React.js 小书

<-- 返回首页

动手实现 React-redux（一）：初始化工程

- 作者：胡子大哈
- 原文链接： http://huziketang.com/books/react/lesson36

（本文未审核）

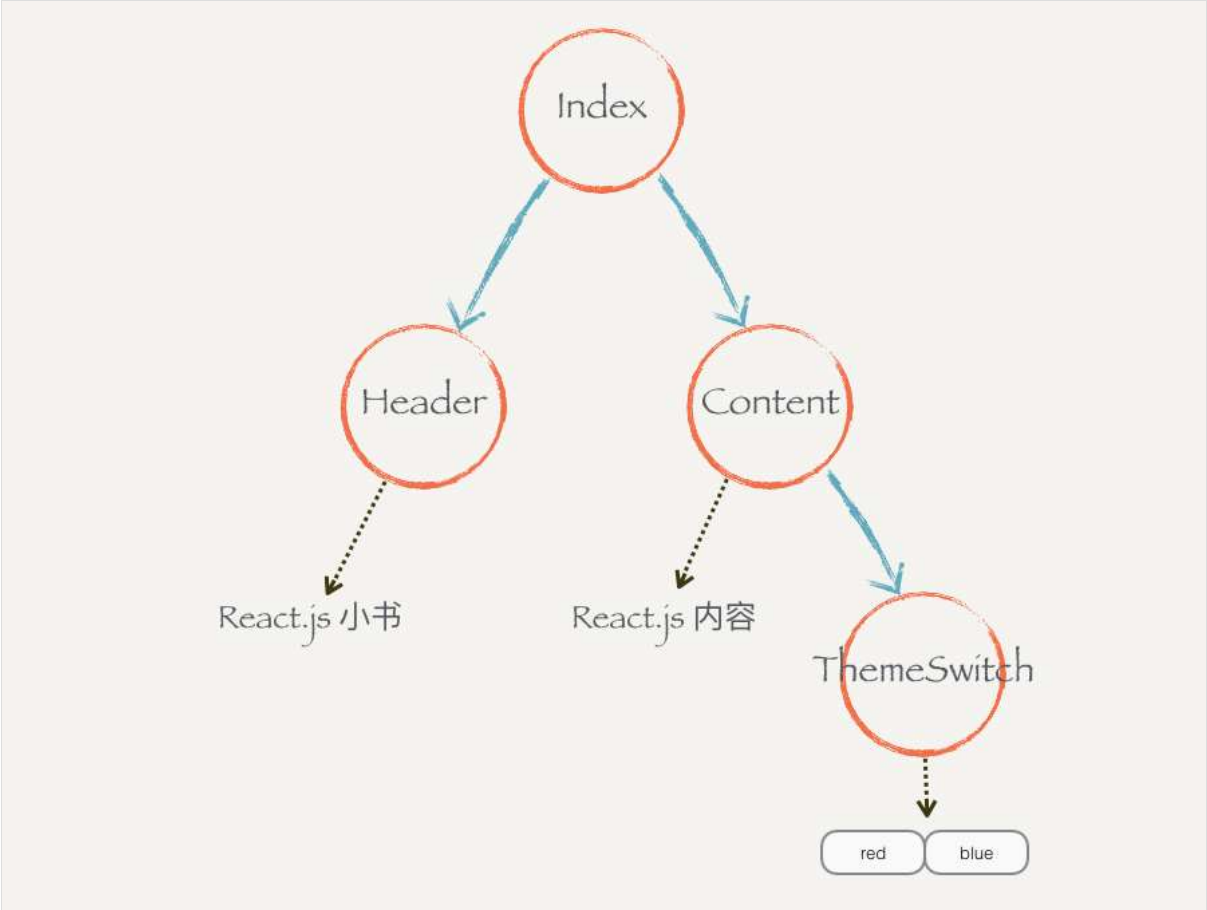
可以看到 Redux 并不复杂，它那些看起来匪夷所思的设定其实都是为了解决特定的问题而存在的，我们把问题想清楚以后就不难理解它的那些奇怪的设定了。这节开始我们来看看如何把 Redux 和 React.js 结合起来，你会发现其实它们也并不复杂。

