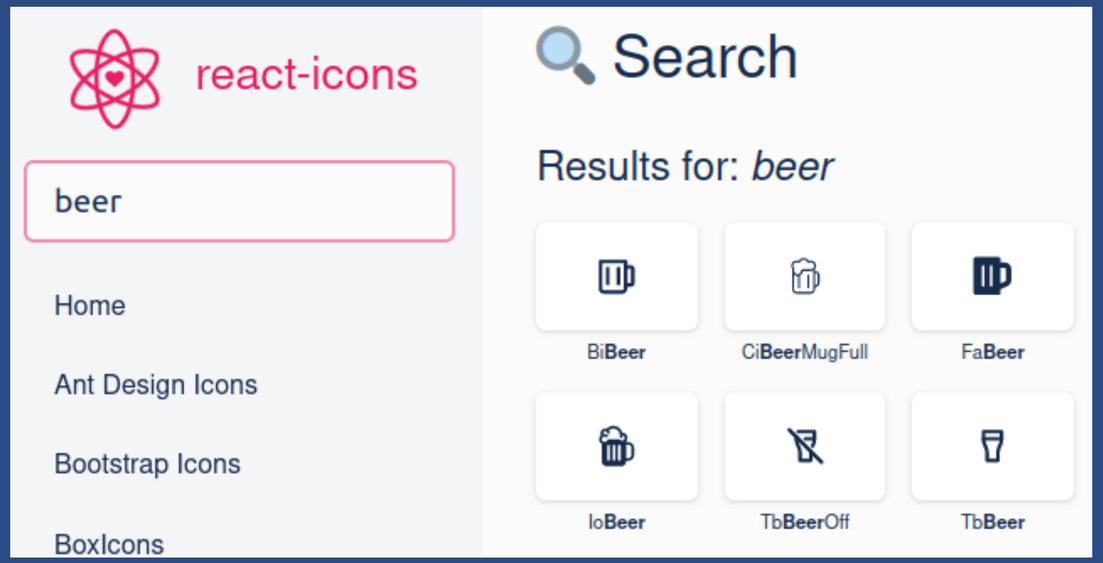
Obejmuje najczęśniej stosowane biblioteki ikon takie jak:

- Font Awesome,
- Ionicons,
- Bootstrap icons,
- Feather

Instalacja

```
npm install react-icons
```

Wybór ikony



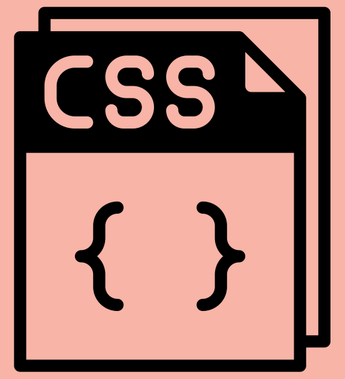
Prefiks w nazwie decyduje z jakiego setu pochodzi ikona, czyli skąd należy ją zaimportować

Implementacja

```
import { FaBeer } from 'react-icons/fa';

class Question extends React.Component {
  render() {
    return <h3> Lets go for a <FaBeer />? </h3>
  }
}
```

