



Тема занятия:

React:

styles,
controlled and
uncontrolled components,
map

Styling Components



1 подход

Обычный CSS

Написание стилей происходит в отдельных CSS файлах и затем они импортируются в ваши компоненты

```
/* styles.css */  
.myComponent {  
  color: blue;  
  font-size: 16px;  
}
```

css файл

```
// MyComponent.js  
import React from 'react';  
import './styles.css'; // Импорт стилей  
  
const MyComponent = () => {  
  return <div className="myComponent">Пример компонента</div>;  
};  
  
export default MyComponent;
```

react
компонент

2 подход

Inline Styles:

Стили указываются напрямую внутри JSX элемента.

```
const MyComponent = () => {  
  return <div style={{ color: 'blue', fontSize: '16px' }}>Пример компонента</div>;  
};
```

```
const MyComponent = () => {  
  const style = {  
    color: 'blue',  
    fontSize: '16px',  
  };  
  
  return <div style={style}>Пример компонента</div>;  
};
```

С помощью
создания
отдельной
переменной

3 подход

CSS-in-JS библиотеки, такой как **styled-components**

Она предоставляет множество возможностей для создания стилей, включая локальную область видимости и использование динамических стилей.

Установка

Для начала, установите styled-components в вашем проекте с помощью npm

```
npm install styled-components
```

3 подход

styled-components

Основная концепция

- styled-components позволяет создавать стилизованные компоненты с использованием синтаксиса тегов.

```
// styles.js
import styled from 'styled-components';

export const MyStyledDiv = styled.div`
  color: blue;
  font-size: 16px;
`;

export const HighlightedDiv = styled.div`
  color: ${(props) => (props.isHighlighted ? 'red' : 'blue')};
  font-size: 16px;
`;
```

styles.js (файл с стилями):

```
// MyComponent.js
import React from 'react';
import { MyStyledDiv, HighlightedDiv } from './styles';

const MyComponent = () => {
  return (
    <>
      <MyStyledDiv>Пример компонента</MyStyledDiv>
      <HighlightedDiv isHighlighted>Подсвеченный компонент</HighlightedDiv>
    </>
  );
};

export default MyComponent;
```

MyComponent.js (компонент,
использующий стили):

Возможности styled-components

1. Вы можете передавать пропсы в styled-components и использовать их для определения стилей.

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

const StyledDiv = styled.div`
  color: ${props => (props.$isHighlighted ? 'red' : 'black')};
  padding: 10px;
  border: 1px solid #ccc;
`;
```

```
const MyComponent = () => {
  return (
    <div>
      <StyledDiv>Normal Text</StyledDiv>
      <StyledDiv $isHighlighted>Highlighted Text</StyledDiv>
    </div>
  );
};
```


Возможности styled-components

2. Расширение стилей

Вы можете расширять стили других styled-components.

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

const BaseButton = styled.button`
  padding: 10px 20px;
`;

const PrimaryButton = styled(BaseButton)`
  background-color: blue;
  color: white;
`;

const SecondaryButton = styled(BaseButton)`
  background-color: white;
  color: blue;
  border: 1px solid blue;
`;
```


Возможности styled-components

3. Использование CSS фрагмента.

Стили, которые прописаны внутри CSS фрагмента могут быть затем использованы внутри стилевых правил

```
import styled, { css } from 'styled-components';

// Создаем css фрагмент
const fontStyles = css`
  font-size: 12px;
  line-height: 14px;
  font-weight: 700;
`;

// Используем стили внутри styled-компонента
const StyledText = styled.span`
  color: blue;
  ${fontStyles} // Включаем созданный CSS-фрагмент
`;
```

Возможности styled-components

4. Глобальные стили

В styled-components глобальные стили могут быть заданы с использованием `createGlobalStyle`. Эта функция создает компонент, который может быть использован для определения стилей, применяемых ко всему приложению.

```
// GlobalStyles.js
import { createGlobalStyle } from 'styled-components';

// Создаем глобальные стили
const GlobalStyles = createGlobalStyle`
  body {
    margin: 0;
    padding: 0;
    font-family: 'Helvetica Neue', sans-serif;
  }

  /* Другие глобальные стили могут быть добавлены здесь */
`;

export default GlobalStyles;
```



Создание глобальных стилей

Возможности styled-components

4. Глобальные стили

```
// App.js
import React from 'react';
import GlobalStyles from './GlobalStyles';

const App = () => {
  return (
    <>
      <GlobalStyles />
      {/* Остальной код вашего приложения */}
    </>
  );
};

export default App;
```

Интеграция глобальных стилей в приложение:

Controlled and uncontrolled components



В React компоненты могут быть разделены на две основные категории:

- контролируемые (controlled)
- неконтролируемые (uncontrolled).

Эти термины относятся к тому, как компонент управляет своим состоянием и данными.



Контролируемые компоненты

Контролируемый компонент - это компонент, который управляет своим состоянием с помощью React.

Любые изменения ввода пользователя или другие события приводят к обновлению состояния компонента через `setState`.

```
import React, { useState } from 'react';

const ControlledComponent = () => {
  const [inputValue, setInputValue] = useState('');

  const handleChange = (e) => {
    setInputValue(e.target.value);
  };

  return (
    <input
      type="text"
      value={inputValue}
      onChange={handleChange}
    />
  );
};
```

Неконтролируемые компоненты

Неконтролируемый компонент - это компонент, в котором состояние не контролируется React.

Вместо этого, данные хранятся в DOM, и доступ к этим данным осуществляется напрямую через ссылки на DOM-элементы.

```
import React, { useRef } from 'react';

const UncontrolledComponent = () => {
  const inputRef = useRef();

  const handleClick = () => {
    alert(`Input value: ${inputRef.current.value}`);
  };

  return (
    <>
      <input type="text" ref={inputRef} />
      <button onClick={handleClick}>Get Value</button>
    </>
  );
};
```

В этом примере **inputRef** представляет собой ссылку на DOM-элемент **<input>**. Значение не хранится в состоянии компонента; вместо этого, при необходимости вы можете получить доступ к значению напрямую через `inputRef.current.value`.

Когда выбирать между контролируруемыми и неконтролируемыми компонентами:

- **Контролируемые компоненты:** Полезны, когда React должен полностью контролировать состояние компонента, особенно при работе с формами. Позволяют React легко управлять вводом и обновлять UI в ответ на изменения.
- **Неконтролируемые компоненты:** Могут быть удобными, когда вам нужно интегрироваться с кодом или библиотеками, которые управляют DOM напрямую. Они также могут уменьшить необходимость в использовании состояния и `useState`.