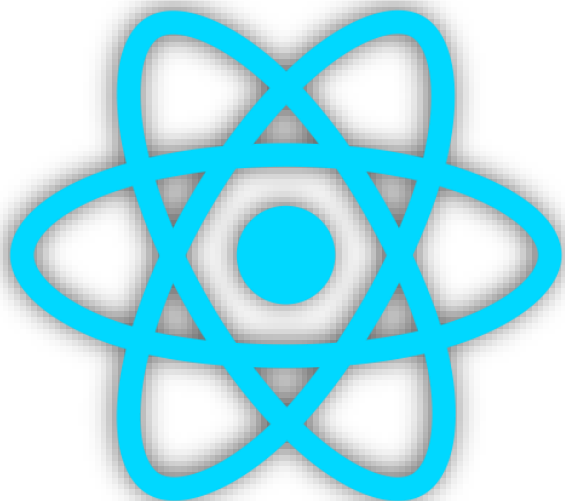


FRONT-END *Academy*

by Present Connection



Conditional rendering

- ◆ Komponentas arba jo dalis yra atvaizduojamas pagal sąlygą
- ◆ Naudojama if sąlyga arba jos trumpiniai (&&, ? :)
- ◆ Jeigu nenorit, kad neįvyktų komponento atvaizdavimas tiesiog grąžinam null



Conditional rendering

```
export default function Greeting(props) {  
  const { isLoggedIn } = props;  
  if (isLoggedIn) {  
    return <h1>Welcome back!</h1>;  
  }  
  return <h1>Please Sign Up</h1>;  
}
```

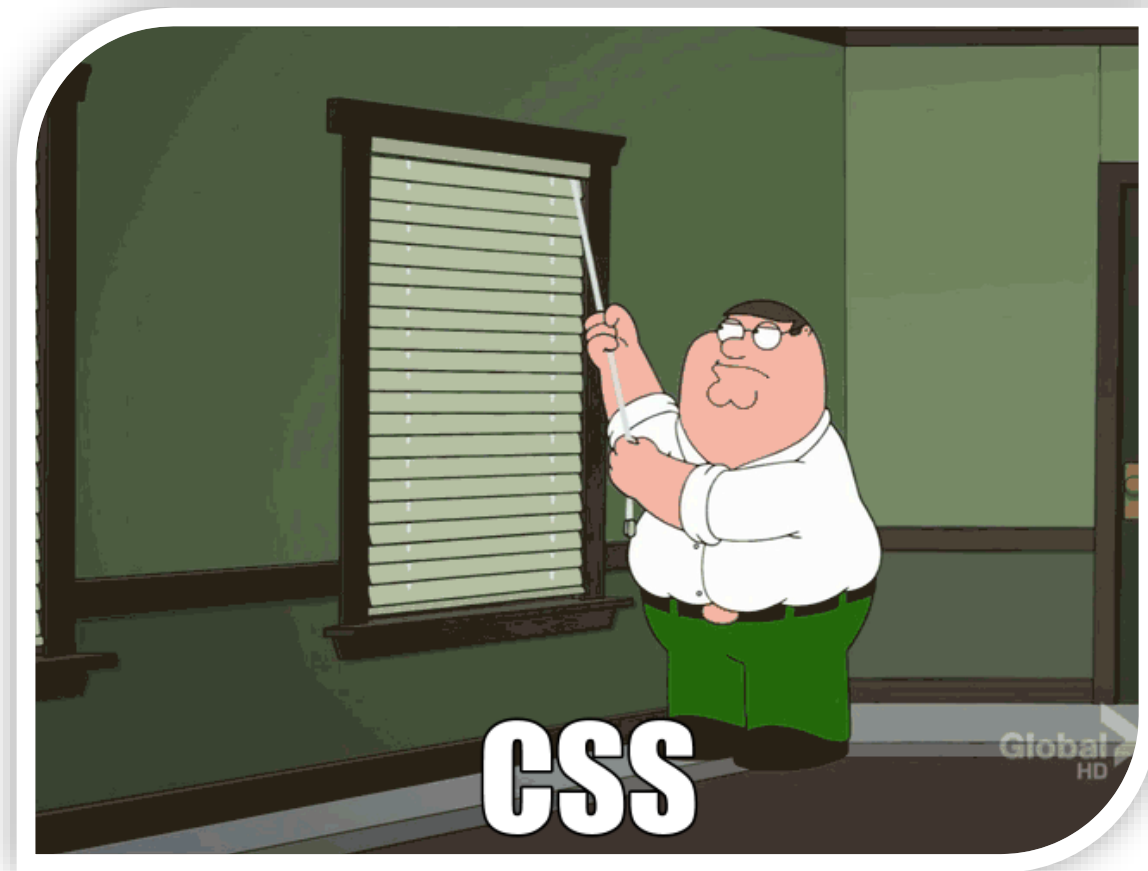
```
export default function Greeting(props) {  
  const { isLoggedIn } = props;  
  return <h1>{isLoggedIn ? 'Welcome back!' : 'Please Sign Up'}</h1>;  
}
```