Stylowanie ikon Reacta



1. W pliku CSS - definiując rodzica

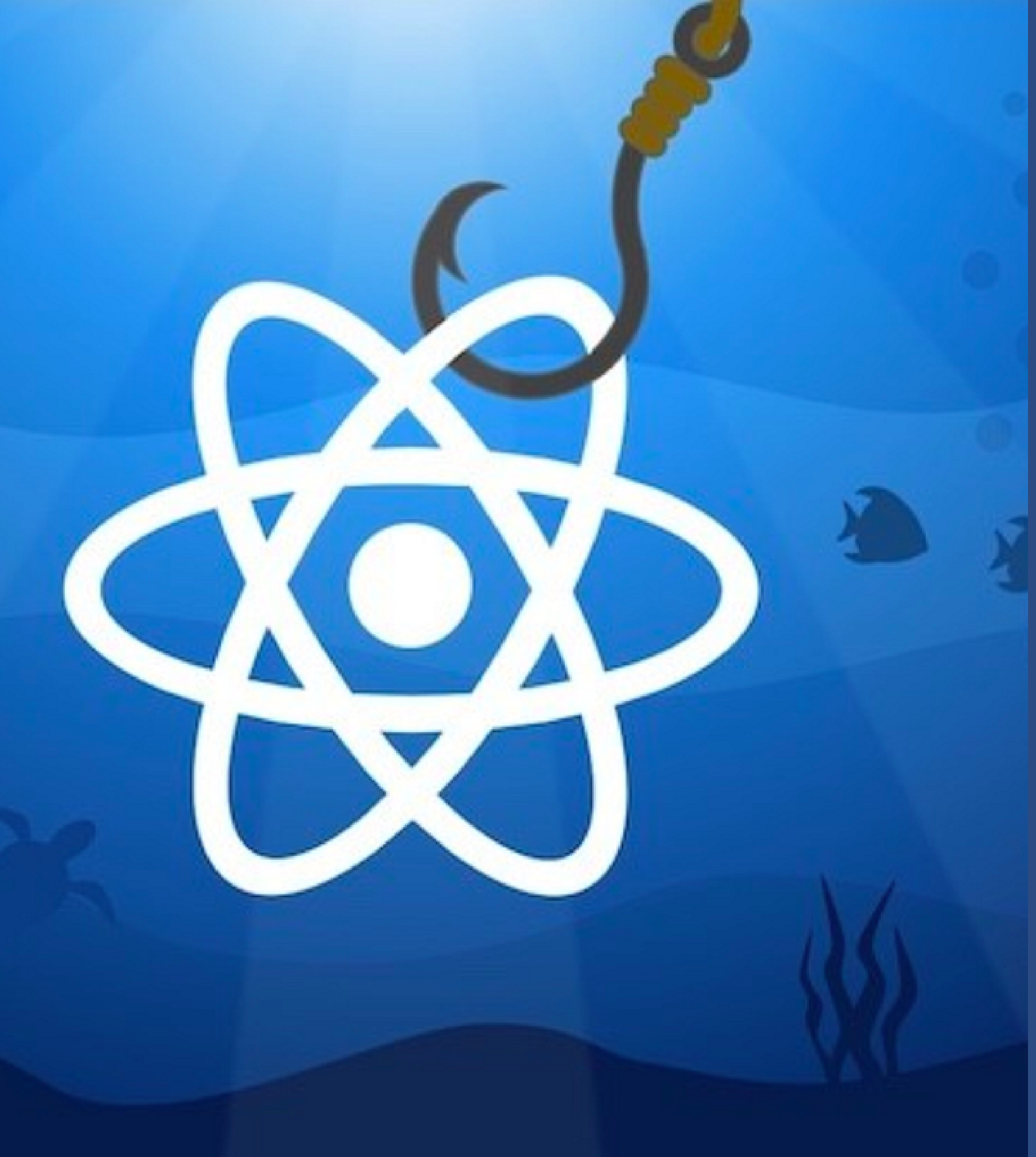
```css
.input-submit svg {
  color: #5e5e5e;
  font-size: 20px;
  margin-top: 2px;
}
```

2. Dodając atrybuty (np. color, size) do komponentu danej ikony.

```jsx
<button className="input-submit">
  <FaPlusCircle
    color="#5e5e5e"
    size="20px"
    className="submit-icon"
  />
</button>
```

