组件设计模式（5）：组合组件

这一小节我们介绍“组合组件”（Compound Component）这种模式。网上介绍这种模式的文章也不少，但是都是一上来直接讲实现细节，却很少说应用于什么场景，所以往往都讲得让读者云里雾里。

这里我要强调，所谓模式，就是特定于一种问题场景的解决办法。

```
模式(Pattern) = 问题场景(Context) + 解决办法(Solution)
```

如果不搞清楚场景，单纯知道有这么一个办法，就好比拿到了一杆枪却不知道这杆枪用于打什么目标，是没有任何意义的。并不是所有的枪都是一样的，有的枪擅长狙击，有的枪适合近战，有的枪只是发个信号。

模式就是我们的武器，我们一定要搞清楚一件武器应用的场合，才能真正发挥这件武器的威力。

组合组件模式要解决的是这样一类问题：父组件想要传递一些信息给子组件，但是，如果用 props 传递又显得十分麻烦。

一看到这个问题描述，读者应该能立刻想到上一节我们介绍过的 Context API，利用 Context，可以让组件之间不用 props 来传递信息。

不过，使用 Context 也不是完美解法，上一节我们介绍过，使用 React 在 v16.3.0 之后提供的新的 Context API，需要让“提供者”和“消费者”共同依赖于一个 Context 对象，而且消费者也要使用 render props 模式。

如果不嫌麻烦，用 Context 来解决问题当然好，但是我们肯定会想有没有更简洁的方式。

问题描述

为了让问题更加具体，我们来解决一个实例。

很多界面都有 Tab 这样的元件，我们需要一个 `Tabs` 组件和 `TabItem` 组件，Tabs 是容器，TabItem 是一个一个单独的 Tab，因为一个时刻只有一个 TabItem 被选中，很自然希望被选中的 TabItem 样式会和其他 TabItem 不同。

这并不是一个很难的功能，首先我们想到的就是，用 Tabs 中一个 state 记录当前被选中的 Tabitem 序号，然后根据这个 state 传递 props 给 TabItem，当然，还要传递一个 `onClick` 事件进去，捕获点击选择事件。

按照这样的设计，Tabs 中如果要显示 One、Two、Three 三个 TabItem，JSX 代码大致这么写：

```
    <TabItem active={true} onClick={this.onClick}>One</TabItem>
    <TabItem active={false} onClick={this.onClick}>Two</TabItem>
    <TabItem active={false} onClick={this.onClick}>Three</TabItem>
```

上面的 TabItem 组件接受 `active` 这个 props，如果 `true` 代表当前是选中状态，当然可以工作，但是，也存在大问题：

1. 每次使用 TabItem 都要传递一堆 props，好麻烦；
2. 每增加一个新的 TabItem，都要增加对应的 props，更麻烦；
3. 如果要增加 TabItem，就要去修改 Tabs 的 JSX 代码，超麻烦。

我们不想要这么麻烦，理想情况下，我们希望可以随意增加减少 TabItem 实例，不用传递一堆 props，也不用去修改 Tabs 的代码，最好代码就这样：

```
   <Tabs>
     <TabItem>One</TabItem>
     <TabItem>Two</TabItem>
     <TabItem>Three</TabItem>
   </Tabs>
```

如果能像上面一样写代码，那就达到目的了。

像上面这样，Tabs 和 TabItem 不通过表面的 props 传递也能心有灵犀，二者之间有某种神秘的“组合”，就是我们所说的“组合组件”。

实现方式

上面我们说过，利用 Context API，可以实现组合组件，但是那样 TabItem 需要应用 render props，至于如何实现，读者可以参照上一节的介绍自己尝试。

在这里，我们用一种更巧妙的方式来实现组合组件，可以避免 TabItem 的复杂化。

我们先写出 TabItem 的代码，如下：

```
const TabItem = (props) => {
  const {active, onClick} = props;
  const tabStyle = {
    'max-width': '150px',
    color: active ? 'red' : 'green',
    border: active ? '1px red solid' : '0px',
  };
  return (
    <h1 style={tabStyle} onClick={onClick}>
      {props.children}
    </h1>
  );
};
```

TabItem 有两个重要的 props：`active` 代表自己是否被激活，`onClick` 是自己被点击时应该调用的回调函数，这就足够了。TabItem 所做的就是根据这两个 props 渲染出 `props.children`，没有任何复杂逻辑，是一个活脱脱的“傻瓜组件”，所以，用一个纯函数实现就可以了。