Conditional rendering

```
export default function Greeting(props) {  
  const { isLoggedIn } = props;  
  if (isLoggedIn) {  
    return <h1>Hello! Welcome back!</h1>;  
  }  
  return <h1>Hello!</h1>;  
}
```

```
export default function Greeting(props) {  
  const { isLoggedIn } = props;  
  return <h1>Hello! {isLoggedIn && 'Welcome back!'}</h1>;  
}
```

Styles



Standartinis css

✓ tournaments

JS index.js

styles.css

```
.container{  
  padding: 100px;  
}
```

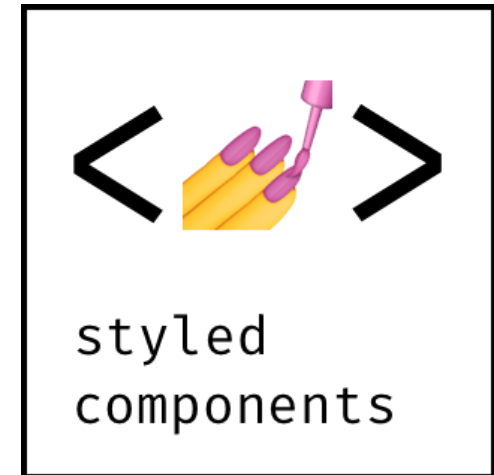
```
import './styles.css';
```

```
return (  
  <div className="container">
```

Styled components

<https://goo.gl/GMitK3>

- ◆ Biblioteka, kuri leidžia naudoti stilizuotus komponentus
- ◆ Css yra rašomas javascript'e
- ◆ Panaikina komponentų ir stiliaus mapinimą
- ◆ Naudoja ecma6



Panaudojimas

```
import styled from 'styled-components';

export const StyledButton = styled.button`
  color:white;
  border: solid black 1px;
  margin:10px;
  background-color: #4267b2;
`;
```

```
▼<button class="style__StyledButton-fZuaZW bwRfdf">
  <span>Get data..</span>
</button>
```

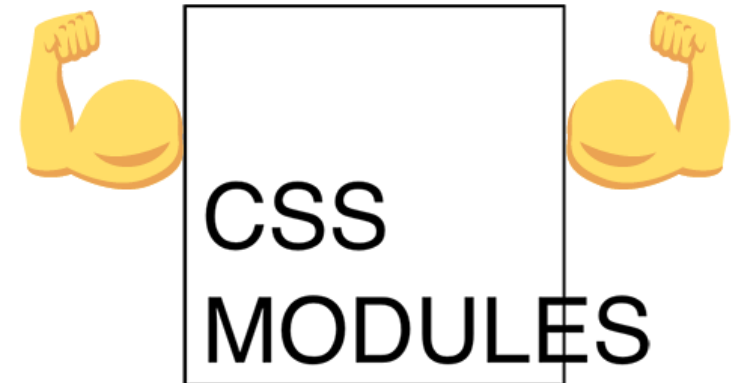

Panaudojimas

```
import {StyledButton} from './style';

function Button(props) {
  return (
    <StyledButton onClick={() => props.click()}>
      <FormattedMessage {...messages.header} />
    </StyledButton>
  );
}
```

CSS modules

- ◆ Failas, kuriame visi css klasių pavadinimai yra lokaliai scoped
- ◆ CSS moduliai kompiliuojami į low-level interchange formatą (ICSS arba Interoperable CSS), bet yra parašyti normaliu css
- ◆ Kai importuojame css modulį gauname objektą, kuris turi visus klasių pavadinimus
- ◆ Create react app jau turi integruotus CSS Modules
- ◆ `styles.module.css`



Panaudojimas

```
.styledButton
{
  color: ■ white;
  border: solid ■ black 1px;
  margin: 10px;
  background-color: ■ #4267b2;
}
```

```
▼ <button class="_2_8T0lG0v5YBAWMVKsy6Vd">
  <span>Get data..</span>
</button>
```

