Styled Components lib

Если в проекте мы используем простые css файлы, без модулей, имена классов будут применяться ко всем компонентам в проекте. В этом заключается проблема. Есть несколько путей решения этой проблемы. Одна из них – использование библиотеки styled components.

Что делает эта либа? Генерирует множество уникальных className, и создает глобальный css файл.

**npm install --save styled-components**

Чтобы использовать эту либу импортируем:

import styled from 'styled-components'

Базовый синтаксис: мы вызываем библиотеку styled и метод, соответствующий элементу – div, h2, button и пр. Стиль прописываем в обратных кавычках.

Родителя в стилях заменяем на & при применении псевдо селекторов.

import styled from 'styled-components'

import './Button.css';

*const* Button = styled.button`

  font: inherit;

  padding: 0.5rem 1.5rem;

  border: 1px solid #8b005d;

  color: white;

  background: #8b005d;

  box-shadow: 0 0 4px rgba(0, 0, 0, 0.26);

  cursor: pointer;

  &:focus {

    outline: none;

  }

  &:hover,

  &:active {

    background: #ac0e77;

    border-color: #ac0e77;

    box-shadow: 0 0 8px rgba(0, 0, 0, 0.26);

  }

`

..ниже обычный реакт компонент:

// const Button = props => {

//   return (

//     <button type={props.type} className="button" onClick={props.onClick}>

//       {props.children}

//     </button>

//   );

// };

export default Button;

Динамическая передача пропсов с Stуled Components

**Вариант 1.**

Для динамической передачи стилей (поменять цвет кнопки при нажатии) нужно просто в созданный с помощью styled component компонент передать className.

Для начала создадим стилизованный компонент из div

import styled from 'styled-components'

*const* FormControl = styled.div`

  margin: 0.5rem 0;

  & label {

    font-weight: bold;

    display: block;

    margin-bottom: 0.5rem;

  }

  & input {

    display: block;

    width: 100%;

    border: 1px solid #ccc;

    font: inherit;

    line-height: 1.5rem;

    padding: 0 0.25rem;

  }

  & input:focus {

    outline: none;

    background: #fad0ec;

    border-color: #8b005d;

  }

  &.invalid input {

    border-color: red;

    background-color: #ffd7d7;

  }

  &.invalid label {

    color: red;

  }

`

*const* CourseInput = *props* *=>* {

*const* [enteredValue, setEnteredValue] = useState('');

*const* [isValid, setIsValid] = useState(true)

*const* goalInputChangeHandler = *event* *=>* {

  console.log(isValid);

  if (*event*.target.value.trim().length > 0) {

    setIsValid(true)

  }

  setEnteredValue(*event*.target.value)

}

*const* formSubmitHandler = *event* *=>* {

*event*.preventDefault();

    if(enteredValue.trim().length === 0) {

      setIsValid(false)

    }

*props*.onAddGoal(enteredValue);

  };

  return (

    <form onSubmit={formSubmitHandler}>

      <*FormControl*>

        <label >Course Goal</label>

        <input type="text" onChange={goalInputChangeHandler} />

        </*FormControl*>

      <*Button* type="submit">Add Goal</*Button*>

    </form>

  );

};

export default CourseInput;

Далее в див передадим условие со стилями:

      <*FormControl* className={!isValid && 'invalid'}>

        <label >Course Goal</label>

        <input type="text" onChange={goalInputChangeHandler} />

        </*FormControl*>

**Вариант 2.**

Также можно прописывать условия прямо в кавычках при создании стилизованных компонент в колл-беке.

      <*FormControl* invalid={!isValid}> // Имя пропса задается любое

        <label >Course Goal</label>

        <input type="text" onChange={goalInputChangeHandler} />

        </*FormControl*>

*const* FormControl = styled.div`

  margin: 0.5rem 0;

  & label {

    font-weight: bold;

    display: block;

    margin-bottom: 0.5rem;

    color: ${*props=>*(*props*.invalid? 'red' : 'black')}

  }

  & input {

    display: block;

    width: 100%;

    border: 1px solid ${*props=>* (*props*.invalid? 'red' : '#ccc')};

    background: ${*props=>*(*props*.invalid? '#ffd7d7':'transparent')};

    font: inherit;

    line-height: 1.5rem;

    padding: 0 0.25rem;

  }

  & input:focus {

    outline: none;

    background: #fad0ec;

    border-color: #8b005d;

  }

`

`

Лишние стили удалены.

Styled Components & Media Queries

Медиа запросы вписываются прямо в стили при создании стилизованного компонента.

*const* Button = styled.button`

  width: 100%;

  font: inherit;

  padding: 0.5rem 1.5rem;

  border: 1px solid #8b005d;

  color: white;

  background: #8b005d;

  box-shadow: 0 0 4px rgba(0, 0, 0, 0.26);

  cursor: pointer;

  @media (min-width:768px) {

    width: auto;

  }

  &:focus {

    outline: none;

  }

  &:hover,

  &:active {

    background: #ac0e77;

    border-color: #ac0e77;

    box-shadow: 0 0 8px rgba(0, 0, 0, 0.26);

  }

`