文 React.js 的介绍 - 针对了解 jQuery 的工程师（译）

web前端开发 react.js 墨白 2015年08月27日发布

> 这是一篇国外上了hacker news头条的文章，作者利用一个小案例的将jQuery以及React进行了对比，解释了React的优势，将这篇文章翻译过来，期望能够同大家一起进步~原文地址，下面是正文：

我也听说React.js非常棒，最近也花了一些时间来研究它。现在我使用React起来感觉非常舒服，我决定在这一方面写一个简单的教程。

目标人群：了解 jQuery 的前端工程师

在开始之前，我想声明一下我写这篇文章的受众。

“learn code the hard way”系列的作者Zed Shaw 最近写了一篇非常棒的博客叫做“Early v.s. Beginning Coders”，在这篇博客里，Zed批评了那些声称他们的教程是完完全全适合初学者的编程培训人员，实际上，这些教程对于大多数初学者来说是不适合的。

我不想犯同样的错误。对于没有尝试过React的工程师里面，有些人喜欢使用像backbone、Ember或者Angular之类的JS框架，有些人对于JavaScript非常了解，有些人仅仅了解jQery。对一类工程师有效的教程未必对于另一类也有用。

在这个教程里面，我这篇文章的受众是我上面提到的第三种类型的工程师：了解jQuery的工程师，详细点：

- 可以做简单的HTML/CSS/jQuery编写的设计者 知道怎样使用jQuery插件的开发者
- 依赖Bootstrap以及简单的jQuery来实现简单的前端效果的后端开发者
- 任何在编写JavaScript代码时习惯复制粘贴而不是自己编写的人

如果你已经习惯自己编写JavaScript代码或者使用其他的前端框架例如Backbone\Ember\Angular，这个教程并不适合你，我的代码风格也会让你疑惑。有很多其他非常好的教程值得你去学习，例如React官方教程

同样的，如果你对React已经足够了解，那么你可能会认为这个教程很低级，因为我在其中大部分都是些的关于React的state的知识，没有涉及到组件以及其他的知识。

我认为这个教程的最适宜受众是那么习惯了使用jQuery的工程师，这门教程可以告诉他们React的优势在哪里~

好的，让我们开始吧~

预计花费时间：1~2小时

如果你学习能力非常好（并且复制粘贴代码而不是自己敲的话），这个教程应该会花费你大概一个小时。如果你用心研读并且自己敲代码的话，这个教程应该会花费你大概两个小时的时间。

如果你被某一处所困扰，可以做下面的：

- 在本页的底部发表你的评价
- 给我发email： shu@chibicode.com
- 在Twitter上联系我：@chibicode
- 在GIthub上面post你的疑问：Github链接

本文的主要任务：利用React编写一个Tweet Box

许多React教程从解释React的工作原理以及为什么React是如此神奇开始，但我的教程不是！

相反，我们会直接建立一个简单的UI项目，选择的工具是jQuery和React，利用这个项目，我们来解释它们之间的不同点，我相信看完这个教程之后你能够思考的更远而不仅仅只是编写了一个UI项目。

我们将要做的UI与推特上面的Tweet Box很像，当然，我们对它的功能进行了精简，希望你能够觉得这个例子能够帮助你更好的了解React~

步骤一：关于JSBin的介绍

（第一步可以略过，各位可以在自己的本地编辑器按照教程的步骤编写就行）

首先介绍这个教程将要使用的一个在线支持HTML/CSS/JS代码的编辑器：JSBin，它可以在线支持jQuery以及React.js代码。你可能比较熟悉相似的工具：codepen或者JSFiddle，选择JSBin的原因仅仅只是我的习惯。

JSBin的示例

左边是HTML语句，右边是即时生成的结果**(建议在自己的编辑器上面直接编写，尝试过有些效果JSBin上面不能完全显示)**

建立一个JSBin的账号

建议你创建一个JSBin的账号，在菜单栏上面点击Login或者Register就可以。

在创建账号之后，你就可以克隆JSBin的开源的代码到你的账户，就如同Github一样，让我们来尝试一下，点击JSBin菜单栏上面的“Save”按钮

如果你在JSBin的网站上面，你可以在菜单栏选择“Add Library”来引入流行的css或者JS框架

可以尝试做以下的事情：

- 点击“Add Library”来添加最新的Bootstrap
- 在button标签上面添加类名“btn btn-primary”

JSBin的示例：JSBin示例1

反馈回来的结果是这样的：

左侧的代码如下：

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery.min.js"></script>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" rel="st
ylesheet" type="text/css" />
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script
>
  <meta charset="utf-8">
  <title>JS Bin</title>
</head>
<body>
  <button class="btn btn-primary">Button</button>
</body>
</html>
```

下面来创建一个Tweet Box：

现在你应该已经习惯了JSBin的用法了吧？好的，让我们来创建一个Tweet Box吧~仍然是在之前的代码中，改变<html>中<body>的内容：

