

React.js 小书

[<-- 返回首页](#)

## 实战分析：评论功能（二）

- 作者：[胡子大哈](#)
- 原文链接：<http://huziketang.com/books/react/lesson15>
- 转载请注明出处，保留原文链接和作者信息。

[上一节](#)我们构建了基本的代码框架，现在开始完善其他的内容。

### 处理用户输入

我们从 `ComponentInput` 组件开始，学习 React.js 是如何处理用户输入的。首先修改 `ComponentInput.js`，完善 `ComponentInput` 的 `render` 函数中的 HTML 结构：

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

class CommentInput extends Component {
  render () {
    return (
      <div className='comment-input'>
        <div className='comment-field'>
          <span className='comment-field-name'>用户名: </span>
          <div className='comment-field-input'>
            <input />
          </div>
        </div>
        <div className='comment-field'>
          <span className='comment-field-name'>评论内容: </span>
          <div className='comment-field-input'>
            <textarea />
          </div>
        </div>
        <div className='comment-field-button'>
          <button>
            发布
          </button>
        </div>
      </div>
    )
  }
}

export default CommentInput
```

在浏览器中可以看到 `ComponentInput` 的结构和样式都已经生效：

用户名:

评论内容:

发布

CommentList

因为还没有加入处理逻辑，所以你输入内容，然后点击发布是不会有什​​么效果的。用户可输入内容一个是用户名（username），一个是评论内容（content），我们在组件的构造函数中初始化一个 `state` 来保存这两个状态：

```
...
class CommentInput extends Component {
  constructor () {
    super()
    this.state = {
      username: '',
      content: ''
    }
  }
  ...
}
```

然后给输入框设置 `value` 属性，让它们的 `value` 值等于 `this.state` 里面相应的值：

```
...
<div className='comment-field'>
  <span className='comment-field-name'>用户名: </span>
  <div className='comment-field-input'>
    <input value={this.state.username} />
  </div>
</div>
<div className='comment-field'>
  <span className='comment-field-name'>评论内容: </span>
  <div className='comment-field-input'>
    <textarea value={this.state.content} />
  </div>
</div>
```

```
    </div>
    ...

```

可以看到接受用户名输入的 `<input />` 和接受用户评论内容的 `<textarea />` 的 `value` 值分别由 `state.username` 和 `state.content` 控制。这时候你到浏览器里面去输入内容看看，你会发现你什么都输入不了。

这是为什么呢？React.js 认为所有的状态都应该由 React.js 的 `state` 控制，只要类似于 `<input />`、`<textarea />`、`<select />` 这样的输入控件被设置了 `value` 值，那么它们的值永远以被设置的值为准。值不变，`value` 就不会变化。

例如，上面设置了 `<input />` 的 `value` 为 `this.state.username`，`username` 在 `constructor` 中被初始化为空字符串。即使用户在输入框里面尝试输入内容了，还是没有改变 `this.state.username` 是空字符串的事实。

所以应该怎么做才能把用户内容输入更新到输入框当中呢？在 React.js 当中必须要用 `setState` 才能更新组件的内容，所以我们需要做的就是：监听输入框的 `onChange` 事件，然后获取到用户输入的内容，再通过 `setState` 的方式更新 `state` 中的 `username`，这样 `input` 的内容才会更新。

```
    ...
    <div className='comment-field-input'>
      <input
        value={this.state.username}
        onChange={this.handleChange.bind(this)} />
    </div>
    ...

