

React.js 小书

[<-- 返回首页](#)

实战分析：评论功能（三）

- 作者：[胡子大哈](#)
- 原文链接：<http://huziketang.com/books/react/lesson16>
- 转载请注明出处，保留原文链接和作者信息。

接下来的代码比较顺理成章了。修改 `CommentList` 可以让它可以显示评论列表：

```
// CommentList.js
import React, { Component } from 'react'

class CommentList extends Component {
  render() {
    const comments = [
      {username: 'Jerry', content: 'Hello'},
      {username: 'Tomy', content: 'World'},
      {username: 'Lucy', content: 'Good'}
    ]

    return (
      <div>{comments.map((comment, i) => {
        return (
          <div key={i}>
            {comment.username}: {comment.content}
          </div>
        )
      })}</div>
    )
  }
}

export default CommentList
```

这里的代码没有什么新鲜的内容，只不过是建立了一个 `comments` 的数组来存放一些测试数据的内容，方便我们后续测试。然后把 `comments` 的数据渲染到页面上，这跟我们之前讲解的章节的内容一样——使用 `map` 构建一个存放 JSX 的数组。就可以在浏览器看到效果：

用户名：

评论内容：

发布

Jerry: Hello
Tomy: World
Lucy: Good

修改 `Comment.js` 让它来负责具体每条评论内容的渲染：

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

class Comment extends Component {
  render () {
    return (
      <div className='comment'>
        <div className='comment-user'>
          <span>{this.props.comment.username} </span>:
        </div>
        <p>{this.props.comment.content}</p>
      </div>
    )
  }
}

export default Comment
```

这个组件可能是我们案例里面最简单的组件了，它只负责每条评论的具体显示。你只需要给它的 `props` 中传入一个 `comment` 对象，它就会把该对象中的 `username` 和 `content` 渲染到页面上。

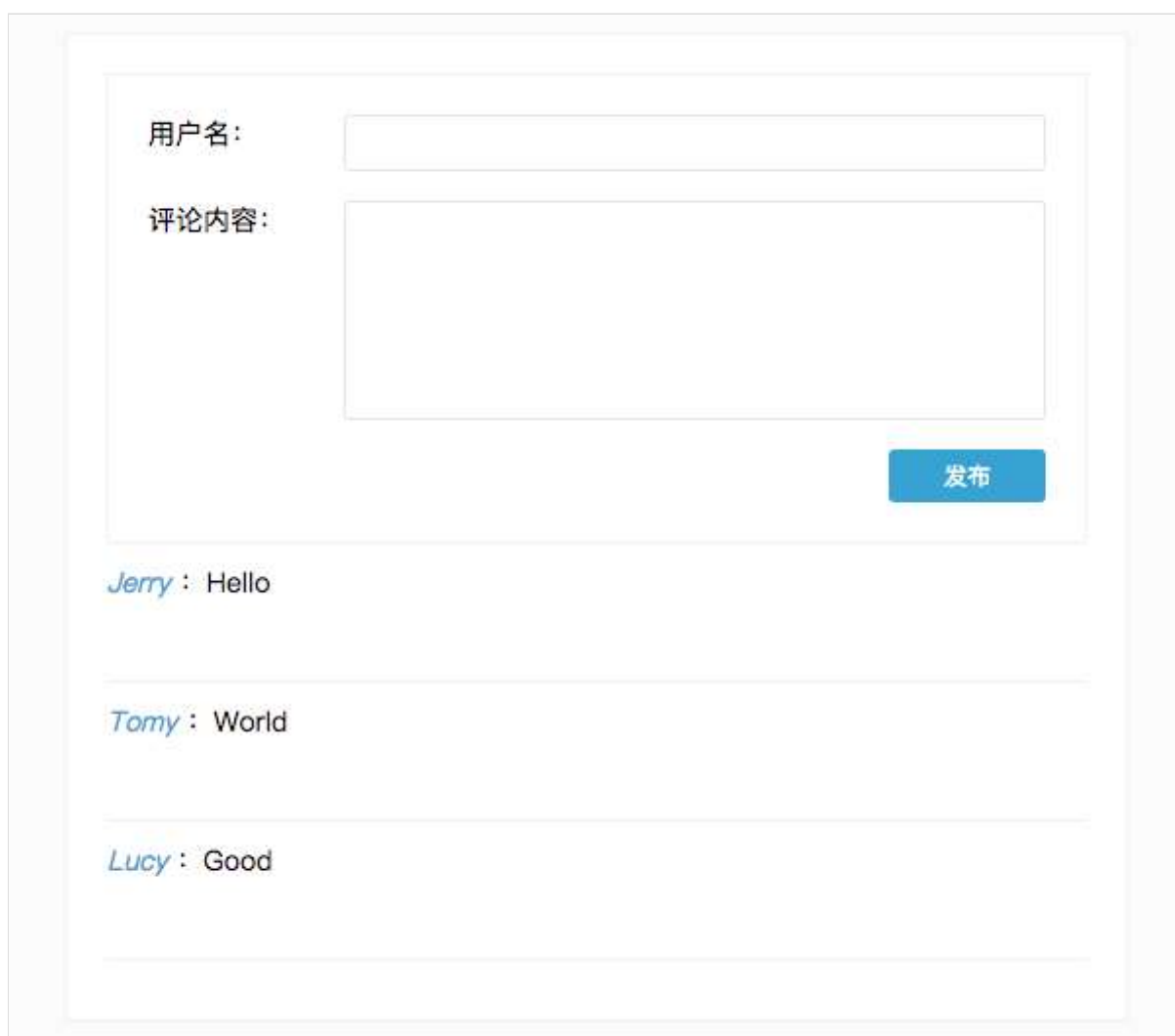
马上把 `Comment` 应用到 `CommentList` 当中，修改 `CommentList.js` 代码：

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

class CommentList extends Component {
  render() {
    const comments = [
```

```
{username: 'Jerry', content: 'Hello'},  
{username: 'Tomy', content: 'World'},  
{username: 'Lucy', content: 'Good'}  
]  
  
return (  
  <div>  
    {comments.map((comment, i) => <Comment comment={comment} key={i} />)}  
  </div>  
)  
}  
}  
  
export default CommentList
```