```
<div class="well clearfix">
```

```
  <textarea class="form-control"></textarea><br/>
  <button class="btn btn-primary pull-right">Tweet</button>
</div>
```

我们将会使用BootStrap中的类名，如：`form-control`、`well`、`clearfix`等等，但是这些仅仅只是为了美观，跟本教程没有任何关系，下面是代码以及结果：

JSBin示例2

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery.min.js"></script>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" rel="st
ylesheet" type="text/css" />
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script
>
  <meta charset="utf-8">
  <title>JS Bin</title>
</head>
<body>
  <div class="well clearfix">
    <textarea class="form-control"></textarea><br/>
    <button class="btn btn-primary pull-right">Tweet</button>
  </div>
</body>
</html>
```

步骤二：第一个需求---“Tweet”按钮未输入的时候是不能点击的

现在，该用一些JS来实现交互了，我们会满足以下的需求：

需求一：Tweet按钮初始状态应该是不能点击的。只有当输入框中有字符输入，按钮才能够点击

下面就是一个demo

为了满足这个需求，在之前的代码中需要添加一些jQuery代码：

```
// 初始化状态
$("button").prop("disabled", true);

// 文本框的值发生变化时
$("textarea").on("input", function() {
  // 只要超过一个字符，就
  if ($(this).val().length > 0) {
    // 按钮可以点击
    $("button").prop("disabled", false);
  } else {
    //否则，按钮不能点击
    $("button").prop("disabled", true);
  }
});
```

代码解释：

- 我使用的是标签名选择器：`button`还有`textarea`，在这个例子中没有必要给他们添加class类名或者id
- 为了实现按钮不能点击，使用代码`$(...).prop(disabled, ...)`
- 为了监听`textarea`的变化，需要使用`input`事件，这个标准浏览器上面都能得到支持

尝试在文本框中输入一些字符，注意观察button状态的改变

如果你感到困惑，建议多研究一下jQuery代码的使用方法，再来继续学习本教程。有很多学习jQuery很好的网站，例如Codecademy、Treehouse、Code school以及国内的慕课网。

第一个需求以及完成了，接下来我们会利用React来实现同样的效果。

步骤三：使用React.js来制作Tweet Box

对于React，首先你需要了解的是你会在js中写标签，而不需要在HTML文档中。

下面来展示一下我说的是什么意思。下面是利用React.js所呈现的同样Tweet Box的代码

建议：你不需要明白代码的含义，仅仅尝试读一下代码就可以了

```
var TweetBox = React.createClass({
  render: function() {
    return (
      <div className="well clearfix">
        <textarea className="form-control"></textarea>
        <br/>
        <button className="btn btn-primary pull-right">Tweet</button>
      </div>
    );
  }
});

React.render(
  <TweetBox />,
  document.body
);
```

一些注意事项：

- `return (...)`不是JavaScript代码，而是一个HTML代码。在React中，你会使用一个特殊的语法叫做JSX，它可以让你在JavaScript代码插入HTML代码。
- 其中的HTML代码仅仅只是与平常我们所编写的HTML代码有一些相像，它们之间还是有一些区别的。注意，它使用`className`而不是`class`，但它们非常像，所以你可以很快的学会它们。
- 浏览器不能解析JSX语法，所以当React运行你的JSX代码的时候，React会自动解析其中的HTML成JavaScript代码，从而使浏览器能够解析。
- 在主文档的HTML结构的`<body></body>`标签中不需要写入HTML标签，我们会在JavaScript中写入标签

常问的问题以及解答：

问题：`React.createClass`还有`React.render`有什么作用？我需要现在就明白它们吗？
回答：现在你不需要对这个感到担心。`React.createClass`创建一个有名字的UI组件（在这个例子中，就

是`TweetBox`）。然后通过`React.render(< TweetBox />,document.body)`插入body这一个DOM节点中，现在你了解这些就足够了。

问题：在本地，我需要做一些特殊的事情去写JSX吗?
回答：是的，但是你只需要引入一个叫做JSX Transformer这个文件就行了。而在JSBin中，你只是需要增加一个React库，下面对这个会有介绍。

问题：在同一个区域将js还有HTML混在一起写，不是说是一种坏的风格吗?
回答：对于简单的页面而言，这样写可能会是一种坏的风格，但是对于大型的网页应用就不一定了。在大型的网页应用中，会有成百上千的UI，每一个UI都包含他自己的标签以及行为。对于每一个UI而言，如果标签还有事件行为放在一起，会更加利于维护。React就是设计用来制作大型的网页应用。众所周知的，React是全球最大的网页应用公司FaceBook开源并且使用的。