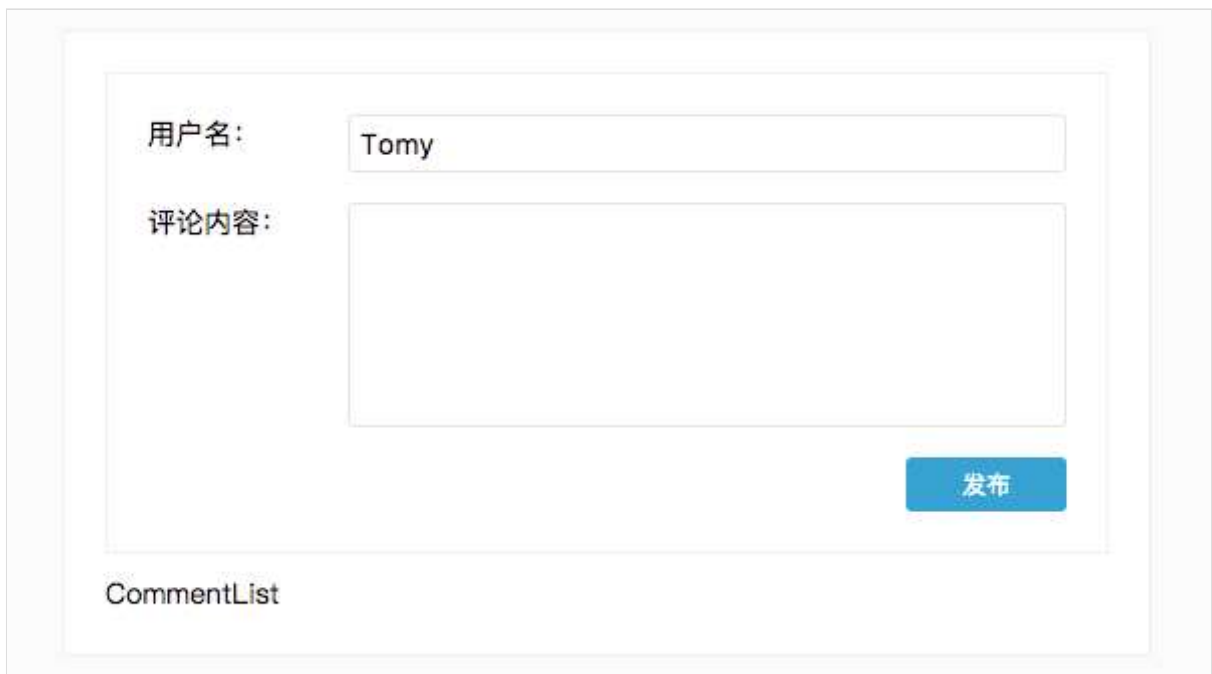
```

上面的代码给 `input` 加上了 `onChange` 事件监听，绑定到 `this.handleChange` 方法中，该方法实现如下：

