

12. Высокоуровневый CSSTransition

Содержание урока

- Обзор;
- Знакомство с адаптерами;
- Анимирование компонента с помощью высокоуровневого `CSSTransition`;
- Анимирование изначального появления компонента на странице;
- Отключение анимации;
- Подведём итоги.

Обзор

Привет! 🙌👨🎓 В этой части конспекта мы разберём подход анимирования компонентов с помощью библиотеки [react-transition-group](#). 🐧

К делу! 🦊

Знакомство с адаптерами

Сам по себе React не предоставляет инструментов для анимирования компонентов. Для этого нам понадобится некий `адаптер`.

🔍 Хозяйке на заметку:


Концепция `адаптера` весьма распространённая и часто применяется не только для анимирования компонентов.

В качестве `адаптеров` (иногда их также называют `аддон` (англ. "add-on")) мы будем использовать компоненты из библиотеки [react-transition-group](#), открывающие интерфейс в виде «крюков» (англ. «hooks»), для триггеринга анимации компонента в определённые точки во времени. В каком-то смысле такой интерфейс можно сравнить с методами "жизненного цикла" компонента, хотя это не одно и то же.

[react-transition-group](#) предоставляет несколько компонентов-адаптеров:

- `CSSTransition` — это компонент с высокоуровневым API, основанный на компоненте-адаптере `Transition`. Он использует CSS-переходы для анимации;
- `Transition` — это самостоятельный компонент-основа для `CSSTransition`. Его мы рассмотрим в следующей части этого конспекта;
- `TransitionGroup` — компонент-менеджер. Он управляет группами компонентов

`CSSTransition` или `Transition`, чаще всего используется при анимировании списков.

 Хозяйке на заметку:

Раньше компоненты-адаптеры, описанные выше, были частью библиотеки React. Однако с усилением фокуса на «разделение ответственности» вышеописанные компоненты вынесли из React. И теперь они доступны в качестве независимого npm-пакета [react-transition-group](#).

Анимирование компонента с помощью высокоуровневого `CSSTransition`

С учетом того, что в основе `CSSTransition` лежит компонент `Transition`, давай рассмотрим основные состояния, поддерживаемые компонентом `Transition`:

- `entering`;
- `entered`;
- `exiting`;
- `exited`.

`CSSTransition` обладает контролирующим булевым пропсом `in`. Получение пропсом `in` значения `true` сигнализирует о переходе дочернего компонента из состояния `entering` в состояние `entered` на протяжении времени, определённого пропсом `timeout`.

Механика работы `CSSTransition` заключается в подстановке CSS-классов на протяжении переходов анимируемого компонента между вышеописанными состояниями.

Сперва применяется базовый CSS-класс, а затем следующий CSS-класс с префиксом «active», который и активирует CSS-анимацию.

 Пример кода 12.1:

```
1  .shoppingList {
2    display: flex;
3    flex-direction: column;
4    align-items: center;
5    padding: 15px;
6    list-style: none;
7  }
8
9  .shoppingListItem {
10   margin: 10px 0;
11 }
12
13 .shoppingListItem > span {
14   position: relative;
```

```

15 }
16
17 .shoppingListItem > span code {
18   position: absolute;
19   right: -15px;
20   top: 2px;
21   color: firebrick;
22   cursor: pointer;
23 }
24
25 .example-enter {
26   opacity: 0.01;
27 }
28
29 .example-enter.example-enter-active {
30   opacity: 1;
31   transition: opacity 500ms ease-in;
32 }
33
34 .example-exit {
35   opacity: 1;
36 }
37
38 .example-exit.example-exit-active {
39   opacity: 0.01;
40   transition: opacity 1000ms ease-in;
41 }

```

```

1  import React, { Component } from 'react';
2  import { CSSTransition, TransitionGroup } from 'react-transition-group';
3  import './styles.css';
4
5  export default class ShoppingList extends Component {
6    state = {
7      itemList:    [ 'Milk', 'Potato', 'Butter', 'Bananas' ],
8      newItemToBuy: '',
9    };
10
11    _addItem = () => {
12      const { itemList, newItemToBuy } = this.state;
13
14      if (newItemToBuy.length) {
15        const newItems = itemList.concat([ newItemToBuy ]);
16
17        this.setState({
18          itemList:    newItems,
19          newItemToBuy: '',
20        });

