React - CSS (стили)

Inline Styles (Инлайн-стили)
Подключение SASS к проекту
Обычные CSS-файлы + SASS-файлы
CSS Modules + SASS Modules
Styled Components
react - bootstrap (UI - библиотеки)
Своя UI библиотека
tailwind

Inline Styles (Инлайн-стили)

Стили можно прописывать прямо в JSX с помощью объекта style

Этот метод позволяет динамически изменять стили на основе пропсов или состояния

Первое это то, что тут две фигурные скобки. То, что мы рендирим — написано на JSX и для JS выражений, которые будут использоваться в JSX, они должны быть вставлены в фигурные скобки.

Первые фигурные скобки вставляют JavaScript в JSX. Внутренние фигурные скобки создают объект литерал. Стили передаются как объект литералы к элементу.

```
function App() {
  return (
    <div style={{ backgroundColor: '#f0f0f0', padding: '20px' }}>
      {/* содержимое компонента */}
    </div>
  );
}
```

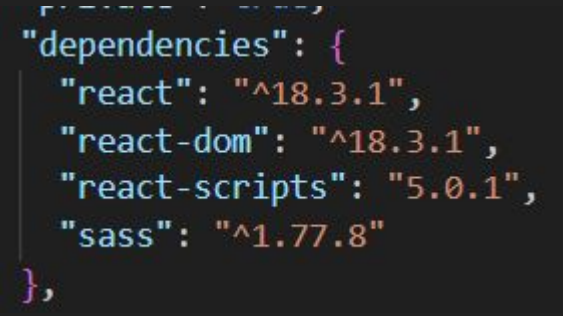
Подключение SASS к проекту

команда

npm install sass

В результате вызова данной команды в проект добавляется зависимость SASS библиотеки (модуля), что можно визуально увидеть в файле package.json

SASS нужен для корректной компиляции SCSS/SASS файлов в проекте, что позволит пользоваться SCSS/SASS в проекте



Обычные CSS-файлы + SASS-файлы

Стили можно писать в обычных CSS-файлах и подключать их в компоненты. Например, если у вас есть файл App.css, его можно импортировать в компоненте

Классы являются глобальными и могут конфликтовать друг с другом

```
import './App.css';

function App() {
  return (
    <div className="app-container">
      {/* содержимое компонента */}
    </div>
  );
}
```

CSS Modules + SASS Modules

CSS-модули позволяют создавать локальные стили для компонентов, чтобы избежать конфликтов имен классов. Файлы именуются с расширением .module.css, а затем импортируются как объект

```
import styles from './App.module.css';

function App() {
  return (
    <div className={styles.appContainer}>
      {/* содержимое компонента */}
    </div>
  );
}
```

Styled Components (CSS in JS)

Это библиотека, которая позволяет писать стили на JavaScript с использованием шаблонных строк. Это популярный подход для CSS-in-JS

Помимо styled-components, существуют и другие библиотеки для стилизации, такие как Emotion, JSS, и Aphrodite, которые также предоставляют возможности для написания CSS на JavaScript.

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

const Container = styled.div`
  background-color: #f0f0f0;
  padding: 20px;
`;

function App() {
  return (
    <Container>
      {/* содержимое компонента */}
    </Container>
  );
}
```

UI библиотеки

UI библиотеки в React — это наборы готовых компонентов, стилей и утилит, которые помогают разработчикам быстрее создавать пользовательские интерфейсы. Эти библиотеки включают в себя готовые элементы интерфейса, такие как кнопки, формы, модальные окна, карточки, таблицы и другие часто используемые компоненты, которые можно легко интегрировать в ваше приложение.

Зачем использовать UI библиотеки?

- **Ускорение разработки:** UI библиотеки предоставляют набор готовых компонентов, которые можно использовать сразу, что значительно ускоряет процесс разработки. Вместо того чтобы разрабатывать компоненты с нуля, вы можете использовать уже готовые решения.
- **Единообразие дизайна:** UI библиотеки обеспечивают консистентность дизайна в вашем приложении. Все компоненты библиотеки выполнены в одном стиле, что упрощает создание интерфейсов с единым внешним видом.
- **Лучшие практики:** В UI библиотеках обычно уже реализованы лучшие практики по разработке интерфейсов, такие как адаптивность, доступность и кроссбраузерная совместимость.
- **Поддержка и обновления:** Популярные UI библиотеки регулярно обновляются и поддерживаются сообществом или компаниями-разработчиками, что позволяет использовать актуальные технологии и решения.

React Bootstrap

Как подключить:

npm install react-bootstrap bootstrap

Оф.сайт (и документация) - https://react-bootstrap.netlify.app

```
import Button from 'react-bootstrap/Button';

function App() {
  return <Button variant="primary">Click me</Button>;
}
```

Material UI

Как подключить:

npm install @mui/material @emotion/react @emotion/styled

Оф.сайт (и документация) - https://mui.com/material-ui/getting-started/

```
import Button from '@mui/material/Button';

function App() {
  return <Button variant="contained">Click me</Button>;
}
```

chakra-ui

Как подключить:

npm i @chakra-ui/react @emotion/react @emotion/styled framer-motion

Оф.сайт (и документация) - https://v2.chakra-ui.com