```
    ...
    handleChange (event) {
      this.setState({
        username: event.target.value
      })
    }
    ...

```

在这个方法中，我们通过 `event.target.value` 获取 `<input />` 中用户输入的内容，然后通过 `setState` 把它设置到 `state.username` 当中，这时候组件的内容就会更新，`input` 的 `value` 值就会得到更新并显示到输入框内。这时候输入已经没有问题了：



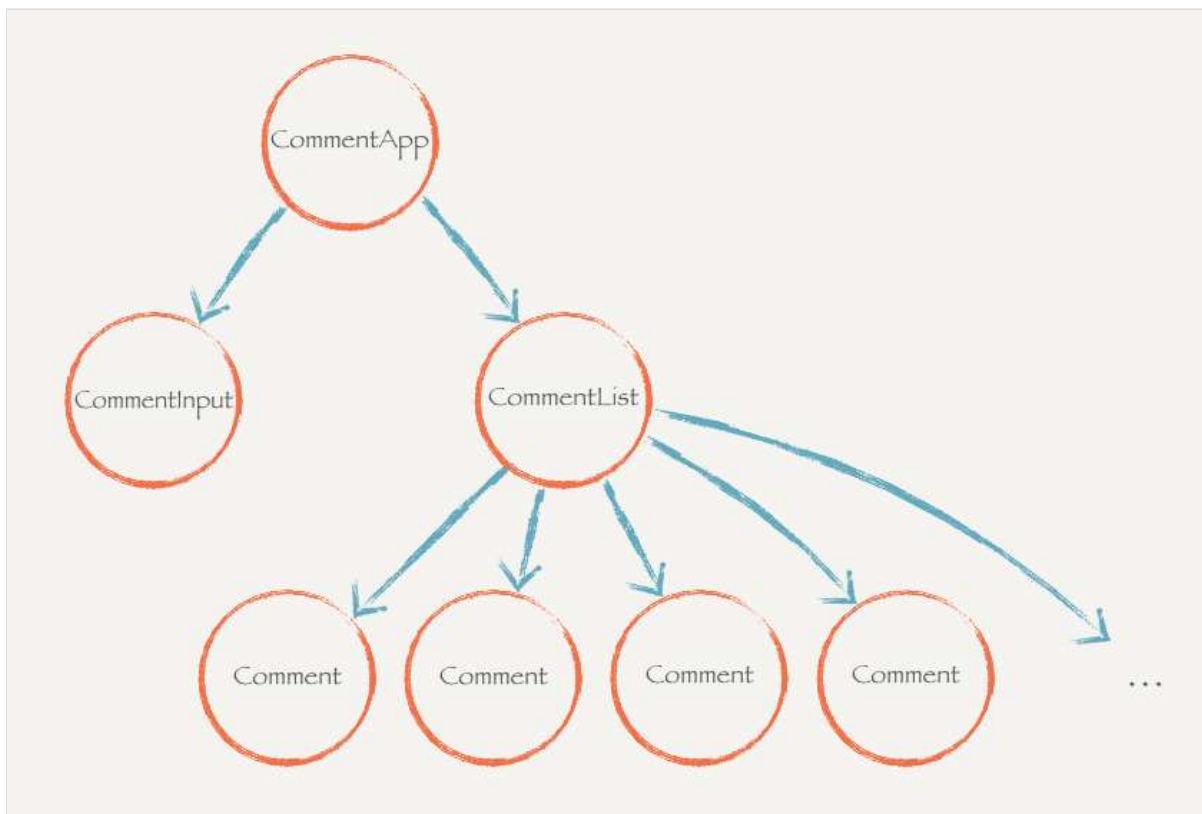
类似于 `<input />`、`<select />`、`<textarea>` 这些元素的 `value` 值被 `React.js` 所控制、渲染的组件，在 `React.js` 当中被称为受控组件（Controlled Component）。对于用户可输入的控件，一般都可以让它们成为受控组件，这是 `React.js` 所推崇的做法。另外还有非受控组件，这里暂时不提及。

同样地，让 `<textarea />` 成为受控组件：

```
...
  handleContentChange (event) {
    this.setState({
      content: event.target.value
    })
  }
...
  <div className='comment-field'>
    <span className='comment-field-name'>评论内容: </span>
    <div className='comment-field-input'>
      <textarea
        value={this.state.content}
        onChange={this.handleContentChange.bind(this)} />
    </div>
  </div>
  ...
```

## 向父组件传递数据

当用户在 `CommentInput` 里面输入完内容以后，点击发布，内容其实是需要显示到 `CommentList` 组件当中的。但这两个组件明显是单独的、分离的组件。我们再回顾一下之前是怎么划分组件的：



可以看到，`CommentApp` 组件将 `CommentInput` 和 `CommentList` 组合起来，它是它们俩的父组件，可以充当桥接两个子组件的桥梁。所以当用户点击发布按钮的时候，我们就将 `CommentInput` 的 `state` 当中最新的评论数据传递给父组件 `CommentApp`，然后让父组件把这个数据传递给 `CommentList` 进行渲染。

`CommentInput` 如何向 `CommentApp` 传递的数据？父组件 `CommentApp` 只需要通过 `props` 给予组件 `CommentInput` 传入一个回调函数。当用户点击发布按钮的时候，`CommentInput` 调用 `props` 中的回调函数并且将 `state` 传入该函数即可。

先给发布按钮添加事件：