可以看到测试数据显示到了页面上：



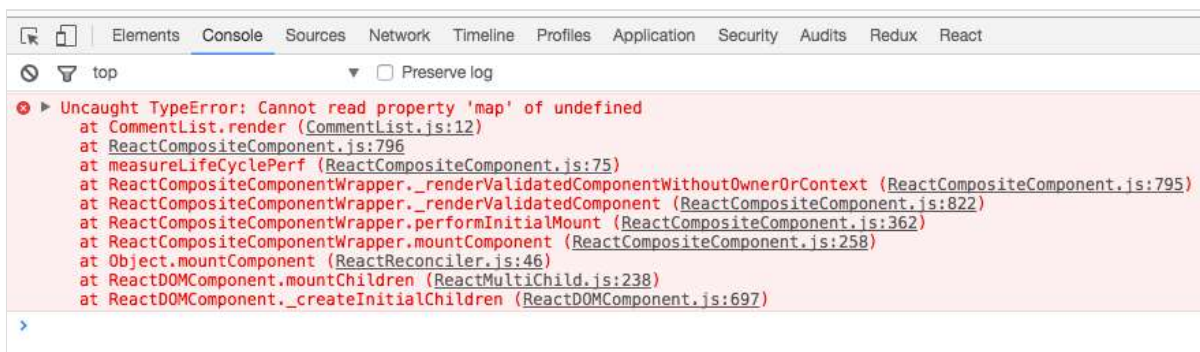
之前我们说过 `CommentList` 的数据应该是由父组件 `CommentApp` 传进来的，现在我们删除测试数据，改成从 `props` 获取评论数据：

```
import React, { Component } from 'react'  
import Comment from './Comment'  
  
class CommentList extends Component {
```

