第3章　React 16新特性

React 16是Facebook在2017年9月发布的React最新版本。React 16基于代号为“Fiber”的新架构实现，几乎对React的底层代码进行了重写，但对外的API基本不变，所以开发者可以几乎无缝地迁移到React 16。此外，基于新的架构，React 16实现了许多新特性。

3.1 render新的返回类型

React 16之前，render方法必须返回单个元素。现在，render方法支持两种新的返回类型：数组（由React元素组成）和字符串。定义一个ListComponent组件，它的render方法返回数组：

```
class ListComponent extends Component {
  render() {
    return [
      <li key="A">First item</li>,
      <li key="B">Second item</li>,
      <li key="C">Third item</li>
    ];
  }
}
```

再定义一个StringComponent组件，它的render方法返回字符串：

```
class StringComponent extends Component {
  render() {
    return "Just a strings";
  }
}
```

App组件的render方法渲染ListComponent和StringComponent：

```
export default class App extends Component {
```

```
  render() {
    return [
      <ul>
        <ListComponent />
      </ul>,
      <StringComponent />
    ];
  }
}
```

本节项目源代码的目录为/chapter-03/react16-render。

3.2 错误处理

React 16之前，组件在运行期间如果执行出错，就会阻塞整个应用的渲染，这时候只能刷新页面才能恢复应用。React 16引入了新的错误处理机制，默认情况下，当组件中抛出错误时，这个组件会从组件树中卸载，从而避免整个应用的崩溃。这种方式比起之前的处理方式有所进步，但用户体验依然不够友好。React 16还提供了一种更加友好的错误处理方式——错误边界（Error Boundaries）。错误边界是能够捕获子组件的错误并对其做优雅处理的组件。优雅的处理可以是输出错误日志、显示出错提示等，显然这比直接卸载组件要更加友好。

定义了componentDidCatch(error, info)这个方法的组件将成为一个错误边界，现在我们创建一个组件ErrorBoundary：

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

  componentDidCatch(error, info) {
    // 显示错误UI
    this.setState({ hasError: true });
    // 同时输出错误日志
    console.log(error, info);
  }
```

```
  render() {
    if (this.state.hasError) {
      return <h1>Oops, something went wrong.</h1>;
    }
    return this.props.children;
  }
}
```

然后在App中使用ErrorBoundary：

```
class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      user: { name: "react" }
    };
  }
  // 将user置为null，模拟异常
  onClick = () => {
    this.setState({ user: null });
  };

  render() {
    return (
      <div>
        <ErrorBoundary>
```

```
          <Profile user={this.state.user} />
        </ErrorBoundary>
        <button onClick={this.onClick}>更新</button>
      </div>
    );
  }
}

const Profile = ({ user }) => <div>name: {user.name}</div>;
```

点击更新按钮后，Profile接收到的属性user为null，程序会抛出TypeError，这个错误会被ErrorBoundary捕获，并在界面上显示出错提示。注意，使用create-react-app创建的项目，当程序发生错误时，create-react-app 会在页面上创建一个浮层显示错误信息，要观察ErrorBoundary的正确效果，需要先关闭错误浮层。

本节项目源代码的目录为/chapter-03/react16-error-boundary。

3.3 Portals

React 16的Portals特性让我们可以把组件渲染到当前组件树以外的DOM节点上，这个特性典型的应用场景是渲染应用的全局弹框，使用Portals后，任意组件都可以将弹框组件渲染到根节点上，以方便弹框的显示。Portals的实现依赖ReactDOM的一个新的API：

```
ReactDOM.createPortal(child, container)
```

第一个参数child是可以被渲染的React节点，例如React元素、由React元素组成的数组、字符串等，container是一个DOM元素，child将被挂载到这个DOM节点。

我们创建一个Modal组件，Modal使用ReactDOM.createPortal()在DOM根节点上创建一个弹框：

```
class Modal extends Component {
  constructor(props) {
    super(props);
  // 根节点下创建一个div节点
    this.container = document.createElement("div");
    document.body.appendChild(this.container);
  }

  componentWillUnmount() {
    document.body.removeChild(this.container);
```

```
    }

    render() {
    // 创建的DOM树挂载到this.container指向的div节点下面
      return ReactDOM.createPortal(
        <div className="modal">
          <span className="close" onClick=
{this.props.onClose}>
            &times;
          </span>
          <div className="content">
            {this.props.children}
          </div>
        </div>,
        this.container
      );
    }
  }
```

在App中使用Modal：

```
  class App extends Component {
    constructor(props) {
      super(props);
      this.state = { showModal: true };
    }
    // 关闭弹框
```

```
    closeModal = () => {
      this.setState({ showModal: false });
    };

    render() {
      return (
        <div>
          <h2>Dashboard</h2>
          {this.state.showModal && (
            <Modal onClose={this.closeModal}>Modal
Dialog</Modal>
          )}
        </div>
      );
    }
  }

  export default App;
```

本节项目源代码的目录为/chapter-03/react16-portals。

3.4 自定义DOM属性

React 16之前会忽略不识别的HTML和SVG属性，现在React会把不识别的属性传递给DOM元素。例如，React 16之前，下面的React元素

```
<div custom-attribute="something" />
```

在浏览器中渲染出的DOM节点为：

```
<div />
```

而React 16渲染出的DOM节点为：

```
<div custom-attribute="something" />
```

本节项目源代码的目录为/chapter-03/react16-custom-dom。

3.5 本章小结

本章介绍了React 16的新特性，主要包括render方法新支持的返回类型、新的错误处理机制和Error Boundary组件、可以将组件挂载到任意DOM树的Portals 特性以及自定义DOM属性的支持。这些只是React 16常用的特性，除此之外，React 16还有其他的新特性，例如setState传入null时不会再触发组件更新、更加高效的服务器端渲染方式等。相信基于新的Fiber架构，React 16还会推出更多更强大的特性。