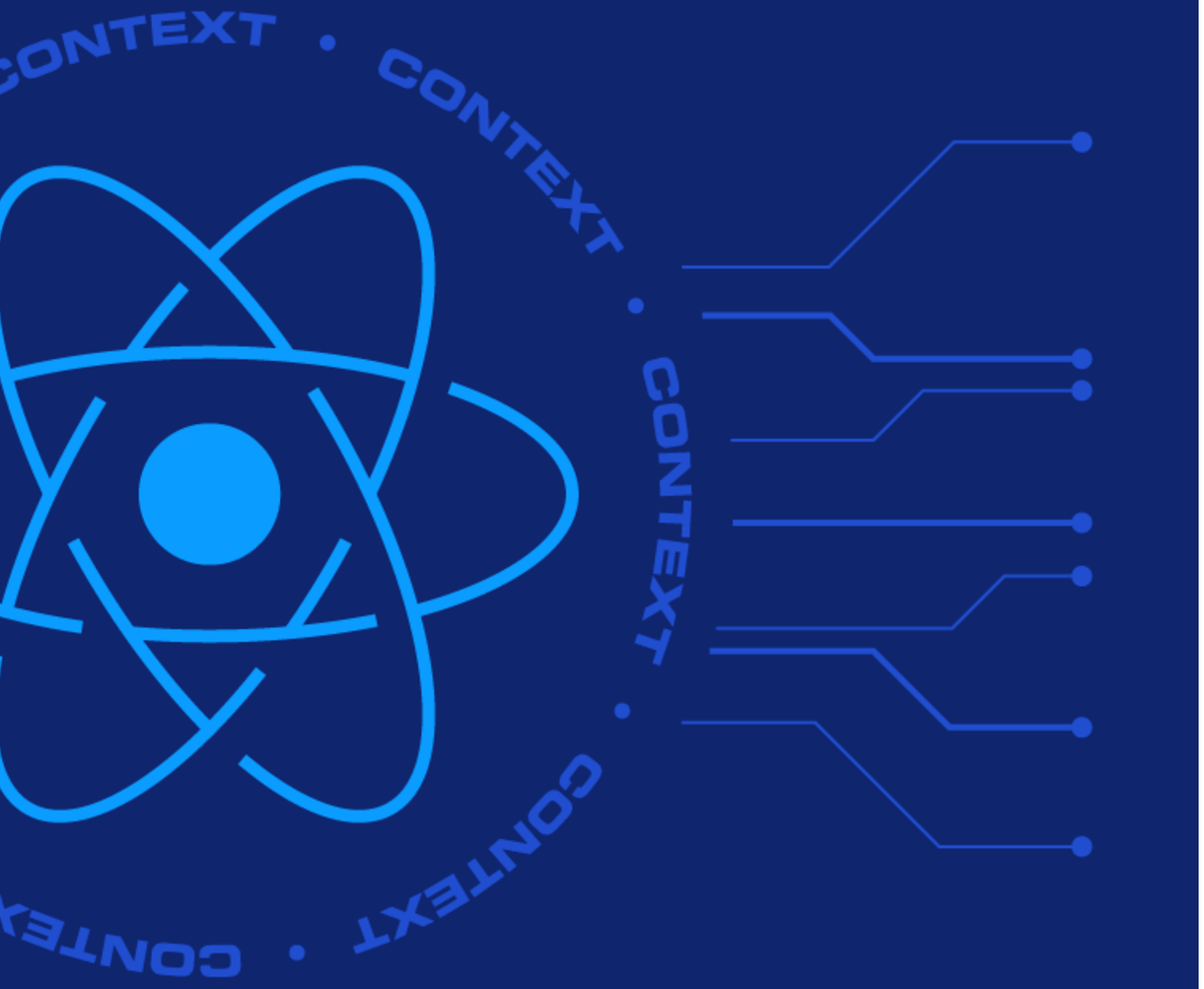
3. Za pomocą atrybutu style

```jsx
<button className="input-submit">
  <FaPlusCircle
    style={{
      color: '#5e5e5e',
      fontSize: '20px',
      marginTop: '2px',
    }}
  />
</button>
```