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

export default CommentList
```

这时候可以看到浏览器报错了：



这是因为 `CommentApp` 使用 `CommentList` 的时候并没有传入 `comments`。我们给 `CommentList` 加上 `defaultProps` 防止 `comments` 不传入的情况：

```
class CommentList extends Component {
  static defaultProps = {
    comments: []
  }
  ...
}
```

这时候代码就不报错了。但是 `CommentInput` 给 `CommentApp` 传递的评论数据并没有传递给 `CommentList`，所以现在发表评论时没有反应的。

我们在 `CommentApp` 的 `state` 中初始化一个数组，来保存所有的评论数据，并且通过 `props` 把它传递给 `CommentList`。修改 `CommentApp.js`：

```
import React, { Component } from 'react'
import CommentInput from './CommentInput'
import CommentList from './CommentList'

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

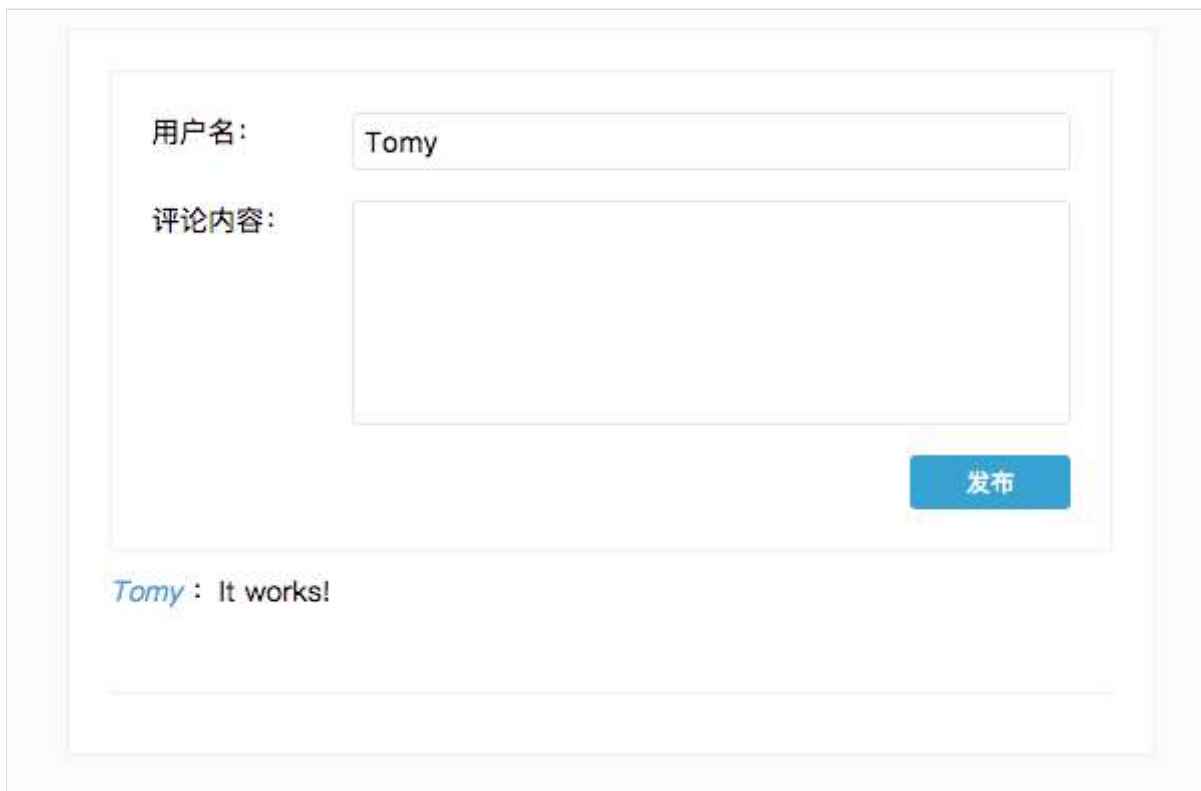
```
handleSubmitComment (comment) {  
  console.log(comment)  
}  
  
render() {  
  return (  
    <div className='wrapper'>  
      <CommentInput onSubmit={this.handleSubmitComment.bind(this)} />  
      <CommentList comments={this.state.comments}/>  
    </div>  
  )  
}  
}  
  
export default CommentApp
```

