React.js 小书

<-- 返回首页

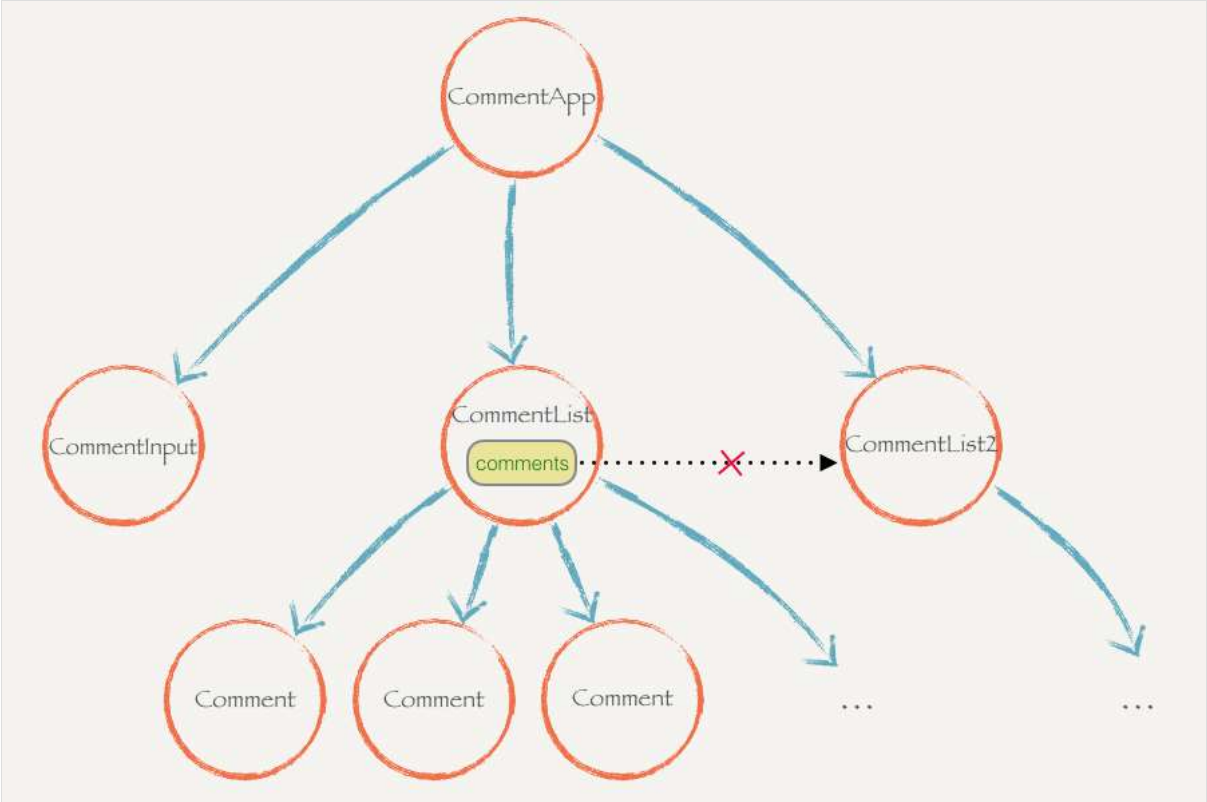
前端应用状态管理 -- 状态提升

- 作者：胡子大哈
- 原文链接： http://huziketang.com/books/react/lesson17

上一个评论功能的案例中，可能会有些同学会对一个地方感到疑惑： `CommentList` 中显示的评论列表数据为什么要通过父组件 `CommentApp` 用 `props` 传进来？为什么不直接存放在 `CommentList` 的 `state` 当中？例如这样做也是可以的：

