**Простое определение стилей**

const Button = styled.button`

background: #3a86ff;

color: white;

font-size: 16px;

padding: 10px 24px;

border-radius: 8px;

border: none;

cursor: pointer;

`;

}

**Динамические стили с помощью пропсов**

const Button = styled.button<{ primary?: boolean; }>`

background: ${props => props.primary ? "#BF4F74" : "white"};

color: ${props => props.primary ? "white" : "#BF4F74"};

`;

render(

<div>

<Button>Normal</Button>

<Button primary>Primary</Button>

</div>

);

**Наследование стилей**

const Button = styled.button`

padding: 10px;

border-radius: 6px;

`;

const DangerButton = styled(Button)`

background: #e63946;

color: white;

`;

<DangerButton>Удалить</DangerButton>

**Псевдоэлементы, вложенности**

const Button = styled.button`

background-color: blue;

color: white;

&:hover {

background-color: darkblue;

}

&::before {

content: '';

position: absolute;

}

> p {

margin: 10px;

background: coral;

}

`;

const List = styled.ul`

list-style-type: none;

li {

padding: 10px;

background: lightgray;

&:nth-child(odd) {

background: darkgray;

}

}

`; **ИЛИ**

const List = styled.ul`

list-style-type: none;

${Item} {

padding: 10px;

background: lightgray;

&:nth-child(odd) {

background: darkgray;

}

}

`;

const InputContainer = styled.div`

position: relative;

input {

padding: 10px;

border: 1px solid #ccc;

&:focus {

border-color: blue;

+ label {

color: blue;

}

}

**Использование props для передачи значений**

const Title = styled.h1`

font-size: ${props => props.size || '32px'};

color: ${props => props.color || '#333'};

`;

<Title size="40px" color="#1976d2">Заголовок</Title>

**Дефолтное значение**

const Title = styled.h1<TitleProps>`

// Для лучшей читаемости - деструктурируем props,

// задаем дефолтное значение если это необходимо

font-weight: ${({ weight = 400 }) => weight};

`

**Создание глобальных стилей**

import { createGlobalStyle } from 'styled-components';

const GlobalStyle = createGlobalStyle`

\*,

\*::before,

\*::after {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

min-height: 100vh;

background: radial-gradient(yellow, orange);

display: grid;

place-content: center;

text-align: center;

}

`;

function App() {

return (

<>

<GlobalStyle />

<YourComponents />

</>

);

}

**Анимации с помощью styled-components**

import styled, { keyframes } from 'styled-components';

const rotate = keyframes`

from { transform: rotate(0deg);}

to { transform: rotate(360deg);}

`;

const Loader = styled.div`

display: inline-block;

width: 50px;

height: 50px;

border: 6px solid #eee;

border-top: 6px solid #1976d2;

border-radius: 50%;

animation: ${rotate} 1s linear infinite;

`;

**Использование темы через ThemeProvider**

import { ThemeProvider } from 'styled-components';

const theme = {

colors: {

primary: '#7986cb',

secondary: '#2b2b2b',

success: '#4caf50',

danger: '#f44336 ',

bg: '#E5E4E8',

font: '#19191B',

},

sizes: {

container: { width: 1200 },

},

media: {

extraLarge: '(max-width: 1140px)',

large: '(max-width: 960px)',

medium: '(max-width: 720px)',

small: '(max-width: 540px)',

},

…..

};

Для контейнера создаем компонент и там применяем значение ширины контейнера

const App = () => {

return (

<ThemeProvider theme={theme}>

<Routing />

<GlobalStyles />

</ThemeProvider>

)

}

**Переиспользуемые и условные стили через css helper**

import styled, { css } from 'styled-components';

const disabledStyle = css`

background: #b0b0b0;

color: #eee;

cursor: not-allowed;

`;

const Button = styled.button`

background: #1976d2;

color: white;

${props => props.disabled && disabledStyle};

`;

<Button disabled>Недоступно</Button>

const styles1 = css`

display: block;

`;

const styles2 = css`

display: block;

`;

const Div = styled.div`

${({ isAlt }) => {

isAlt ? styles1 : styles2;

}}

`;

**Работа с атрибутами элемента**

interface TextInputProps {

size: number;

}

const TextInput = styled.input.attrs<TextInputProps>((props) => ({

type: "text",

onFocus: () => console.log("Focused"),

size: (props.size || 4) + "px",

}))<TextInputProps>`

padding: ${({ size }) => size};

margin: ${({ size }) => size};

`;**Перезапись атрибутов компонента¶**

const PasswordInput = styled(Input).attrs({

type: 'password',

placeholder: 'Secret key...',

})``;

**Лучше не использовать без необходимости статичные свойства (атрибуты) внутри стилизованного компонента, так как зачастую эти свойства задаются в JSX. Просто знайте об этой фиче.**

атрибут "as" компонента "Button". Он позволяет изменять HTML-тег, сохраняя стилизацию (Изменим кнопку на ссылку, сохранив стили).

<Button as=”a”>Теперь ссылка</ Button>