```
import { Button } from "@chakra-ui/react";

function App() {
  return <Button colorScheme="blue">Click me</Button>;
}
```

Что такое StoryBook

Storybook — это инструмент для разработки изолированных компонентов пользовательского интерфейса (UI) и их последующей документации. Он позволяет разработчикам создавать и тестировать UI-компоненты вне контекста основного приложения, предоставляя удобный интерфейс для взаимодействия с компонентами и их состояния.

Своя UI библиотека

Своя UI библиотека — это набор компонентов и стилей, созданных для повторного использования в различных проектах. Она обеспечивает унифицированный интерфейс и помогает поддерживать консистентность дизайна.

Зачем создавать свою UI библиотеку:

- **Унификация:** Обеспечивает единый стиль и поведение компонентов в разных приложениях.
- **Повторное использование:** Упрощает разработку, позволяя повторно использовать готовые компоненты.
- **Контроль и кастомизация:** Позволяет адаптировать и расширять компоненты под специфические нужды проекта.

Плюсы:

- **Повышение производительности:** Сокращает время на разработку и тестирование, используя проверенные компоненты.
- **Согласованность:** Гарантирует единообразие и соблюдение стандартов дизайна в разных проектах.
- **Кастомизация:** Позволяет адаптировать и оптимизировать компоненты под собственные требования.

Минусы:

- **Начальные затраты:** Требует времени и усилий на создание и тестирование компонентов.
- **Поддержка:** Необходимо поддерживать и обновлять библиотеку по мере изменений в требованиях и технологиях.
- **Совместимость:** Может потребоваться дополнительная работа для интеграции с другими библиотеками или фреймворками.

Своя UI библиотека - инструкция

- Инициализируйте новый проект Node.js: ***npm init -y***
- Добавляем главные зависимости:
  npm install react react-dom
  npm install --save-dev typescript @types/react @types/react-dom
- Создайте файл tsconfig.json
- Создайте директорию src и добавьте туда первый компонент, например Button.tsx
- Настройка сборки с помощью Rollup
  npm install --save-dev rollup rollup-plugin-typescript2 @rollup/plugin-node-resolve @rollup/plugin-commonjs
- создать rollup.config.mjs
- В src/index.ts экспортируйте компоненты
- npm run build - собрать библиотеку
- npm publish - опубликовать библиотеку

Конфигурация package.json файла

Конфигурация package.json управляет зависимостями, скриптами и метаданными проекта. Она упрощает установку и обновление зависимостей, автоматизацию задач и публикацию пакета. Также она помогает настроить инструменты сборки и другие конфигурации проекта.

Конфигурация файла rollup.config.mjs

Файл rollup.config.mjs содержит настройки сборщика Rollup для компиляции и упаковки вашего кода. Он определяет входные и выходные точки, используемые плагины и формат сборок. Конфигурация в формате ES модулей (с расширением .mjs) позволяет использовать синтаксис import и export для настройки сборщика.

JS rollup.config.mjs > ...

```mjs
import typescript from "rollup-plugin-typescript2";
import resolve from "@rollup/plugin-node-resolve";
import commonjs from "@rollup/plugin-commonjs";

export default {
  input: "src/index.ts",
  output: [
    {
      file: "dist/index.js",
      format: "cjs",
      sourcemap: true,
    },
    {
      file: "dist/index.es.js",
      format: "es",
      sourcemap: true,
    },
  ],
  plugins: [
    resolve(),
    commonjs(),
    typescript({ useTsconfigDeclarationDir: true }),
  ],
  external: ["react", "react-dom"],
};
```

Конфигурация tsconfig.json для настройки TypeScript-а

tsconfig.json > ...

```json
{
  "compilerOptions": {
    "moduleResolution": "node",
    "target": "ES5",
    "module": "ESNext",
    "declaration": true,
    "outDir": "dist",
    "strict": true,
    "jsx": "react-jsx",
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true,
  },
  "include": ["src/**/*"],
  "exclude": ["node_modules", "dist"]
}
```

Файл tsconfig.json настраивает компилятор TypeScript, указывая, какие файлы включать в сборку и как их компилировать. Он определяет параметры компиляции, такие как целевая версия JavaScript, модули и проверки типов. Конфигурация помогает управлять проектом, обеспечивая правильную обработку и преобразование кода.

Использование своей библиотеки

Открыть страницу публикации библиотеки и следовать инструкции

Обычно команда ***npm install <название библиотеки>***

Далее после установки библиотеки в проект использовать код

import {<компоненты библиотеки>} from “<название библиотеки>”

TailWind

Tailwind CSS — это утилитарный CSS-фреймворк для быстрого создания современных веб-интерфейсов. В отличие от традиционных CSS-фреймворков, таких как Bootstrap, Tailwind не предоставляет готовые компоненты, а вместо этого предлагает набор низкоуровневых утилитарных классов, которые можно комбинировать для создания любого дизайна.

Оф.сайт (и документация) - https://tailwindcss.com/docs/guides/create-react-app

Инструкция по установке на проект:

- Создаем react проект - ***npx create-react-app my-project***
- Устанавливаем глобально TailWind - ***npm install -D tailwindcss***
- Генерируем TailWindconfig file - ***npx tailwindcss init***
- Настраиваем данный конфиг под свой проект (под себя)
- Добавляем в главный файл css (index.css) в проекте ***@tailwind base; @tailwind components; @tailwind utilities;***
- Пользуемся TailWind-ом (встроенными классами)

Ресурсы

Пример кода (стили) - ТЫК

Деплой примера (стили) - ТЫК

Стили в react - ТЫК

Статья “Способы стилизации в React” - ТЫК

Styled Components - ТЫК

Пример UI библиотеки - ТЫК

Исходник UI библиотеки - ТЫК

Styled Components (EN) - ТЫК

Добавление стилей (EN) - ТЫК

Добавление CSS модулей (EN) - ТЫК

Добавление SASS на проект (EN) - ТЫК

react bootstrap (EN) - ТЫК

chakra UI (EN) - ТЫК

material UI (EN) - ТЫК

tailwind (EN) - ТЫК