Создание модально окна

№ урока: 10 **Курс:** React Essential

Средства обучения: Браузер Chrome, редактор кода VS Code или любой другой.

Обзор, цель и назначение урока

Узнать, что такое Рефы, познакомиться с Context API и Рендер пропсами. Создать модальное окно с использованием контекста и рендер пропс.

Изучив материал данного занятия, учащийся сможет

- Использовать Рендер пропсы.
- Использовать Рефы.
- Использовать Context API.
- Создавать компоненты, используя Рендер пропсы, Рефы, Context API.

Содержание урока

1. Рефы и DOM
2. Context API
3. Рендер-пропсы

Резюме

Рефы — дают возможность получить доступ к DOM-узлам или React-элементам, созданным в рендер-методе.

Когда использовать рефы:

- Управление фокусом, выделение текста или воспроизведение медиа;
- Императивный вызов анимаций;
- Интеграция со сторонними DOM-библиотеками.

Избегайте использования рефов в ситуациях, когда задачу можно решить декларативным способом.

Создание рефов - Рефы создаются с помощью React.createRef() и прикрепляются к React-элементам через ref атрибут. Обычно рефы присваиваются свойству экземпляра класса в конструкторе, чтобы на них можно было ссылаться из любой части компонента.

Доступ к рефам - Когда реф передаётся элементу в методе render, ссылка на данный узел доступна через свойство рефа current (ref.current).

Значение рефа отличается в зависимости от типа узла:

- Когда атрибут ref используется с HTML-элементом, свойство current созданного рефа в конструкторе с помощью React.createRef() получает соответствующий DOM-элемент.
- Когда атрибут ref используется с классовым компонентом, свойство current объекта-рефа получает экземпляр смонтированного компонента.

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Нельзя использовать ref атрибут с функциональными компонентами, потому что для них не создаётся экземпляров.

Если вам нужен реф на функциональный компонент, можете воспользоваться forwardRef (возможно вместе с useImperativeHandle), либо превратить его в классовый компонент.

Рефы и функциональные компоненты

По умолчанию **нельзя использовать атрибут ref с функциональными компонентами**, потому что для них не создаётся экземпляров. Тем не менее, можно **использовать атрибут ref внутри функционального компонента** при условии, что он ссылается на DOM-элемент или классовый компонент с использованием хука useRef().

Колбэк-рефы - Вместо того, чтобы передавать объект ref, созданный с помощью createRef(), вы можете передать функцию. Данная функция получит экземпляр React-компонента или HTML DOM-элемент в качестве аргумента, которые потом могут быть сохранены или доступны в любом другом месте. React вызовет ref колбэк с DOM-элементом при монтировании компонента, а также вызовет его со значением null при размонтировании. Рефы будут хранить актуальное значение перед вызовом методов componentDidMount или componentDidUpdate. Вы можете передавать колбэк-рефы между компонентами точно так же, как и объектные рефы, созданные через React.createRef().

Context API - позволяет передавать данные через дерево компонентов без необходимости передавать пропсы на промежуточных уровнях.

Когда использовать контекст - контекст разработан для передачи данных, которые можно назвать «глобальными» для всего дерева React-компонентов (например, текущий аутентифицированный пользователь, UI-тема или выбранный язык). По возможности не используйте его, так как это усложняет повторное использование компонентов. Если вы хотите избавиться от передачи некоторых пропсов на множество уровней вниз, обычно композиция компонентов является более простым решением, чем контекст.

React.createContext - создание объекта Context. Когда React рендерит компонент, который подписан на этот объект, React получит текущее значение контекста из ближайшего подходящего Provider выше в дереве компонентов. вниз, обычно композиция компонентов является более простым решением, чем контекст.

Аргумент defaultValue используется **только** в том случае, если для компонента нет подходящего Provider выше в дереве. Это может быть полезно для тестирования компонентов в изоляции, без необходимости оборачивать их. Обратите внимание: если передать undefined как значение Provider, компоненты, использующие этот контекст, не будут использовать defaultValue.

