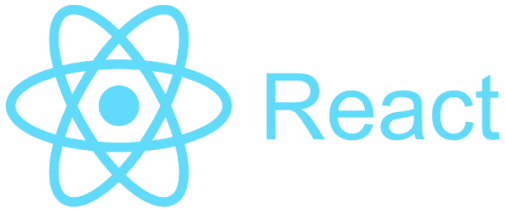


React Essential

Создание страницы ошибки

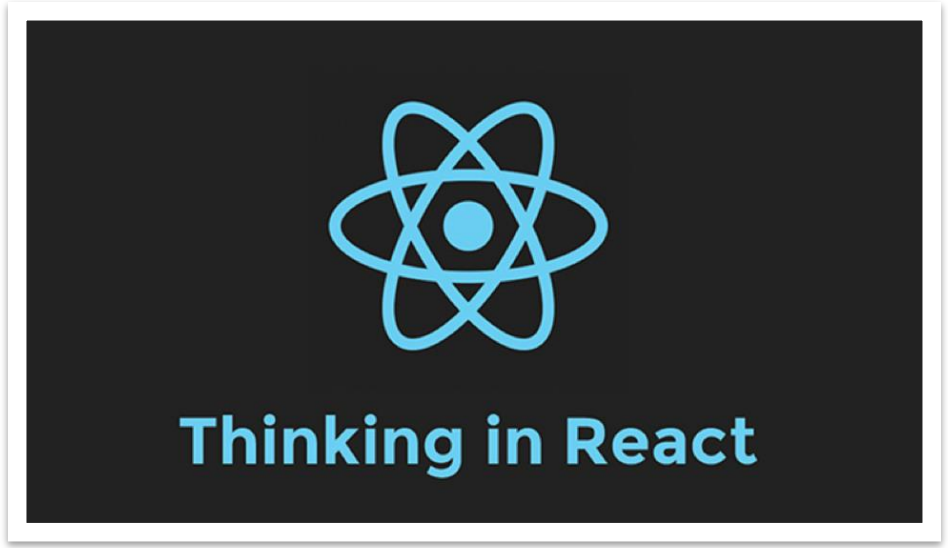
Introduction

Муляк Дмитрий

Front-end developer at GlobalTechMakers

dmitriymuliak

dmitriymuliak



Тема

Создание страницы ошибки

Создание страницы ошибки

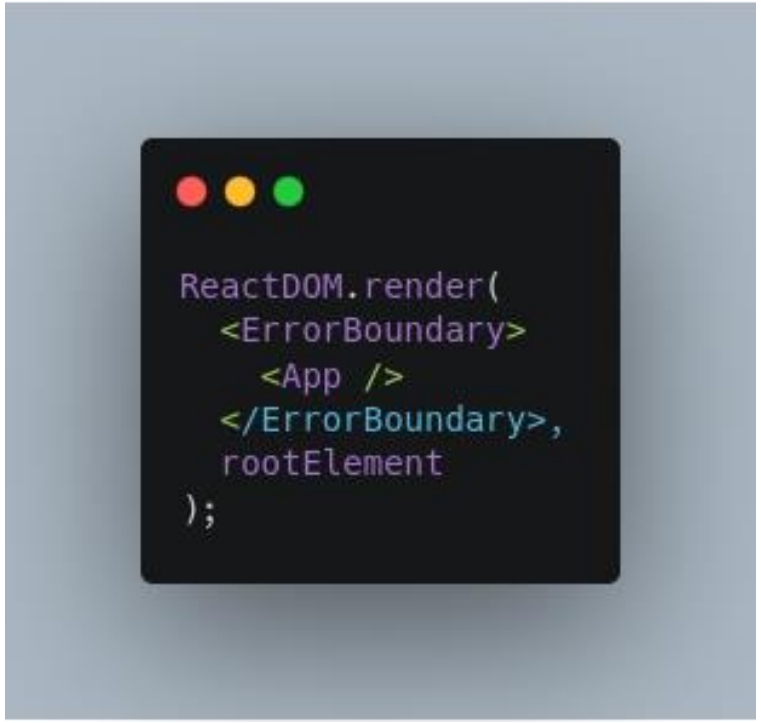
1. Предохранители
2. React.Fragment
3. Строгий режим

Предохранители

Ошибка JavaScript где-то в коде UI не должна прерывать работу всего приложения. Чтобы исправить эту проблему для React-пользователей, React 16 вводит концепцию «предохранителя» (error boundary).

Предохранители — это компоненты React, которые **отлавливают ошибки JavaScript в любом месте деревьев их дочерних компонентов, сохраняют их в журнале ошибок и выводят запасной UI**, вместо рухнувшего дерева компонентов.