下面我会带着你一步一步的来写上面展示的React的代码。

步骤四：写你的第一个React代码

首先，制作了一个简单的HTML页面，我在其中引入了React以及Bootstrap，代码如下：
JSBin示例1

```
<!DOCTYPE html>
<html>
<head>
<script src="//fb.me/react-0.13.1.js"></script>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" rel="st
ylesheet" type="text/css" />
  <meta charset="utf-8">
  <title>JS Bin</title>
</head>
<body>

</body>
</html>
```

在JSBin上面输入上面的代码之后，打开JavaScript标签选择“JSX(React)”

本地操作方法：在本地文档上可以在安装React之后，引入JSXTransformer文件*

```
    <script src="../build/react.js"></script>
    <script src="../build/JSXTransformer.js"></script>
    <script type="text/jsx"></script>
```

上面是本地使用React的示例，具体的大家可以参考网上的其他教程，这里就不过多赘述了

现在你可以来写一些React代码了。可以尝试跟我一起敲下面的代码：

```
var TweetBox = React.createClass({
  render: function() {
  }
});
```

上面的代码是创建以UI组件的主体，它就像是jQuery中的`$(function(){...})`一样重要。

为了构建一个UI，我们必须在`render()`方法中输入一些代码。现在，我们来用`div`标签来做一些简单的事情

```
var TweetBox = React.createClass({
  render: function() {
    return (
      <div>
        Hello World!
      </div>
    );
  }
});
```

如上面所示，在`return`后面加一对括号`(...)`，然后在里面写上标签。

JSX的一些注意事项

对于JSX，你需要记住一件事情：`render()`语句中，在`return(...)`只能有一个返回的大的闭合标签

下面的示例不会起作用（**没有返回标签**）：

```
return (
  Hello World!
);
```

下面的示例也不会起作用**(有两个返回标签)**：

```
return (
  <span>
    Hello
  </span>
  <span>
    World
  </span>
);
```

将UI组建插入到DOM节点：

现在我们想将上面的“hello world”UI组件插入到DOM中，我们需要加上代码React.render()在我们刚刚写的代码下面：

```
var TweetBox = React.createClass({
  render: function() {
    return (
      <div>
        Hello World!
      </div>
    );
  }
});

React.render(
  <TweetBox />,
```

```
  document.body
);
```

`React.render`接收两个参数，第一个是UI对象，就是`<VariableName />`,第二个参数是DOM对象(在这个例子中指`document.body`)。将两个参数结合到一起，代表着将`TweetBox`这个UI组件插入到`body`里面。

现在你应该看到`Hello World!`出现了，恭喜，你已经成功的写出了第一个React组件！

下面，我们来写我们真正的`TweetBox`组件

现在，我们在TweetBox里面插入真正的HTML结构，而不是`Hello World`。在之前`render`代码的基础上面替换成下面的代码。

```
return (
  <div className="well clearfix">
    <textarea className="form-control"></textarea>
    <br/>
    <button className="btn btn-primary pull-right">Tweet</button>
  </div>
);
```

这里有**两件事情**需要注意:

1. 不要使用`class`，使用`className`，因为JSX代码最终要翻译成JS代码, 而`class`在最新版的JS标准中属于保留字.
2. 如果你只是用`<br>`，而不是`<br />`,这个标签不会起作用。记住一定要在自闭合标签中加上`/`

其他的应该跟之前的jQuery的示例相同。

现在你应该可以在JSBin中看到TweetBox了。如果什么都没有，也许你应该仔细的检查一下你的代码。

这就是步骤四，这是这一步的JSBin。（可能JSBin不能呈现出我们想要的效果，我依旧建议自己在本地敲这些代码）。

步骤五：利用React实现tweet按钮的第一个需求：初始状态禁用

首先，我们利用jQuery来实现这样的需求。

需求一：按钮初始状态禁用，当输入框内有字符输入的时候，按钮可以进行点击。

下面是我们写的完成这个需求的jQuery代码：

```
// Initially disable the button
$("button").prop("disabled", true);

// When the value of the text area changes...
$("textarea").on("input", function() {
  // If there's at least one character...
  if ($(this).val().length > 0) {
    // Enable the button.
    $("button").prop("disabled", false);
  } else {
    // Else, disable the button.
    $("button").prop("disabled", true);
  }
});
```