```

```

21     }
22   };
23
24   _removeItem = index => {
25     const newItems = this.state.itemsList.slice();
26
27     newItems.splice(index, 1);
28
29     this.setState(({ itemsList }) => ({
30       itemsList: itemsList.filter(
31         (item, itemIndex) => itemIndex !== index,
32       ),
33     }));
34   };
35
36   _updateNewItemToAdd = event => {
37     this.setState({
38       newItemToBuy: event.target.value,
39     });
40   };
41
42   render() {
43     const { newItemToBuy, itemsList } = this.state;
44
45     const items = itemsList.map((item, index) => {
46       return (
47         <CSSTransition
48           classNames = 'example'
49           key = { item }
50           timeout = { { enter: 500, exit: 1000 } }
51         >
52         <li className = 'shoppingListItem'>
53           { item }
54           <span onClick = { () => this._removeItem(index) }>
55             <code>x</code>
56           </span>
57         </li>
58       </CSSTransition>
59     );
60   });
61
62   return (
63     <ul className = 'shoppingList'>
64       <li>
65         <input
66           placeholder = 'Item to buy'
67           type = 'text'
68           value = { newItemToBuy }

```

```

69         onChange = { this._updateNewItemToAdd }
70     }
71     <button onClick = { this._addItem }>Add
Item</button>
72     </li>
73     <TransitionGroup>{ items }</TransitionGroup>
74 </ul>
75     );
76 }
77 }
78

```

В примере кода 12.1 описывается анимирование элементов списка покупок. На строке кода 45 происходит перебор элементов из состояния компонента и оборачивания их в компонент-адаптер `CSSTransition`. На строке кода 48 объявлен пропс `classNames`, описывающий базовое имя CSS-классов, которые применяются в момент соответствующей фазы анимирования компонента. Соответствующие CSS-классы объявлены в файле с CSS. Список элементов, каждый из которых обернут компонентом `CSSTransition`, предоставлен компоненту `TransitionGroup` в виде `children`.

Механику применения CSS-классов можно описать так:

- `example-enter`: применяется перед переходом анимируемого компонента в состояние `enter`;
- `example-enter.example-enter-active`: описывает стиль по достижению состояния `enter`;
- `example-exit`: применяется перед переходом анимируемого компонента в состояние `exit`;
- `example-exit.example-exit-active`: описывает стиль по достижению состояния `exit`.

Компонент-адаптер `CSSTransition` имеет в распоряжении еще несколько интересных возможностей. Более детально с ними ты можешь ознакомиться, обратившись к официальной документации на [github](#). 🦄

Анимирование изначального появления компонента на странице

С помощью компонента `CSSTransition` можно также анимировать изначальное появление анимируемого компонента на странице (когда у компонента вызывается метод "жизненного цикла" `componentDidMount`).

 Пример кода 12.2:

```

1  render () {
2
3      return (
4          <CSSTransition
5              classNames = 'example'
6              appear = { true }
7              timeout = { 500 }>
8              <span>This element is animated on initial mount</span>
9          </CSSTransition>
10     );
11 }

```

```

1  .example-appear {
2      opacity: 0.01;
3  }
4
5  .example-appear.example-appear-active {
6      opacity: 1;
7      transition: opacity .5s ease-in;
8  }

```

В примере кода 12.2 на строке кода 5 в файле с JavaScript для компонента `CSSTransition` объявлен пропс `appear`. Это даст инструкцию `CSSTransition` включить в цикл анимирования еще одну фазу изначального появления компонента и анимировать компонент стилями, взятыми из соответствующих CSS-классов.

Отключение анимации

В случаях, когда необходимо предотвратить анимирование компонента, можно добавить булевой пропс с именем фазы анимирования, которую нужно отключить, и передать этому пропсу `false` в виде значения. Например, `enter = { false }` или `exit = { false }` соответственно. Если передать `false` обоим пропсам, анимация отключится полностью. 🧑

Подведём итоги

В этом уроке мы рассмотрели компонент-адаптер для анимирования React-компонентов `CSSTransition`, а также компонент стейт-машину `TransitionGroup`, помогающий анимировать списки из `CSSTransition` (или `Transition`, который мы рассмотрим в следующей части конспекта). Механика работы `CSSTransition` заключается в применении CSS-классов, описывающих желаемую анимацию в определенные точки во времени. Вследствие чего компонент плавно переходит из одного состояния в другое.

Спасибо, что остаёшься с нами! 🙌 В следующей части конспекта мы рассмотрим компонент-адаптер `Transition`, на котором основан `CSSTransition`. До встречи! 🐓

Мы будем очень признательны, если ты оставишь свой фидбек в отношении этой части конспекта на нашу электропочту hello@lectrum.io.