React 16.3 的 Context API 能否完全取代 Redux?

关注问题 写回答

React Redux

关注者 164 被浏览 35,515

React 16.3 的 Context API 能否完全取代 Redux?

Redux 在什么情况下才能依然有用武之地? [图片] 或者说，什么情况下才是「actually need Redux」? 显示全部

关注问题 写回答 邀请回答 好问题 1 1 条评论 分享 ···

10 个回答 默认排序

拼音 + 关注

17 人赞同了该回答

这个问题包括两个部分：

- Redux和React 16.3中的新Context API分别解决了什么问题?
- 什么情况下更适合用Redux?

1 Redux和React 16.3中的新Context API分别解决了什么问题?

看了下，React 16.3的新Context API大概是这种感觉：

React对象多了一个新方法

```
React.createContext<T>(defaultValue: T)
```

这个方法接受一个值作为默认上下文，返回一个包括了两个组件类的对象：

```
{
    Provider: React.ComponentType<{value: T}>,
    Consumer: React.ComponentType<{children: (value: T)=> React.ReactNode}>
}
```

之后，创建Consumer组件时，Consumer组件的children函数会收到外部对应的Provider组件的props.value作为参数，例如：

```
type Theme = 'light' | 'dark';
// Pass a default theme to ensure type correctness
const ThemeContext: Context<Theme> = React.createContext('light');

class ThemeToggler extends React.Component {
  state = {theme: 'light'};
  render() {
    return (
      // Pass the current context value to the Provider's `value` prop.
      // Changes are detected using strict comparison (Object.is)
      <ThemeContext.Provider value={this.state.theme}>
        <button
          onClick={() =>
            this.setState(state => ({
```


▲ 赞同 17 ▼ 添加评论 分享 收藏 喜欢 ···

React 16.3 的 Context API 能否完全取代 Redux?

```
          }>
            Toggle theme
          </button>
          {this.props.children}
        </ThemeContext.Provider>
      );
    }
  }

  class Title extends React.Component {
    render() {
      return (
        // The Consumer uses a render prop API. Avoids conflicts in the
        // props namespace.
        <ThemeContext.Consumer>
          {theme => (
            <h1 style={{color: theme === 'light' ? '#000' : '#fff'}}>
              {this.props.children}
            </h1>
          )}
        </ThemeContext.Consumer>
      );
    }
  }
```

注意Consumer组件中的children函数收到了theme作为参数。

跟之前的Context API相比，现在只有每次调用 `createContext` 返回的Provider和Consumer之间可以互相传递信息，当然，和旧的Context API类似，Consumer仍然需要在Provider之内渲染才能收到由Provider提供的上下文，其他情况下只能收到上面提到的defaultValue作为上下文。

那么，新API解决了什么问题?

- 和组件props相比，新旧的Context API和Redux都解决了props存在的“只要是子组件需要的信息，即使父组件不需要，也必须先传给父组件然后一层层传到子组件”的问题
- 和Redux相比，新旧的Context API都解决了Redux存在的“一些信息的内容需要根据组件的包含关系决定，而Redux难以处理这类包含关系”的问题
- 和旧的Context API相比，新API解决了旧API无法处理“两个互相嵌套的组件提供的两个Context中，key相同的部分会冲突”的问题

但至少Redux解决的以下问题，在Context API中仍然没有得到解决:

- **逻辑/数据/视图分离的代码结构**（reducer/store/component），很好地划分了代码职责
- **在不同项目之间通用的存储和事件机制**，从而允许redux-devtools和time travel这种通用的开发工具、以及类似redux-observable这种强大中间件的存在（store/action）

2 什么情况下更适合用Redux?

至于什么时候actually need Redux，个人感觉没有固定的标准。

个人认同的是RxJS 5之后的维护者Ben Lesh的看法:

> Ask yourself: can I use everything in my store to render an app elsewhere? A mobile app? A desktop app? A custom canvas UI? Is there anything I couldn't use in those? Then it probably doesn't belong in my store.

