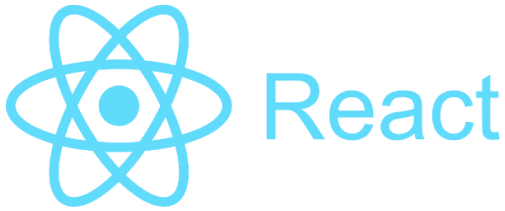


React Essential

Стилизация в React

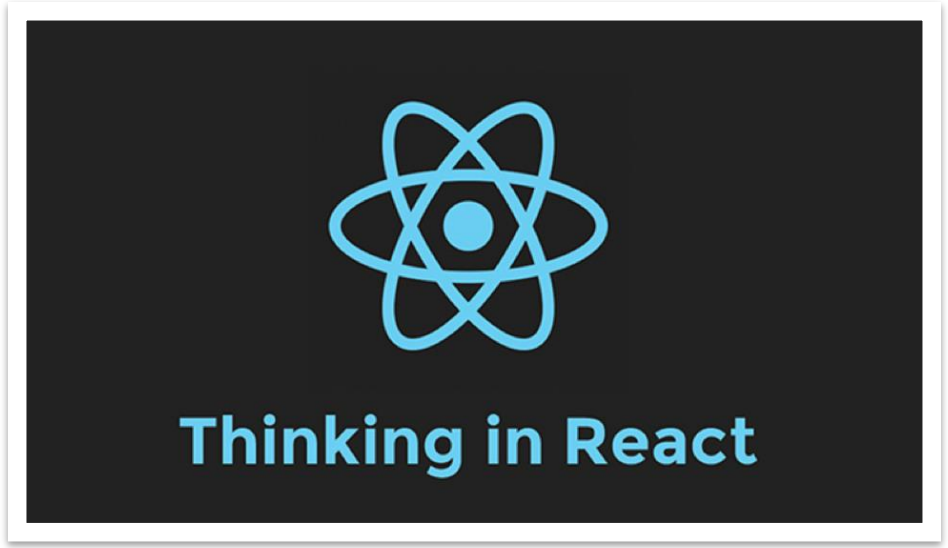
Introduction

Муляк Дмитрий

Front-end developer at GlobalTechMakers

dmitriymuliak

dmitriymuliak



Тема

Стилизация в React

Стилизация в React

1. Обзор способов стилизации компонентов
2. Inline styles
3. Подключение css
4. Динамические классы
5. CSS модули

Обзор способов стилизации компонентов

React дает свободу действий разработчикам - какой способ стилизации применить в проекте.

Существует много способов стилизации, вот несколько из них:

- Inline CSS
- CSS in JS (for example styled-components)
- CSS Modules
- Sass, SCC, Stylus
- PostCSS

[9 CSS in JS Libraries]

Inline styles

Атрибут style принимает JavaScript-объект со свойствами в camelCase вместо CSS-строк.

Такой подход повышает эффективность и защищает от XSS.

```
const divStyle = {
  color: 'blue',
  backgroundImage: 'url(' + imgUrl + ')',
};

function HelloWorldComponent() {
  return <div style={divStyle}>Привет, мир!</div>;
}
```

Браузерные префиксы и единицы измерений

Обратите внимание, что браузерные префиксы не добавляются к стилям автоматически. Для поддержки старых браузеров необходимо добавить соответствующие свойства стиля.

```
const divStyle = {
  WebkitTransition: 'all', // обратите внимание на заглавную букву W
  msTransition: 'all' // 'ms' — единственный префикс в нижнем регистре
};

function ComponentWithTransition() {
  return <div style={divStyle}>Это будет работать в разных браузерах</div>;
}
```

Ключи стилей записываются в camelCase для обеспечения доступа к аналогичным свойствам DOM из JS (например, node.style.backgroundImage). Префиксы браузера, кроме ms, должны начинаться с заглавной буквы. Вот почему WebkitTransition начинается с заглавной «W».

React автоматически добавит суффикс «px» к свойствам стилей с числовым значением. Если вы хотите использовать единицы измерения, отличные от «px», укажите значение в виде строки с желаемой единицей измерения.

```
// Результат: '10px'
<div style={{ height: 10 }}>
  Привет, мир!
</div>

// Результат: '10%'
<div style={{ height: '10%' }}>
  Привет, мир!
</div>
```

Подключение css

Create React App проект использует Webpack для обработки всех assets. Webpack предлагает собственный способ «расширить» концепцию импорта за пределы JavaScript. Чтобы выразить, что файл JavaScript зависит от файла CSS, вам необходимо импортировать CSS из файла JavaScript:

Это не обязательно для React, но многие считают эту функцию удобной. Вы можете прочитать о преимуществах этого подхода здесь.

В процессе разработки такое выражение зависимостей позволяет перезагружать ваши стили по мере их редактирования. В production сборке все CSS-файлы будут объединены в один уменьшенный файл .css .