接下来要做的，就看 Tabs 如何把 `active` 和 `onClick` 传递给 TabItem。

我们再来看一下使用组合组件的 JSX 代码：

```
   <Tabs>
     <TabItem>One</TabItem>
     <TabItem>Two</TabItem>
     <TabItem>Three</TabItem>
   </Tabs>
```

没有 props 的传递啊，怎么悄无声息地把 `active` 和 `onClick` 传递给 TabItem 呢？

Tabs 虽然可以访问到作为 props 的 `children`，但是到手的 `children` 已经是创造好的元素，而且是不可改变的，Tabs 是不可能把创造好的元素再强塞给 `children` 的。

怎么办？

办法还是有的，如果 Tabs 并不去渲染 `children`，而是把 `children` 拷贝一份，就有机会去篡改这份拷贝，最后渲染这份拷贝就好了。

我们来看 Tabs 的实现代码：

```
class Tabs extends React.Component {
  state = {
    activeIndex:  0
  }

  render() {
    const newChildren = React.Children.map(this.props.children, (child, index) => {
      if (child.type) {
        return React.cloneElement(child, {
          active: this.state.activeIndex === index,
          onClick: () => this.setState({activeIndex: index})
        });
      } else {
        return child;
      }
    });

    return (
      <Fragment>
        {newChildren}
      </Fragment>
    );
  }
}
```

在 render 函数中，我们用了 React 中不常用的两个 API：

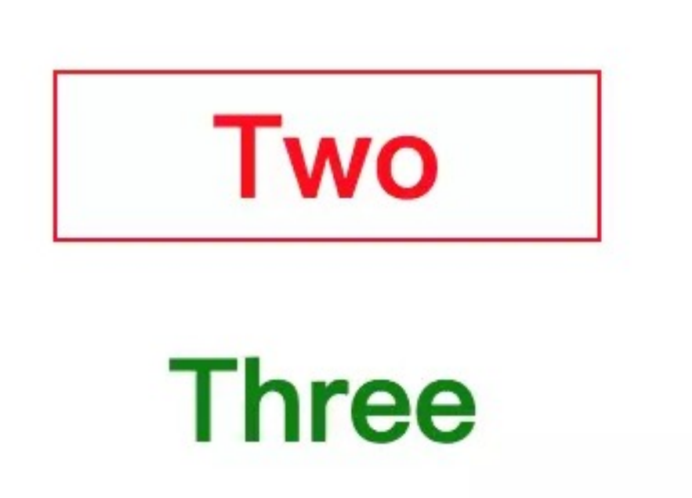
1. React.Children.map
2. React.cloneElement

使用 `React.Children.map`，可以遍历 `children` 中所有的元素，因为 `children` 可能是一个数组嘛。

使用 `React.cloneElement` 可以复制某个元素。这个函数第一个参数就是被复制的元素，第二个参数可以增加新产生元素的 props，我们就是利用这个机会，把 `active` 和 `onClick` 添加了进去。

这两个 API 双剑合璧，就能实现不通过表面的 props 传递，完成两个组件的“组合”。

最终的效果如下：



点击任何一个 TabItem，其样式就会立刻改变。而维护哪个 TabItem 是当前选中的状态，则是 Tabs 的责任。

实际应用

从上面的代码可以看出来，对于组合组件这种实现方式，TabItem 非常简化；Tabs 稍微麻烦了一点，但是好处就是把复杂度都封装起来了，从使用者角度，连 props 都看不见。

所以，应用组合组件的往往是共享组件库，把一些常用的功能封装在组件里，让应用层直接用就行。在 antd 和 bootstrap 这样的共享库中，都使用了组合组件这种模式。

如果你的某两个组件并不需要重用，那么就要谨慎使用组合组件模式，毕竟这让代码复杂了一些。

小结

这一小节介绍了组合组件（Compound Component）这种模式，这是一种比较高级的模式，如果要开发需要关联的成对组件，可以采用这个方案。

留言

评论将在后台进行审核，审核通过后对所有人可见

sanseo 前端工程师
jsx可以使用map生成对应的TabItem呀，为什么要一个个写TabItem
▲ 1 评论 18天前

xiari 高级前端工程师 @ 拼多多
antd 中用了很多这种实践。自身实践，在某次实现树形组件的时候就采用了组合方式，可以做些比如UI 和 画svg path等等的操作。
▲ 0 评论 25天前

yadong 前端开发 @ 知乎
React.Children.map(this.props.children, () => {}) 这种写法是啥意思，为啥第一个参数是this.props.children
▲ 0 收起评论 26天前