相关问题

什么时候react.js合适使用redux与不使用? 17 个回答

react 框架中如何使用 Redux? 0 个回答

redux 如何关联 react 的方法? 0 个回答

如何优雅的在react中使用redux? 3 个回答

react 开发框架的数据如何通过 Redux 流动? 0 个回答

React 16.3 的 Context API 能否完全取代 Redux?

- 作为路由信息的补充，也就是**完全可以序列化到路由，但因为设计原因无法放进路由**的那些内容
- 存储跟领域模型相关的数据，例如**后端Model的缓存以及会话认证信息**等

其他例如窗口大小、临时Modal状态、视觉主题等等状态，都适合放在新Context或props中。

编辑于 2018-02-16 10:13

伊撒尔
做个好人

+ 关注

2 人赞同了该回答

特地来回答一下这个问题，因为昨天我写了一个基于最新的context api的状态管理的库，可以

```
npm i smox
```

来感受一下，顺便我也写了一个demo：

我可以很……明确的说，可以完完全全的抛弃redux了……

但是是否是context api取代的redux……那或许不完全是，但是这并不影响和redux说拜拜

这个api没有太难的地方，也不需要多么仔细的研究，但凡你写了试了，就知道它比redux缺了点啥，然后瞎搞搞就可以了

当然我也是不介意你用smox的，但是也许过几天就有很多基于新的context api的库诞生，而且绝对比我写得好……

但是这还不影响我们抛弃redux……

所以，redux是真的没需要了，不管你是等新的库，还是自己手动瞎搞搞，都不影响和redux说再见

目前我们网站已经全面移除mobx（是了……我们之前用的mobx），换上了smox，暂时还ok

另外，不要听别人的侃侃而谈，嫩不能取代写一下，心里还没有数吗?

编辑于 2018-04-02 01:35

▲ 赞同 2 ▼ 2 条评论 分享 收藏 喜欢 ··· 收起

MSDimos
程序员

需要操作更清晰，数据扁平化，就使用redux。需要一个状态管理的开发工具，就使用redux。

发布于 2018-02-14 21:10

▲ 赞同 ▼ 添加评论 分享 收藏 喜欢 ··· 收起

React 16.3 的 Context API 能否完全取代 Redux?

即将离开

1 人赞同了该回答

最近打算用一下，发现如下：1，context相当于新增一条下发路线，与props路线相互独立，互不干扰。2，context似乎不方便子组件修改数据？并且要求数据源在根组件，而不能在子组件？

与redux对比，感觉context适合很少修改，主要从根节点下发数据的情形，比如locale/主题。个人登录信息呢？这还有点迷惑。

redux啰嗦点，但万能。

----------补充----------

今天试用，发现context一样啰嗦，只是另一种啰嗦。多个文件都要导入创建的context来进行注入，页面都占了不少。可以参考官网给出的几个例子，感觉性价比不高的。

放弃了，万物归redux。

编辑于 2019-06-07 06:15

▲ 赞同 1 ▼ 添加评论 分享 收藏 喜欢 ··· 收起

元彦

莫愁前路无知己

6 人赞同了该回答

我们的结论是暂时不会，仔细研究过这个API的人我想可能会投反对票的，原因是利大于弊。

关于利，createContext 从能力上只比 组件context 多一个能力，就是当父组件实现了 shouldComponentUpdate return false时，子组件也可以更新 context value。

但多了这个功能的要付出代价却不小，新的createcontext 使用上必须先引用，比如在皮肤机制里对编码的约束会有些大，尤其在多人协作时，没有原先来的便利。

另外redux另一个关键概念是store，而react不太可能会去新加createStore。

发布于 2018-02-15 23:39

▲ 赞同 6 ▼ 添加评论 分享 收藏 喜欢 ··· 收起