如下，我们一个个传递属性是不是很麻烦

class App extends React.Component {

    render() {

        return <Toolbar theme="dark" />;

    }

}

function Toolbar(props) {

    return (

        <div>

            <ThemedButton theme={props.theme} />

        </div>

    );

}

class ThemedButton extends React.Component {

    render() {

        return <Button theme={this.props.theme} />;

    }

}

Context允许我们组件将某个属性保存到context中，在其某个后代中，从context取出该属性

// 创建 context，light 为默认值

const ThemeContext = React.createContext('light');

class App extends React.Component {

    render() {

        // 应用 context，dark 为当前组件使用的值

        return (

            <ThemeContext.Provider value="dark">

                <Toolbar />

            </ThemeContext.Provider>

        );

    }

}

// 中间的组件再也不必指明往下传递 theme 了。

function Toolbar() {

    return (

        <div>

            <ThemedButton />

        </div>

    );

}

class ThemedButton extends React.Component {

    // contextType 指定查找最近的父组件的 context 而不是根

static contextType = ThemeContext;

    render() {

        // 使用 context

        return <Button theme={this.context} />;

    }

}