Предохранители отлавливают ошибки при рендеринге, в методах жизненного цикла и конструкторах деревьев компонентов, расположенных под ними.



Предохранители

ErrorBoundary Component

Классовый компонент является предохранителем, если он включает хотя бы один из следующих методов жизненного цикла:
static getDerivedStateFromError() или componentDidCatch().

Предохранители работают как JavaScript-блоки catch {}, но только для компонентов. Только классовые компоненты могут выступать в роли предохранителей.

И можно дальше пользоваться, как обыкновенным компонентом:

```
<ErrorBoundary>
  <MyWidget />
</ErrorBoundary>
```

```
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }

  static getDerivedStateFromError(error) {
    // Обновить состояние с тем, чтобы следующий рендер показал запасной UI.
    return { hasError: true };
  }

  componentDidCatch(error, errorInfo) {
    // Можно также сохранить информацию об ошибке в соответствующую службу журнала ошибок
    logErrorToMyService(error, errorInfo);
  }

  render() {
    if (this.state.hasError) {
      // Можно отрендерить запасной UI произвольного вида
      return <h1>Что-то пошло не так.</h1>;
    }

    return this.props.children;
  }
}
```

React.Fragment

Возврат нескольких элементов из компонента является распространённой практикой в React. Фрагменты позволяют формировать список дочерних элементов, не создавая лишних узлов в DOM.

```
render() {
  return (
    <React.Fragment>
      <ChildA />
      <ChildB />
      <ChildC />
    </React.Fragment>
  );
}
```

Фрагменты, объявленные с помощью <React.Fragment>, могут иметь ключи. Например, их можно использовать при создании списка определений, преобразовав коллекцию в массив фрагментов.
key — это единственный атрибут, допустимый у Fragment.

```
class Columns extends React.Component {
  render() {
    return (
      <>
        <td>Привет</td>
        <td>Мир</td>
      </>
    );
  }
}
```

Сокращённая запись

Существует сокращённая запись объявления фрагментов. Она выглядит как пустые теги:

Строгий режим

StrictMode — инструмент для обнаружения потенциальных проблем в приложении. Также как и Fragment, StrictMode не рендерит видимого UI. Строгий режим активирует дополнительные проверки и предупреждения для своих потомков.

На данный момент StrictMode помогает в:

- Обнаружении небезопасных методов жизненного цикла
- Предупреждении об использовании устаревшего API строковых реф
- Предупреждении об использовании устаревшего метода findDOMNode
- Обнаружении неожиданных побочных эффектов
- Обнаружении устаревшего API контекста

```
import React from 'react';

function ExampleApplication() {
  return (
    <div>
      <Header />
      <React.StrictMode>
        <div>
          <ComponentOne />
          <ComponentTwo />
        </div>
      </React.StrictMode>
      <Footer />
    </div>
  );
}
```

Проверка знаний

TestProvider.com

Проверьте как Вы усвоили данный материал на TestProvider.com

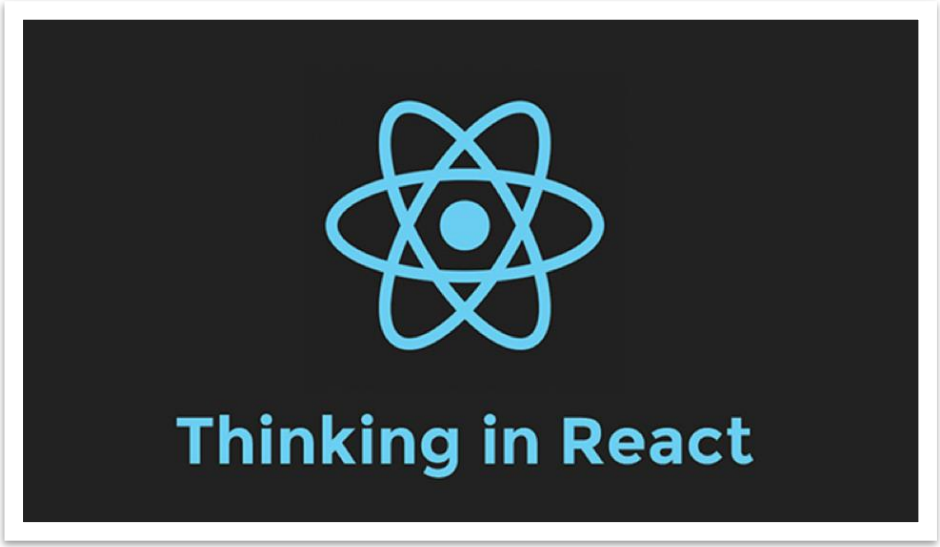
TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Спасибо за внимание! До новых встреч!

Муляк Дмитрий

Front-end Developer at GTM



Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association