Button.css

```css
.Button {
  padding: 20px;
}
```

Button.js

```js
import React, { Component } from 'react';
import './Button.css'; // Tell webpack that Button.js uses these styles

class Button extends Component {
  render() {
    // You can use them as regular CSS styles
    return <div className="Button" />;
  }
}
```

create-react-app: adding-a-stylesheet

Динамические классы

Вы можете формировать значения для атрибута «className» на лету. Так как это JSX - вы можете использовать тернарный или логические операторы для формирования значения атрибута className.

```
import React from "react"
import containerStyles from "./container.module.css"

export default function Container({ children }) {
  return (
    <section className={`container ${containerStyles.container}`}>
      {children}
    </section>
  )
}
```

Такие библиотеки как Styled Components (css-in-js подход) имеют свои пути решения добавления динамических стилей, так как они формируются на лету - https://styled-components.com

```
const Avatar = styled.div`
  display: inline-block;
  ${props => props.primary && css`
    background: white;
    color: palevioletred;
  `}
`;

render(
  <div>
    <Avatar primary>Name</Avatar>
  </div>
)
```

Так как React оставляет за разработчиками выбор того, какой способ стилизации применять в проекте - вы также можете использовать CSS Custom Properties

clsx для динамического className

Есть множество библиотек для условного построения строк className. Одна из популярных это clsx с небольшим весом (229B).

```
import clsx from 'clsx';

// Strings (variadic)
clsx('foo', true && 'bar', 'baz');
//=> 'foo bar baz'

// Objects
clsx({ foo:true, bar:false, baz:isTrue() });
//=> 'foo baz'

// Objects (variadic)
clsx({ foo:true }, { bar:false }, null, { '--foobar':'hello' });
//=> 'foo --foobar'

// Arrays
clsx(['foo', 0, false, 'bar']);
//=> 'foo bar'

// Arrays (variadic)
clsx(['foo'], ['', 0, false, 'bar'], [['baz', [['hello'], 'there']]]);
//=> 'foo bar baz hello there'
```

CSS модули

CSS Модули позволяют использовать одно и то же имя класса CSS в разных файлах, не беспокоясь о конфликтах имен.

Create React App поддерживает [css modules,](#) наряду с обычными таблицами стилей, используя соглашение об именах файлов [name] .module.css. CSS Модули позволяют инкапсулировать CSS, автоматически создавая уникальное имя класса в формате [имя файла]_[имя класса]__[хэш].

Button.module.css

```
.error {
  background-color: red;
}
```

Another-stylesheet.css

```
.error {
  color: red;
}
```

Button.js

```
import styles from './Button.module.css'; // Import css modules stylesheet
import './another-stylesheet.css'; // Import regular stylesheet

class Button extends Component {
  render() {
    // reference as a js object
    return <button className={styles.error}>Error Button</button>;
  }
}
```

Result in DOM

```
<!-- This button has red background but not red text -->
<button class="Button_error_ax7yz">Error Button</button>
```

Проверка знаний

TestProvider.com

Проверьте как Вы усвоили данный материал на TestProvider.com

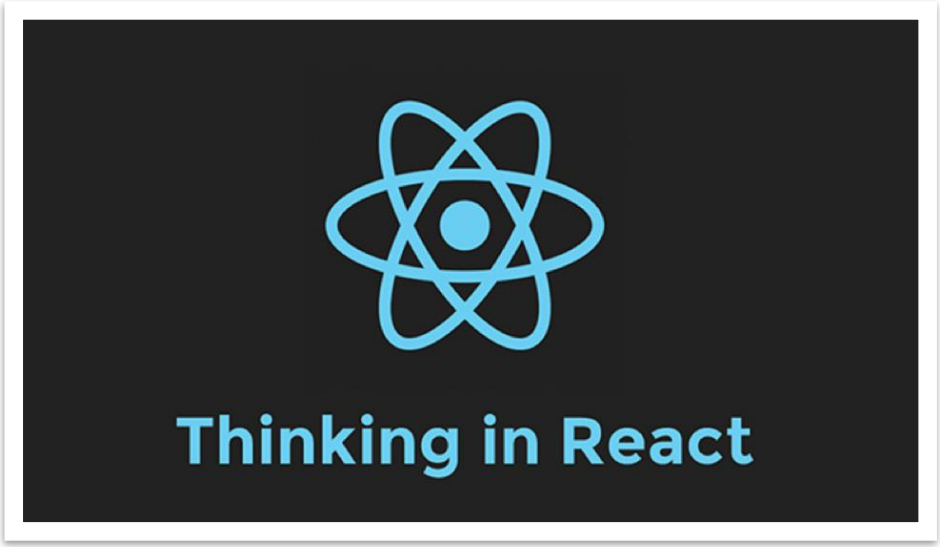
TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Спасибо за внимание! До новых встреч!

Муляк Дмитрий

Front-end Developer at GTM



Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association