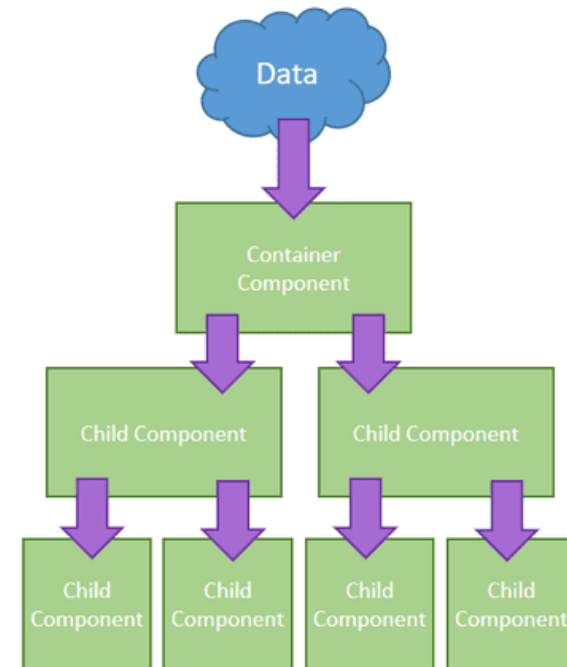
Panaudojimas

```
import styles from './style.css';

function Button(props) {
  return (
    <button className={styles.styledButton} onClick={() => props.click()}>
      <FormattedMessage {...messages.header} />
    </button>
  );
}
```

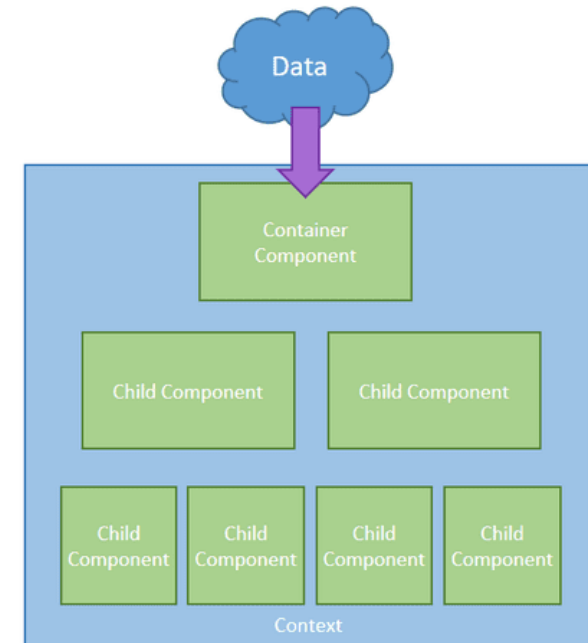
React duomenų srautas, props drilling

- ◆ Duomenys juda viena kryptimi
- ◆ Nuo tėvo -> sūnui
- ◆ Nėra aišku kaip komunikuoti komponentams kurie neturi tiesioginio tėvo – sūnaus ryšio
- ◆ Toks bendravimas yra laikomas bloga praktika



Context duomenų srautas

- ◆ Sukuria tarsi duomenų bloką, kurį gali pasiekti visi komponentai esantys viduje
- ◆ Duomenys tampa globalūs visiems komponentams esantiems tam bloke
- ◆ Nebereikia perdavinėti duomenų per props
- ◆ Bet kuris komponentas esantis viduje Context gali pasiekti duomenis arba juos pakeisti naudojantis funkcijomis
- ◆ useContext hook



API

- ◆ Sukuria duomenų bloko objektą
- ◆ Kai komponentas naudoja Context jis nuskaito dabar esamą reikšmę objekte
- ◆ Priima vienintelį parametą nusakantį pradinę reikšmę

```
const MyContext = React.createContext(defaultValue);
```

Context Provider

- ◆ Komponentas, kuris „apgaubia“ kitus komponentus, kurie naudos duomenis
- ◆ value paduodam objektą duomenims saugoti
- ◆ Jeigu value pasikeičia, komponentai kurie naudoja tą reikšmę persipiešia

```
<MyContext.Provider value={/* some value */}>
```


useContext

- ◆ Hook'sas, kuris priima konteksto objektą bei grąžina dabartinę reikšmę

```
function ThemedButton() {  
  const theme = useContext(ThemeContext);  
  return (  
    <button style={{ background: theme.background, color: theme.foreground }}>  
      I am styled by theme context!  
    </button>  
  );  
}
```