接下来，修改 `handleSubmitComment`：每当用户发布评论的时候，就把评论数据插入 `this.state.comments` 中，然后通过 `setState` 把数据更新到页面上：

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

小提示：这里的代码直接往 `state.comments` 数组里面插入数据其实违反了 React.js 的 [state 不可直接修改的原则](#)。但其实这个原则是为了 `shouldComponentUpdate` 的优化和变化的跟踪，而这种目的在使用 `React-redux` 的时候其实会自然而然达到，我们很少直接手动地优化，这时候这个原则就会显得有点鸡肋。所以这里为了降低大家的理解成本就不强制使用这个原则，有兴趣的朋友可以参考：[Tutorial: Intro To React - React](#)。

现在代码应该是可以按照需求正常运作了，输入用户名和评论内容，然后点击发布：



为了让代码的健壮性更强，给 `handleSubmitComment` 加入简单的数据检查：

```
...
handleSubmitComment (comment) {
  if (!comment) return
  if (!comment.username) return alert('请输入用户名')
  if (!comment.content) return alert('请输入评论内容')
  this.state.comments.push(comment)
  this.setState({
    comments: this.state.comments
  })
}
...
```

到这里，我们的第一个实战案例——评论功能已经完成了！完整的案例代码可以在这里 [comment-app](#) 找到， [在线演示](#) 体验。

总结

在这个案例里面，我们除了复习了之前所学过的内容以外还学习了新的知识点。包括：

1. 实现功能之前先理解、分析需求，划分组件。并且掌握划分组件的基本原则——可复用性、可维护性。
2. 受控组件的概念，React.js 中的 `<input />`、`<textarea />`、`<select />` 等元素的 `value` 值如果是受到 React.js 的控制，那么就是受控组件。
3. 组件之间使用 `props` 通过父元素传递数据的技巧。

当然，在真实的项目当中，这个案例很多地方是可以优化的。包括组件可复用性方面（有没有发现其实 `CommentInput` 中有重复的代码？）、应用的状态管理方面。但在这里为了给大家总结和演示，实现到这个程度也就足够了。

到此为止，React.js 小书的第一阶段已经结束，你可以利用这些知识点来构建简单的功能模块了。但是在实际项目如果要构建比较系统和完善的功能，还需要更多的 React.js 的知识还有关于前端开发的一些认知来协助我们。接下来我们会开启新的一个阶段来学习更多关于 React.js 的知识，以及如何更加灵活和熟练地使用它们。让我们进入第二阶段吧！

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：前端应用状态管理 — 状态提升](#)

[上一节：实战分析：评论功能（二）](#)

如果你觉得小书写得还不错，可以请胡子大哈喝杯茶 :)

赞赏

或者传播一下知识也是一个很好的选择

7 条评论，5 人参与。



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



最是奢侈年少梦 • 4月27日 20:37
请问你们谁知道，这个脚手架，路由怎么配置？
1顶 • 回复 • 分享»



偉 • 4月22日 16:59
发布了，打印台有信息，页面却没有渲染出来
顶 • 回复 • 分享»



anye • 4月10日 10:38
已解决
顶 • 回复 • 分享»



anye • 4月7日 17:40
求解答。。
顶 • 回复 • 分享»



anye • 4月7日 17:34
是版本问题吗？最后运行时候报错了 Uncaught TypeError: Cannot read property 'push' of undefined
顶 • 回复 • 分享»



幸福未满 -anye • 4月17日 17:02
请问如何解决的？谢谢！
顶 • 回复 • 分享»



全美第一瓜 • 4月1日 11:27
如何上传的博客？
1顶 • 回复 • 分享»

友言？