Context.Provider - каждый объект Контекста используется вместе с Provider компонентом, который позволяет дочерним компонентам, использующим этот контекст, подписаться на его изменения.
Принимает проп value, который будет передан во все компоненты, использующие этот контекст и являющиеся потомками этого Provider компонента.

Один Provider может быть связан с несколькими компонентами, потребляющими контекст. Так же Provider компоненты могут быть вложены друг в друга, переопределяя значение

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

контекста глубже в дереве. Все потребители, которые являются потомками Provider, будут повторно рендериться, как только проп value у Provider изменится. Потребитель (включая .contextType и useContext) перерендерится при изменении контекста, даже если его родитель, не использующий данный контекст, блокирует повторные рендеры с помощью shouldComponentUpdate.

Изменения определяются с помощью сравнения нового и старого значения, используя алгоритм, аналогичный Object.is.

Class.contextType - в свойство класса contextType может быть назначен объект контекста, созданный с помощью React.createContext(). Это позволяет вам использовать ближайшее и актуальное значение указанного контекста при помощи this.context. В этом случае вы получаете доступ к контексту, как во всех методах жизненного цикла, так и в рендер методе. Вы можете подписаться только на один контекст, используя этот API. В случае, если вам необходимо использовать больше одного, смотрите Использование нескольких контекстов. Если вы используете экспериментальный синтаксис публичных полей класса, вы можете использовать static поле класса, чтобы инициализировать ваш contextType.

Context.Consumer — это React-компонент, который подписывается на изменения контекста. В свою очередь, это позволяет вам подписаться на контекст в функциональном компоненте. Consumer принимает функцию в качестве дочернего компонента. Эта функция принимает текущее значение контекста и возвращает React-компонент. Передаваемый аргумент value будет равен ближайшему (вверх по дереву) значению этого контекста, а именно пропу value Provider компонента. Если такого Provider компонента не существует, аргумент value будет равен значению defaultValue, которое было передано в createContext().

Context.displayName - Объекту Context можно задать строковое свойство displayName. React DevTools использует это свойство при отображении контекста.

Рендер-пропсы - относится к возможности компонентов React разделять код между собой с помощью пропа, значение которого является функцией. Компонент с рендер-пропом берёт функцию, которая возвращает React-элемент, и вызывает её вместо реализации собственного рендера. Такой подход, в частности, применяется в библиотеках React Router, Downshift и Formik. Иными словами, рендер-проп — это функция, которая сообщает компоненту что необходимо рендерить.

Важно запомнить, что из названия паттерна «рендер-проп» вовсе не следует, что для его использования *вы должны обязательно называть проп render*. На самом деле, *любой* проп, который используется компонентом и является функцией рендеринга, технически является и «рендер-пропом». Несмотря на то, что мы используем render, мы можем также легко использовать проп children. Проп children не обязательно именовать в списке «атрибутов» вашего JSX-элемента. Вместо этого, вы можете поместить его прямо *внутрь* элемента!

Использование рендер-пропа может свести на нет преимущество, которое даёт React.PureComponent, если вы создаёте функцию внутри метода render. Это связано с тем, что поверхностное сравнение пропсов всегда будет возвращать false для новых пропсов и каждый render будет генерировать новое значение для рендер-пропа.

Закрепление материала

- Когда можно использовать рендер просы?

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Что такое колбэк-рефы?
- Зачем нужен React.Fragment?

Дополнительное задание

Задание
Прочитать документацию React, а именно - раздел «Рендер-пропсы» и «Context API». Разобраться как работают ref ссылки.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.

Задание 2
Создайте контекст с использованием Context API, в котором будет храниться название текущей темы. Тему можно переключать с помощью кнопки. При переключении темы меняется цвет фона тега body.

Рекомендуемые ресурсы

Рефы и DOM
Context API
Рендер-пропсы

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com