

Цели занятия

- рассмотреть разные способы работы со стилями в React;
- поверхностно изучить со Styled Components;
- познакомиться с Material UI;

Полезные материалы

- [Использование CSS Modules в React](#)
- [Документация Styled Components](#)
- [Документация Material UI](#)

Разные способы работы со стилями

React это современная библиотека для разработки интерфейсов, которая поддерживает несколько разных способов для работы со стилями внутри компонентов. Каждый из них имеет свои плюсы и минусы и может использоваться для определенной задачи, единственно правильно варианта не существует

1. В виде обычного CSS-файла

Это самый привычный и самый часто используемый способ, который мы с вами используем на протяжении всего курса. В данном способе каждый компонент подключает отдельный css-файл, который хранит необходимые стили, а в разметку добавляются className.

```
// Подключаем файл со стилями
import './Example.css';

function Example() {
```

```

return (
  <div className="example">
    <div className="example__text">Hello, Frontenders!</div>
  </div>
);
}

export default Example;

```

2. Используем inline-стили

В некоторых случаях нам нужно вставить внутрь CSS какое-то значение, которое хранится только внутри JavaScript. Например, мы получаем цвет от бэкенда и нам нужно отобразить текст с указанным цветом — тогда мы вынуждены использовать inline-стили.

```

// Подключаем файл со стилями
import './Example.css';

function Example(props) {
  const { color } = props; // Мы получаем цвет в виде свойства

  return (
    <div
      className="example"
      // Добавляем inline-стили
      style={{
        color: `${color}`,
        // Здесь все свойства пишутся в camelCase
        fontSize: "14px",
      }}
    >
      <div className="example__text">Hello, Frontenders!</div>
    </div>
  );
}

```

3. Используем CSS Modules

В больших проектах может быть несколько сотен компонентов и становится тяжелее следить за тем, чтобы все классы для компонентов не пересекались и были уникальными, чтобы не получить конфликт стилей. Для этого была создана технология CSS Modules, которая автоматически создаёт уникальные классы.

В данном случае стили всё ещё хранятся в отдельном файле, но теперь его расширение должно быть `.module.css`. Во время импорта такого файла мы получим объект с ключами, которые можно вставлять внутрь `className`, чтобы применить стили.

```
// Теперь мы импортируем файл в переменную
import styles from './Example.module.css';

function Example() {
  return (
    // Вместо строк теперь используем ключи из этой переменной
    // Каждый ключ соответствует классу в CSS-файле
    // В разметке вместо ключа будет сгенерировано уникальное
    значение
    <div className={styles.example}>
      <div className={styles.text}>Hello, Frontenders!</div>
    </div>
  );
}

export default Example;
```

4. Используем Styled Components

А вот этот способ уже придуман не разработчиками React, а отдельными людьми, которые написали свой инструмент. Его можно установить через NPM, а с подробной документацией можно ознакомиться по следующей [ссылке](#).

Его основная идея заключается в том, что мы пишем все стили сразу внутри JSX-файла и вообще не используем для этого классы — библиотека сама сгенерирует эти классы, когда компонент будет вставлен на страницу.

```
// Используем отдельную библиотеку
import styled from "styled-components";

// Создаем отдельные части компонента
// и сразу пишем нужные стили
const Container = styled.div`
  padding: 40px;
  border: 2px solid red;
`;

const Text = styled.div`
  color: red;
  font-size: 14px;
`;

// Собираем из этих частей компонент
function Example() {
  return (
    <Container>
      <Text>Hello, Frontenders!</Text>
    </Container>
  );
}
```

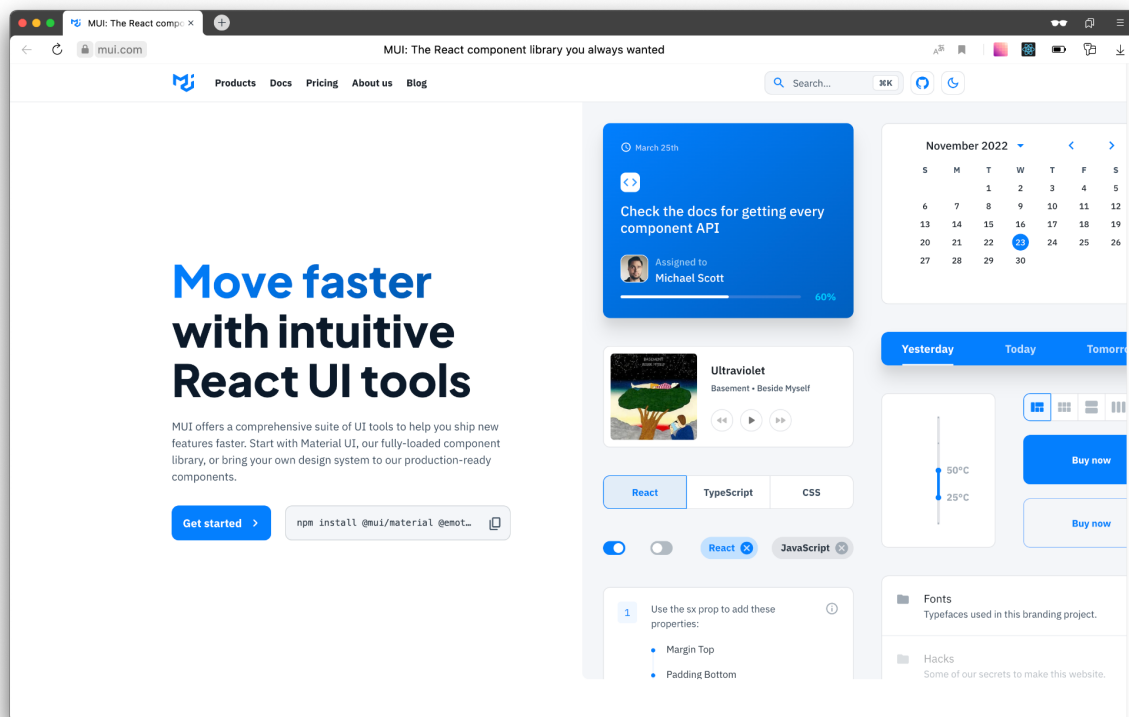
Material UI

В сообществе frontend-разработчиков существует огромное количество инструментов и готовых решений, которые вы можете установить себе в проект и использовать по мере необходимости.

Например, есть целые готовые дизайн-системы, которые имеют набор React-компонентов для многих повторяющихся элементов интерфейса, как например библиотека [Material UI](https://material-ui.com/)

Их использование зависит от конкретного проекта и в некоторых случаях может сэкономить время на разработке (если вас устраивают готовые компоненты), но не стоит тратить время на изучение подобных проектов без определенной задачи или цели.

Как правило, в крупных проектах разработчики самостоятельно разрабатывают подобную библиотеку, которая будет отвечать всем требованиям именно этого проекта, а не пытаться подогнать чужую работу под свои критерии.



На онлайн занятии мы попробуем установить эту библиотеку и используем несколько готовых компонентов из неё, чтобы вживую посмотреть какие возможности она предоставляет. Приходите!