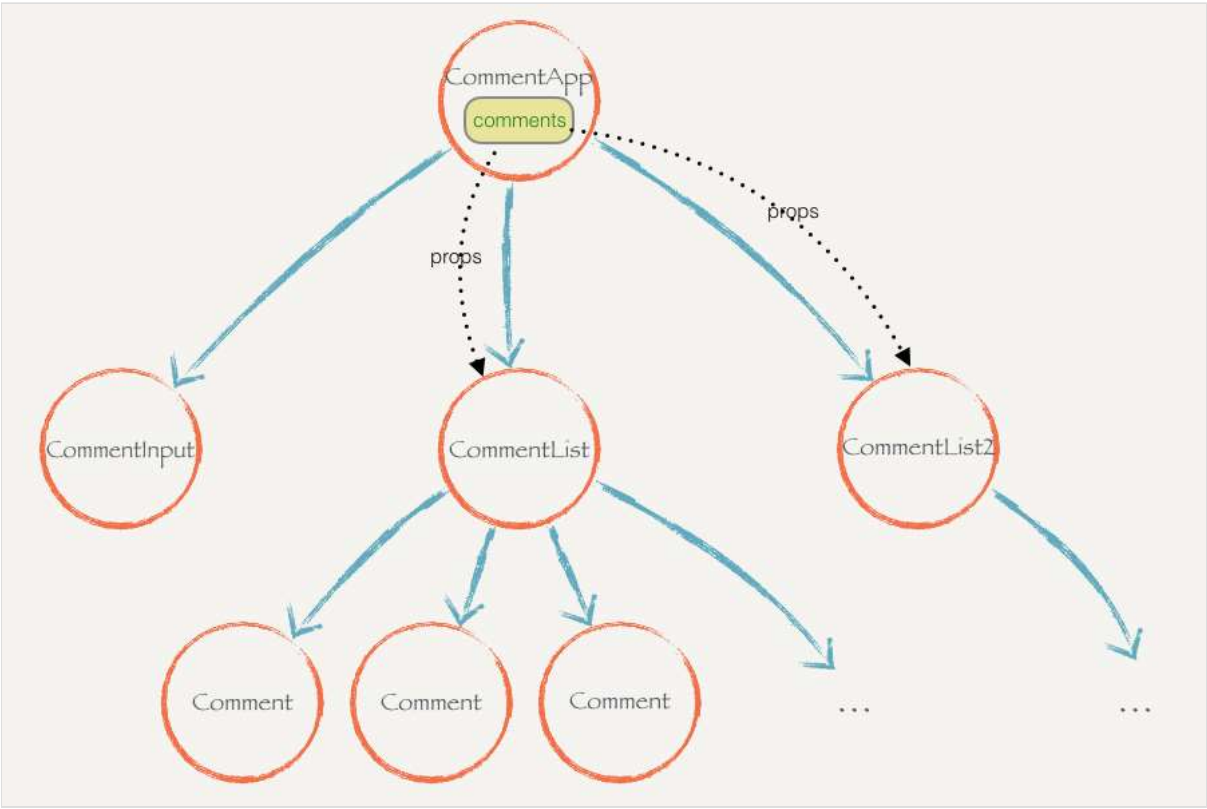
```
class CommentList extends Component {
  constructor () {
    this.state = { comments: [] }
  }

  addComment (comment) {
    this.state.comments.push(comment)
    this.setState({
      comments: this.state.comments
    })
  }

  render() {
    return (
      <div>
        {this.state.comments.map((comment, i) =>
          <Comment comment={comment} key={i} />
        )}
      </div>
    )
  }
}
```

如果把这个 `comments` 放到 `CommentList` 当中，**当有别的组件也依赖这个 `comments` 数据或者有别的组件会影响这个数据**，那么就带来问题了。举一个数据依赖的例子：例如，现在我们有另外一个和 `CommentList` 同级的 `CommentList2` ，也是需要显示同样的评论列表数据。



`CommentList2` 和 `CommentList` 并列为 `CommentApp` 的子组件，它也需要依赖 `comments` 显示评论列表。但是因为 `comments` 数据在 `CommentList` 中，它没办法访问到。

