МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КАЛУЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. К. Э. ЦИОЛКОВСКОГО»

Инженерно-технологический институт
кафедра информатики и информационных технологий

О Т Ч Е Т

ДИСЦИПЛИНА: "Прикладное программирование"
Выполнил: студент гр. Б-ИСиТ-41 Варавин Е.А._____
Проверил: Винокуров А.В._____

Дата сдачи (защиты) лабораторной работы:
Результаты сдачи (защиты):
Количество рейтинговых баллов
Оценка

Калуга, 2021 г.

Библиотека Styled Components

Введение

Styled Components являются одним из новых способов использования CSS в современном JavaScript. Они предназначен для написания CSS, который ограничен одним компонентом.

Что важно понять о Styled Components, так это то, что его название следует понимать буквально. Ты больше не стилизуешь элементы HTML на основе их класса

```
<button className="btn">Click</button>
```

```
button.btn {
  font-size: 2em;
  background-color: black;
  color: white;
}
```

Вместо этого ты определяешь стилизованные компоненты, которые имеют свои собственные инкапсулированные стили

```
const Button = styled.button`
  font-size: 2em;
  background-color: black;
  color: white;
`;
```

Styled Components позволяют писать простой CSS в компонентах, не беспокоясь о конфликтах имен классов.

Styled Components используют шаблонные строки (template literals) для стилизации компонентов; это простой JavaScript и способ передачи аргумента функции.

Когда ты определяешь свои стили, на самом деле ты создаешь нормальный React компонент, к которому прикреплены твои стили.

Импортировав styled объект, можно начать создавать стилизованные компоненты.

```
import styled from "styled-components";

const Button = styled.button`
  // тут стили
`;

render(
  <div>
    <Button> Button </Button>
  </div>,
);
```

Кастомизация компонентов с помощью props

Когда ты передаешь props в Styled Component, то они передаются нижестоящим DOM узлам.

```
const Input = styled.input`
  font-size: 18px;
  padding: 10px;
  margin: 10px;
  background: papayawhip;
  border: none;
  border-radius: 3px;
  ::placeholder {
    color: palevioletred;
  }
`;

render(
  <div>
    <Input type="text" placeholder="Name" />
    <Input type="text" placeholder="Last Name" />
  </div>,
);
```

Он, вставит эти props в качестве атрибутов HTML. Также, на основе значений этих props, можно кастомизировать компонент.

```
const Button = styled.button`
  color: ${(props) => (props.primary ? "white" : "palevioletred")};
  background: ${(props) => (props.primary ? "palevioletred" : "white")};
  font-size: 20px;
  margin: 10px;
  padding: 5px 20px;
  border: 2px solid palevioletred;
  border-radius: 3px;
`;

render(
  <div>
    <Button> Normal Button </Button>
    <Button primary> Primary Button </Button>
  </div>,
);
```

Расширение стилей компонента

Если у тебя есть один компонент, и ты хочешь создать аналогичный, поменяв немного стили, ты можешь просто поместить его в конструктор styled() и он унаследует стили другого

```
const Button = styled.button`
  color: palevioletred;
  font-size: 20px;
  margin: 10px;
  padding: 5px 20px;
  border: 2px solid palevioletred;
  border-radius: 3px;
`;

// Новый компонент на основе Button, но с новыми стилями
const TomatoButton = styled(Button)`
  color: tomato;
  border-color: tomato;
`;

render(
  <div>
    <Button> Normal Button </Button>
    <TomatoButton>Tomato Button</TomatoButton>
  </div>,
);
```

В некоторых случаях может возникнуть необходимость изменить какой-то тег, например, при построении панели навигации, где есть несколько якорных ссылок и кнопок, но они должны быть стилизованы одинаково.

Ты можешь использовать полиморфный prop as, чтобы динамически поменять элемент, который получает твои стили.

```
const Button = styled.button`
  font-family: monospace;
  display: inline-block;
  color: palevioletred;
  font-size: 18px;
  margin: 20px;
  padding: 10px 20px;
  border: 2px solid palevioletred;
  border-radius: 3px;
  display: block;
  text-decoration: none;
`;

const TomatoButton = styled(Button)`
  color: tomato;
  border-color: tomato;
`;

render(
  <div>
    <Button>Normal Button</Button>
    <Button as="a" href="/">
      Link with Button styles
    </Button>
    <TomatoButton as="a" href="/">
      Link with Tomato Button styles
    </TomatoButton>
  </div>,
);
```

Это также работает и с кастомными компонентами

```
const Button = styled.button`
  // тут стили
`;

const ReversedButton = (props) => (
  <button {...props} children={props.children.split("").reverse()} />
);

render(
  <div>
    <Button>Normal Button</Button>
    <Button as={ReversedButton}>Custom Button with Normal Button styles</Button>
  </div>,
);
```

Стилизация любого компонента

Метод styled() отлично работает на всех собственных или сторонних компонентах, если они используют переданный prop className элементу DOM.

```
// Это может быть ссылка Link с react-router-dom, например:
const Link = ({ className, children }) => (
  <a className={className}>{children}</a>
);

const StyledLink = styled(Link)`
  color: palevioletred;
  font-weight: bold;
`;

render(
  <div>
    <Link>Unstyled, boring Link</Link>
    <br />
    <StyledLink>Styled, exciting Link</StyledLink>
  </div>,
);
```

Псевдоэлементы, псевдо-селекторы и вложенность

Псевдо-селекторы и псевдоэлементы без дальнейшей доработки автоматически присоединяются к компоненту

```
const Thing = styled.button`
  color: blue;

  ::before {
    content: "🚀";
  }

  :hover {
    color: red;
  }
`;

render(<Thing>Hello world!</Thing>);
```

Для более сложных шаблонов, амперсанд (&) может использоваться для обращения к основному компоненту. Если ты используешь селекторы без амперсанда, они будут ссылаться на дочерние элементы компонента.

Также, амперсанд может быть использован для повышения специфичности правил для компонента; это может быть полезно, если ты имеешь дело со смесью стилевых компонентов и vanilla CSS, где могут быть конфликтующие стили

```
const Thing = styled.div`
  && {
    color: blue;
  }
`;

const GlobalStyle = createGlobalStyle`
  div${Thing} {
    color: red;
  }
`;

render(
  <React.Fragment>
    <GlobalStyle />
    <Thing>I'm blue, da ba dee da ba daa</Thing>
  </React.Fragment>,
);
```