TEMAT 12

React Context API



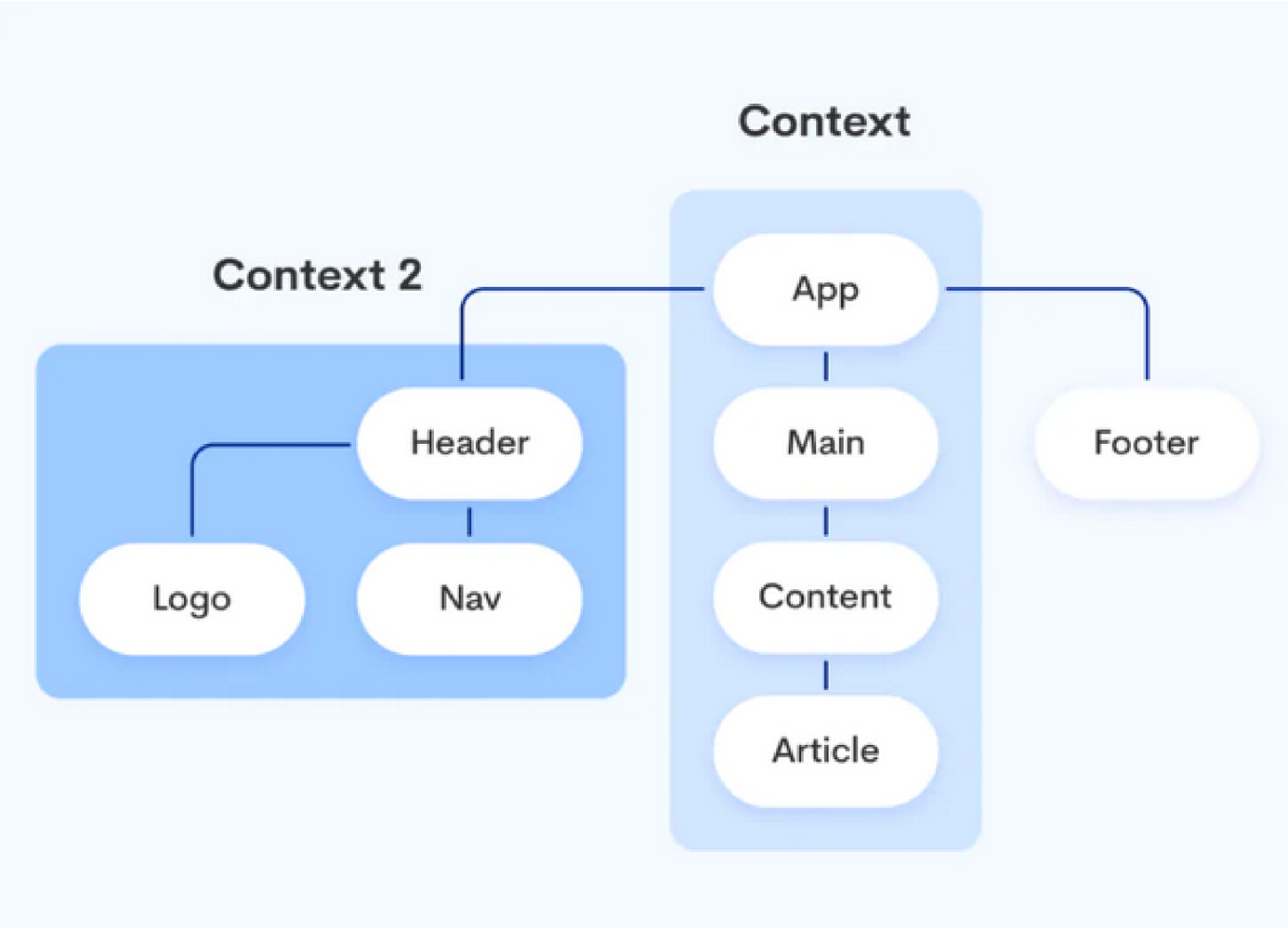
Czym jest Context API?

Context API to wbudowane narzędzie React, zaprojektowane do **dzielenia się stanem aplikacji**, czyli jest to coś, co pozwala nam zarządzać stanem naszej aplikacji nie tylko przez przekazywanie propsów w dół, ale przez **tworzenie pewnych kontenerów, które trzymają nasz stan i umożliwiają przekazywanie stanu w dół w drzewie komponentów do dowolnego miejsca w aplikacji.**

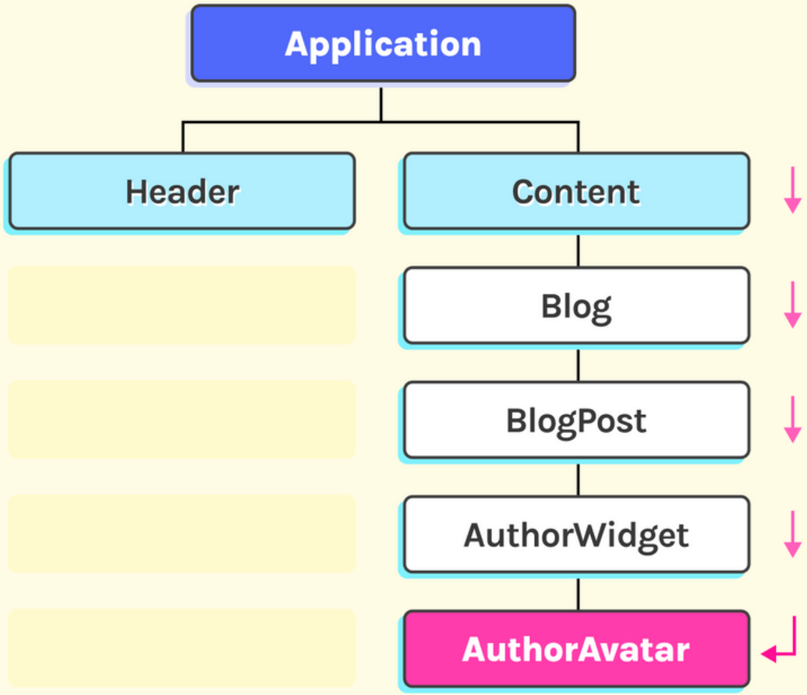
Na czym polega Context API?

