13. Низкоуровневый Transition

Содержание урока

- Обзор;
- Анимирование компонента с помощью низкоуровневого `Transition`;
- Подведём итоги.

Обзор

Привет! 👋🏻 В этой части конспекта мы продолжим разбирать подходы аминирования компонентов посредством использования компонента-адаптера `Transition` из библиотеки react-transition-group. 🐳

К делу! 👊🏻

Анимирование компонента с помощью низкоуровневого `Transition`

Компонент-адаптер `Transition` обладает простым декларативным API. Он позволяет описать переход компонента из одного состояния в другое в течение определенного периода времени. Чаще всего используется для анимирования появления компонента на странице, однако также может быть использован для анимации «здесь и сейчас». 🦖

Основное отличие компонента `Transition` от `CSSTransition` в том, что `Transition` предоставляет интерфейс, оптимизированный для анимирования компонента с помощью JavaScript вместо CSS.

По умолчанию `Transition` не меняет поведение анимируемого компонента. Он только «следит» за переходом компонента из одного состояния в другое.

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

🖥️ Пример кода 13.1:

```
1  state = { in: false };
```

```
2
3  toggleEnterState = () => {
4    this.setState(({ in }) => ({ in: !in }));
5  }
6
7  render() {
8    return (
9      <div>
10       <Transition in = { this.state.in } timeout = { 500 } />
11       <button onClick = { this.toggleEnterState }>Click to
   Enter</button>
12     </div>
13   );
14 }
```

В `примере кода 13.1` по клику на элемент `<button>` компонент `Transition` перейдет в состояние `entering` и пробудет там `500` миллисекунд (значение пропса `timeout`), после чего перейдет в состояние `entered`. При повторном клике на элемент `<button>` происходит обратное: компонент `Transition` перейдет из состояния `exiting` в состояние `exited` соответственно.

Давай используем полученные знания и создадим анимированный компонент-мячик для Флаффи, кота Оскара. 🐈

🖥️ Пример кода 13.2:

```
1  import React from 'react';
2  import { Transition } from 'react-transition-group';
3  import TweenMax from 'gsap';
4
5  export default class Room extends Component {
6      state = {
7          isFetching: true,
8      };
9
10     _throwBall = () => {
11         this.setState(({ isFetching }) => ({
12             isFetching: !isFetching,
13         }));
14     };
15
16     _handleEnter = elelemnt => {
17         TweenMax.fromTo(elelemnt, 0.3, { y: 100, opacity: 0 }, { y: 0,
   opacity: 1 });
18     };
19
20     _handleExit = elelemnt => {
21         TweenMax.fromTo(elelemnt, 0.3, { y: 0, opacity: 1 }, { y: -100,
   opacity: 0 });
```

```
22      };
23
24      render() {
25          return (
26              <>
27                  <Transition
28                      in = { this.state.isFetching }
29                      timeout = { 500 }
30                      onEnter = { this._handleEnter }
31                      onExit = { this._handleExit }
32                  >
33                      <div
34                          style = { {
35                              width:           100,
36                              height:          100,
37                              backgroundColor: 'red',
38                              position:        'absolute',
39                              top:             '50%',
40                              right:           '50%',
41                              borderRadius:    '50%',
42                          } }
43                      />
44                  </Transition>
45                  <button onClick = { this._throwBall }>Fetch!</button>
46              </>
47          );
48      }
49  }
50
```

В `примере кода 13.2` на `строке кода 33` описана анимация элемента `<div>`, играющего роль мячика Флаффи. 🏀

Данный `<div>` является дочерним компонентом по отношению к компоненту-адаптеру `Transition`. Мы знаем, что пропс `in`, объявленный на `строке кода 28`, управляет состоянием компонента `Transition` и его дочернего компонента. Кнопка `<button>` на `строке кода 45` переключает пропс `in` для компонента `Transition` между значениями `true` и `false`. Пропс `onEnter`, объявленный на `строке кода 30`, принимает значение коллбека, срабатывающего перед переходом компонета `Transition` в состояние `entering`. Пропс `onExit` работает аналогично, только сработает он перед переходом в состояние `exiting`. Обработчики для обоих состояний получают ссылку на анимируемый DOM-элемент первым параметром, чтобы осуществить императивное анимирование, описанное библиотекой `gsap`.

Таким образом, при повторяющихся нажатиях на кнопку `Fetch!` дочерний компонент-мячик будет анимированно переходить из одного состояния в другое.

Компонент-адаптер `Transition` имеет в распоряжении еще несколько интересных возможностей. Более детально с ними ты можешь ознакомиться, обратившись к официальной документации на github. 🦄

Подведём итоги

В данном уроке мы рассмотрели принцип работы компонента-адаптера `Transition`.

Данный компонент предоставляет некоторые `«хуки»` для императивной обработки смены состояний анимируемого компонента с помощью JavaScript.

Спасибо, что остаёшься с нами! 👋🏼 В следующей части конспекта мы рассмотрим основы тестирования кода. До встречи!

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