回顾一下，我们在 前端应用状态管理 — 状态提升 中提过，前端中应用的状态存在的问题：一个状态可能被多个组件**依赖**或者**影响**，而 React.js 并没有提供好的解决方案，我们只能把状态提升到**依赖**或者**影响**这个状态的所有组件的公共父组件上，我们把这种行为叫做状态提升。但是需求不停变化，共享状态没完没了地提升也不是办法。

后来我们在 React.js 的 context 中提出，我们可用把共享状态放到父组件的 context 上，这个父组件下所有的组件都可以从 context 中直接获取到状态而不需要一层层地进行传递了。但是直接从 context 里面存放、获取数据增强了组件的耦合性；并且所有组件都可以修改 context 里面的状态就像谁都可以修改共享状态一样，导致程序运行的不可预料。

既然这样，为什么不把 context 和 store 结合起来？毕竟 store 的数据不是谁都能修改，而是约定只能通过 `dispatch` 来进行修改，这样的话每个组件既可以去 context 里面获取 store 从而获取状态，又不用担心它们乱改数据了。

听起来不错，我们动手试一下。我们还是拿“主题色”这个例子做讲解，假设我们现在需要做下面这样的组件树：



`Header` 和 `Content` 的组件的文本内容会随着主题色的变化而变化，而 `Content` 下的子组件 `ThemeSwitch` 有两个按钮，可以切换红色和蓝色两种主题，按钮的颜色也会随着主题色的变化而变化。

用 `create-react-app` 新建一个工程，然后在 `src/` 目录下新增三个文件：`Header.js`、`Content.js`、`ThemeSwitch.js`。

修改 `src/Header.js`：

```
import React, { Component, PropTypes } from 'react'

class Header extends Component {
  render () {
    return (
      <h1>React.js 小书</h1>
    )
  }
}

export default Header
```

修改 `src/ThemeSwitch.js`：

```
import React, { Component, PropTypes } from 'react'

class ThemeSwitch extends Component {
```

```
  render () {
    return (
      <div>
        <button>Red</button>
        <button>Blue</button>
      </div>
    )
  }
}

export default ThemeSwitch
```

修改 `src/Content.js`：

```
import React, { Component, PropTypes } from 'react'
import ThemeSwitch from './ThemeSwitch'

class Content extends Component {
  render () {
    return (
      <div>
        <p>React.js 小书内容</p>
        <ThemeSwitch />
      </div>
    )
  }
}

export default Content
```

修改 `src/index.js`：

```
import React, { Component, PropTypes } from 'react'
import ReactDOM from 'react-dom'
import Header from './Header'
import Content from './Content'
import './index.css'

class Index extends Component {
  render () {
    return (
      <div>
        <Header />
        <Content />
      </div>
    )
  }
}

ReactDOM.render(
  <Index />,
```

```
  document.getElementById('root')
)
```

这样我们就简单地把整个组件树搭建起来了，用 `npm start` 启动工程，然后可以看到页面上显示：

当然现在文本都没有颜色，而且点击按钮也不会有什么反应，我们还没有加入表示主题色的状态和相关的业务逻辑，下一节我们就把相关的逻辑加进去。

> 因为第三方评论工具有问题，对本章节有任何疑问的朋友可以移步到 React.js 小书的论坛 发帖，我会回答大家的疑问。

下一节：动手实现 React-redux（二）：结合 context 和 store

上一节：动手实现 Redux（六）：Redux 总结

1 条评论，1 人参与。

0

我有话说...

使用社交帐号登录

发布前先点击左边的按钮登录

最新评论

余恬穆灬 • 4月20日 16:11

no-useless-constructor 怎么破

顶 • 回复 • 分享»

友言?