- Tworzymy kontekst, czyli **miejsce do przechowywania stanu** i przekaźnik stanu do innych komponentów w postaci **„providerów”.**

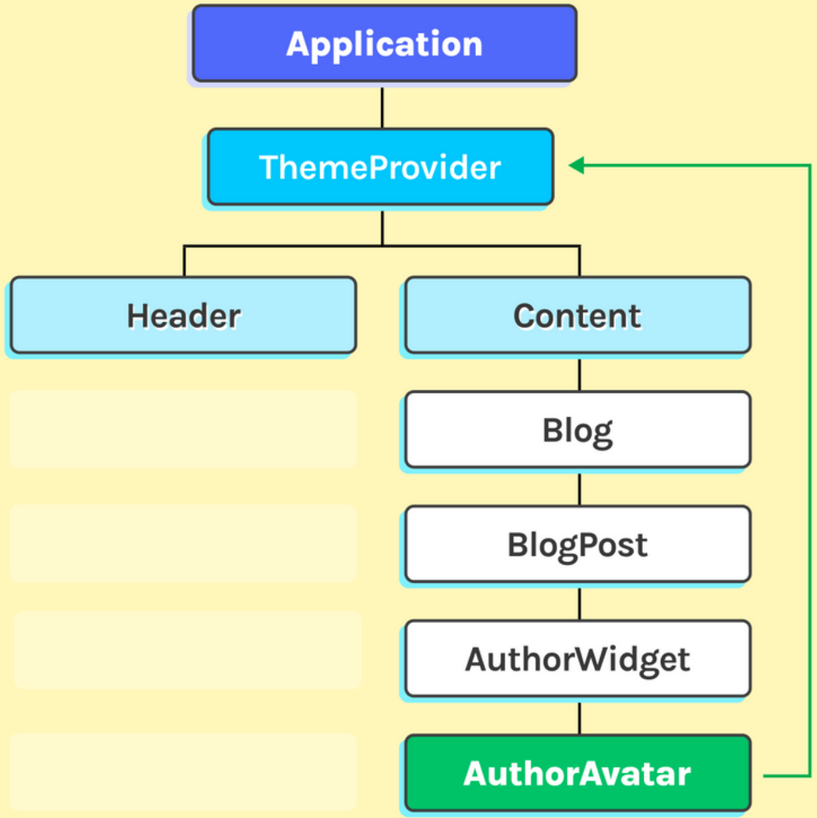
- Kontekst pozwala nam niejako „opleść” komponenty, które chcemy, by miały dostęp do danej rzeczy i w bardzo łatwy sposób **przekazywać dane wartości/stan w drzewie komponentów.**



Prop Drilling



React Context API



scrimba

1 createContext	2 Context.Provider	3 useContext
umożliwia tworzenie kontekstu, który komponenty mogą dostarczać lub odczytywać	umożliwia dostarczanie kontekstu komponentom	React Hook, który pozwala czytać i używać dany kontekst

1

```
import { createContext } from 'react';

const ThemeContext = createContext('light');
```

2

```
function App() {
  const [theme, setTheme] = useState('light');
  // ...
  return (
    <ThemeContext.Provider value={theme}>
      <Page />
    </ThemeContext.Provider>
  );
}
```

3

```
function Button() {
  // ✅ Recommended way
  const theme = useContext(ThemeContext);
  return <button className={theme} />;
}
```

Kiedy stosować Context API?

Jest idealny do zarządzania stanem, który **rzadko się zmienia**, a przekazywanie propsów do wielu głęboko zagnieżdżonych komponentów staje się problematyczne.

- **uwierzytelnianie użytkownika/ dane logowania**
- **motyw interfejsu,**
- **preferowany język,**
- **preferencje dotyczące ustawień regionalnych**

1. Unikaj umieszczania wszystkich danych stanu w kontekście globalnym. **Jeśli stan może pozostać lokalny w komponencie, zachowaj go lokalnie.**

2. Umieszczaj tylko **prawdziwie globalny stan** w Context Store.

3. Miej na uwadze, że każdy komponent korzystający z danych kontekstowych zostanie zaktualizowany i ponownie wyrenderowany!

Źródła:

https://ibaslogic.com/react-tutorial-for-beginners/
https://react-icons.github.io/react-icons/
https://it.legacy.reactjs.org/docs/context.html
https://react.dev/reference/react/useContext