```
...  
    <div className='comment-field-button'>  
      <button  
        onClick={this.handleSubmit.bind(this)}>  
        发布  
      </button>  
    </div>  
...
```

用户点击按钮的时候会调用 `this.handleSubmit` 方法：

```
...  
handleSubmit () {  
  if (this.props.onSubmit) {  
    const { username, content } = this.state  
    this.props.onSubmit({username, content})  
  }  
  this.setState({ content: '' })  
}
```

```
}  
...
```

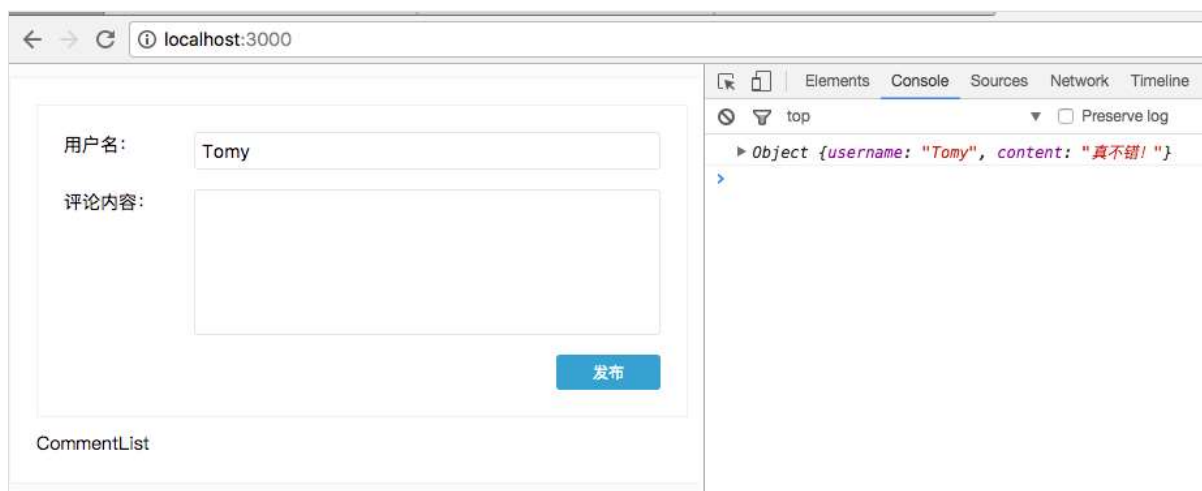
`handleSubmit` 方法会判断 `props` 中是否传入了 `onSubmit` 属性。有的话就调用该函数，并且把用户输入的用户名和评论数据传入该函数。然后再通过 `setState` 清空用户输入的评论内容（但为了用户体验，保留输入的用户名）。

修改 `CommentApp.js`，让它可以通过传入回调来获取到新增评论数据：

```
class CommentApp extends Component {  
  handleSubmitComment (comment) {  
    console.log(comment)  
  }  
  
  render() {  
    return (  
      <div className='wrapper'>  
        <CommentInput  
          onSubmit={this.handleSubmitComment.bind(this)} />  
        <CommentList />  
      </div>  
    )  
  }  
}
```

在 `CommentApp` 中给 `CommentInput` 传入一个 `onSubmit` 属性，这个属性值是 `CommentApp` 自己的一个方法 `handleSubmitComment`。这样 `CommentInput` 就可以调用 `this.props.onSubmit(...)` 把数据传给 `CommentApp`。

现在在 `CommentInput` 中输入完评论内容以后点击发布，就可以看到 `CommentApp` 在控制台打印的数据：



这样就顺利地把数据传递给了父组件，接下来我们开始处理评论列表相关的逻辑。

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：实战分析：评论功能（三）](#)

[上一节：实战分析：评论功能（一）](#)

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

0 条评论，0 人参与。

★ 2



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

还没有评论

友言?