遇到这种情况，我们**将这种组件之间共享的状态交给组件最近的公共父节点保管**，然后通过 `props` 把状态传递给子组件，这样就可以在组件之间共享数据了。



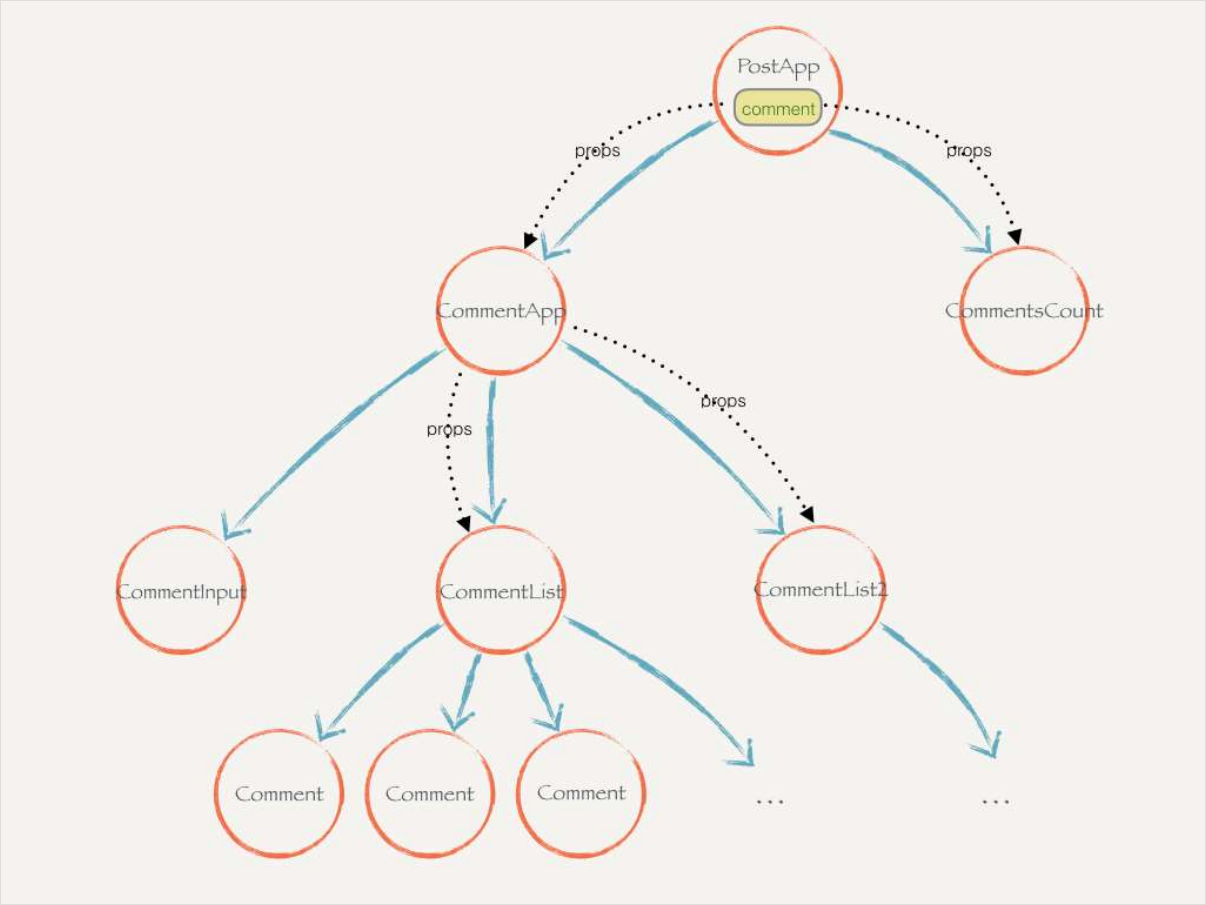
在我们的例子当中，如果把 `comments` 交给父组件 `CommentApp` ，那么 `CommentList` 和 `CommentList2` 都可以通过 `props` 获取到 `comments`，React.js 把这种行为叫

做“状态提升”。

但是这个 `CommentList2` 是我们临时加上去的，在实际案例当中并没有涉及到这种组件之间依赖 `comments` 的情况，为什么还需要把 `comments` 提升到 `CommentApp`？那是因为有个组件会影响到 `comments` ，那就是 `CommentInput`。`CommentInput` 产生的新的评论数据是会插入 `comments` 当中的，所以我们遇到这种情况也会把状态提升到父组件。

总结一下：当某个状态被多个组件**依赖**或者**影响**的时候，就把该状态提升到这些组件的最近公共父组件中去管理，用 `props` 传递**数据或者函数**来管理这种**依赖**或着**影响**的行为。

我们来看看状态提升更多的例子，假设现在我们的父组件 `CommentApp` 只是属于更大的组件树 `PostApp` 的一部分：



而这个更大的组件树的另外的子树的 `CommentsCount` 组件也需要依赖 `comments` 来显示评论数，那我们就只能把 `comments` 继续提升到这些依赖组件的最近公共父组件 `PostApp` 当中。

现在继续让我们的例子极端起来。假设现在 `PostApp` 只是另外一个更大的父组件 `Index` 的子树。而 `Index` 的某个子树的有一个按钮组件可以一键清空所有 `comments`（也就是说，这个按钮组件可以影响到这个数据），我们只能继续 `commenets` 提升到 `Index` 当中。

你会发现这种无限制的提升不是一个好的解决方案。一旦发生了提升，你就需要修改原来保存这个状态的组件的代码，也要把整个数据传递路径经过的组件都修改一遍，好让数据能够一层层地传递下去。这样对代码的组织管理维护带来很大的问题。到这里你可以抽象一下问题：

> 如何更好的管理这种被多个组件所依赖或影响的状态？

你可以看到 React.js 并没有提供好的解决方案来管理这种组件之间的共享状态。在实际项目当中状态提升并不是一个好的解决方案，所以我们后续会引入 Redux 这样的状态管理工具来帮助我们来管理这种共享状态，但是在讲解到 Redux 之前，我们暂时采取状态提升的方式来进行管理。

对于不会被多个组件依赖和影响的状态（例如某种下拉菜单的展开和收起状态），一般来说只需要保存在组件内部即可，不需要做提升或者特殊的管理。

课后练习

- 百分比换算器

> 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：挂载阶段的组件生命周期（一）

上一节：实战分析：评论功能（三）