Artyhacker 前端 @ 北京某小国企
React官方api就是这样，我猜是重写了map方法吧，没细看这部分
20天前

评论审核通过后显示 评论

HaoliangWu 前端/全栈工程师 @ 云匠软件
您好，我一般在实际工作中，使用类似的模式时，没有通过文中提及的方式，是通过父组件使用 props 传递一个类似 register 的方法给子组件，然后子组件在生命周期中调用该方法将自身注册到父组件某个实例属性（比如 tabs）中，然后父组件利用该实例属性进行渲染。

之所以这么做的原因是因为，由于我使用 vue 和 angular 比较多，通常子组件是可以直接获取父组件的引用的，在 react 中我似乎没有找到合适的方式，因此就直接将父组件的方法通过 props 来传递了。

我的问题是，这种组合方式和文中提及的方式是一种模式吗？这种方式在 react 中是... 展开全部
▲ 0 收起评论 1月前

程墨 Hulu
要用上ref吗？如果用ref，那就算是anti-pattern了。
1月前

评论审核通过后显示 评论

zhangyanling77 前端开发 @ 成都
老师，那个child.type哪里来的？
▲ 0 收起评论 1月前

程墨 Hulu
child是一个函数参数，可以访问它的type字段
1月前

评论审核通过后显示 评论

请叫我王磊同学 前端工程师 @ 帆软软件
有的枪擅长狙击，有的枪适合近战，有的枪只是发个信号。
哈哈哈哈，一看吃鸡没少玩。
▲ 0 收起评论 1月前

程墨 Hulu
我比较old school，还真没有玩过吃鸡，话说，吃鸡游戏里有信号枪吗？
1月前

评论审核通过后显示 评论

Moorez 腾讯前端实习生 @ 腾讯
👍
▲ 0 评论 1月前

fantasy525 web前端 @ 某创业公司
这个思路适合用来写单选，多选按钮组件，当然其他的也可以，我第一次看是elementui的vue单选按钮组件实现方式
▲ 0 评论 1月前

portrait--东 web前端 @ 商状元
这个模式真的很可以，开阔了react操作的思路了
▲ 1 评论 1月前

blacker
这种方式只是把复杂度封装到了实现里面，根本的工作量并没有减少。另外，我还有个疑问:如果tabs组件重新渲染了，导致tabItem组件也重新渲染，因为使用了React.cloneElement，那么每次tabs组件重新渲染会不会都产生新的tabItem？即我的tabItem组件的componentDidUpdate还会不会执行？
▲ 0 收起评论 1月前

程墨 Hulu
如我所说，这种模式的好处就是封装复杂性，把麻烦留给组件开发者，把方便送给使用者。至于你提的问题，你其实可以自己尝试一下，在componentDidUpdate里面加上consule.log输出，就能看出结果来。
1月前

fantasy525 web前端 @ 某创业公司
感觉如果把click那里改一下就可以了，item组件无state就看props是否变化了，如果两个prop都不变就不会重复渲染，不过用vue就不用担心这个问题
1月前

评论审核通过后显示 评论

Farris 前端工程师
写得很棒，学习了~~
▲ 0 收起评论 1月前

程墨 Hulu
感谢！
1月前

评论审核通过后显示 评论

拖出去斩了
老师，问一下newChildren是不是把之前Tabs中的children替换调了。
▲ 1 收起评论 1月前

程墨 Hulu
newChildren只是一些对象，render函数把newChildren放在返回结果里，等于告诉React不要渲染原有children，改为渲染提供的newChildren。
1月前

评论审核通过后显示 评论

SonnyWu 前端菜鸟
老师,我有一个疑惑，为什么外层直接map tabItem呢?
▲ 0 收起评论 1月前

程墨 Hulu
抱歉，没大明白这个问题，你的意思是可以怎么写？
1月前

fantasy525 web前端 @ 某创业公司
回复 程墨：他意思是循环生成item
1月前

评论审核通过后显示 评论