DEMO



Workshop

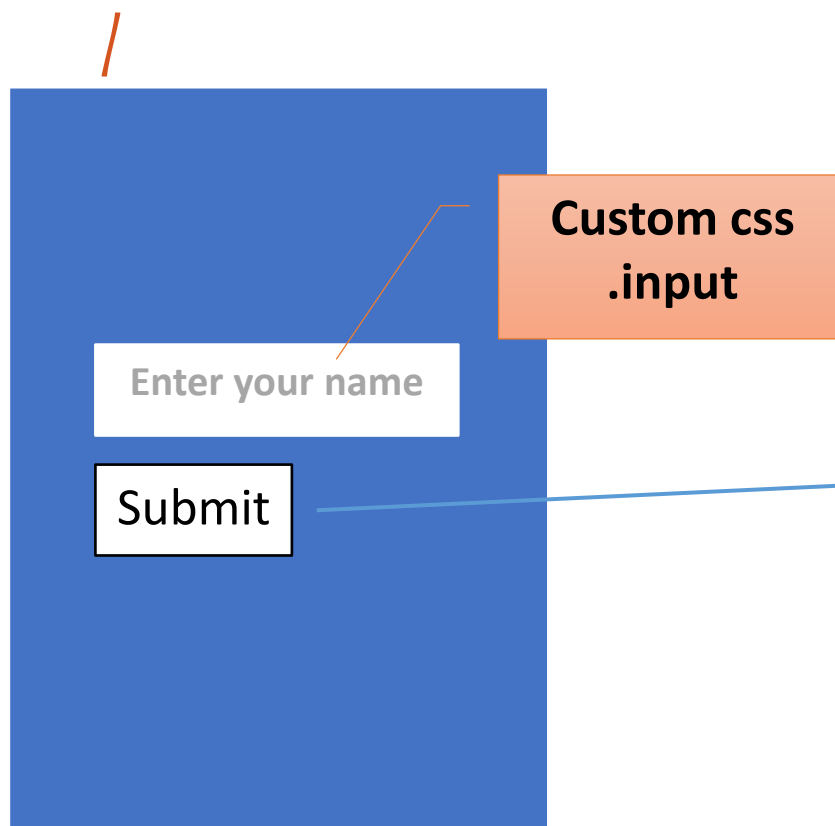
.input -> Su BG spalva ir rounded

.input -> Dropdown be arrow

.dropdown -> Paprastas dropdown su kitokiu font

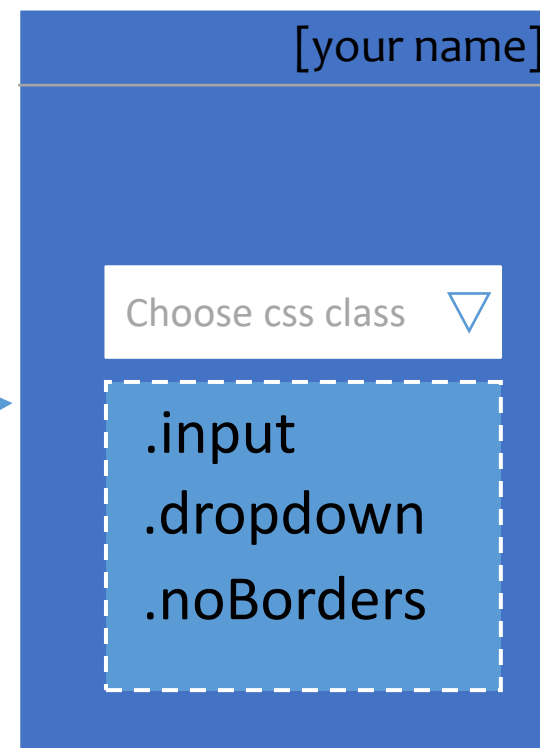
.noBorders -> Paprastas dropdown be borderių

/



A diagram of a web form on a blue background. It contains a white text input field with the placeholder text "Enter your name" and a white "Submit" button below it. An orange box labeled "Custom css .input" has an arrow pointing to the text input field.

/styles/yourname



A diagram of a web form on a blue background. At the top, it says "[your name]". Below that is a white dropdown menu with the text "Choose css class" and a downward arrow. Below the dropdown is a dashed blue box containing the text ".input", ".dropdown", and ".noBorders". A blue arrow points from the "Submit" button in the previous form to this dashed box.

Namų darbai

- ◆ Atsižvelgiam į feedback pasitaikom projektus ir toliau dirbam