下面来看看我们怎么用React来完成这样的需求：

接着前面的JSBin的代码示例开始：

首先，让我们给按钮设置初始状态：

```
render: function() {
  return (
    ...
    <button className="..." disabled>Tweet</button>
    ...
  );
}
```

按钮现在的状态应该是禁用的了。

请注意，我们是这样利用jQuery来实现同样的功能。

```
$("button").prop("disabled", true);
```

现在，我们要实现文本框内有字符，button按钮可用的需求。

处理状态改变的事件

首先，我们需要等待文本的输入。利用jQuery可以这样写：

```
$("textarea").on("input", function() {
  ...
}
```

而在React中，我们需要写一个事件处理方法，就叫它：`handleChange`吧

```
React.createClass({
  handleChange: function(event) {
  },
  render: function() {
    ...
  }
});
```

接下来，当文本框发生改变的时候，我们调用这个方法。为此，我们要对`textarea`标签做些修改：

```
<textarea className="form-control"
          onChange={this.handleChange}></textarea>
```

- 在jQuery中，我们使用`input`事件，而在React中，我们使用`onChange`。接下来，你会从React文档中接触到事

件在React JSX中的不同用法，所以不用太过担心。

- 更重要的是，我们在JSX文档的HTML语句中使用`{...}`语法来处理JavaScript代码。在这个示例中，因为`handleChange`在UI组件中是一个方法，所以我们在`handleChange`的前面添加了`this`来调用它。
- 如果你已经习惯了编写jQuery代码，那么这看起来是一个非常不好的代码风格。再次申明，无须担心这个问题。在大型应用中，由于标签以及事件都组合在一个，代码会更加易于阅读以及修改。

为了确保`handleChange`方法被调用，我们在方法中加入`console.log`语句。

```
handleChange: function(event) {
  console.log(event.target.value);
},
```

`event`对象包含`target`，即是`textarea`，我们使用`.value`来输出`textarea`的值。

在你的JSBin中，打开`console`控制台来查看输出，然后，在文本框中随便输入一些字符。

你也可以在这里尝试JSBin console尝试。

这就是步骤五的所有啦。tips：验证成功，可以在JSBin中关闭`console`，在下一步，我们不再需要它。

步骤六： 事件状态的应用(重点)

接下来，我会重点解释jQuery代码风格以及React代码风格的最大不同。

在jQuery中，当某些事件发生的时候，你会经常性改变DOM(像我们之前做的那样)：

在React中，你并不直接改变DOM。不同的是，当事件发生时，你改变的是“状态”，而这，是通过调用`this.setState`来实现。

之后，当每次“状态”改变的时候，`render`会被调用，在调用的过程中，可以改变“状态”

这就是在事件发生时更新UI的过程。没错，它确实比较难懂，所以，接下来，我会通过代码来解释这一过程。

写一个事件处理方法

接着上一步的JSBin的代码。首先，我们需要初始化“状态”这一对象。如果没有这一步，什么作用都没有。

首先，我们需要写一个特别的方法叫做`getInitialState`，它是React自带的方法，它会返回一个JS对象，即初始状态。

在这个JS对象中，我们要做些什么呢？让我们在其中创建一个`key`叫做`text`。

```
var TweetBox = React.createClass({
  getInitialState: function() {
    return {
      text: ""
    };
  },
  handleChange: ...
  render: ...
});
```

接下来，我们会设置一个事件处理器将“状态”中的`text`设置成文本框内的内容。我们利用一个叫做`setState`的内置函数将文本框里的内容传递给`text`。

```
handleChange: function(event) {
  this.setState({ text: event.target.value });
},
```

现在，让我们来利用一些简单的debug语句检查一下是不是正确设置了text。

只需要在靠近`render`末尾处添加`this.state.text`，并且使用`{...}`语句来调用JSX中的js代码。

```
render: function() {
  return (
    <div ...>
      ...
      <button ...>Tweet</button>

      <br/>
      {this.state.text}
    </div>
  )
}
```

可以尝试在文本框中输入一些文本，同样的内容应该出现的按钮下方。你也可以在JSBin上面尝试。

现在，你应该对之前的内容有了深一点的理解。

删除之前的debug代码

一旦你确认了“状态”被正确设置，删除之前添加的debug代码。

```
<br/>
{this.state.text}
```

可用/禁用 按钮

到这里，我们可以稍微暂停一下，观察按钮的状态随着文本框的内容有无而改变。

通过“状态”，我们可以使用以下逻辑:

- 如果`this.state.text.length === 0`，按钮应该是被禁用的。在React中，添加`disabled`属性，并且设置返回值`this.state.text.length === 0`。因为它是js代码，我们需要把它`“{}”`起来。

```
<button className="btn btn-primary pull-right"
        disabled={this.state.text.length === 0}>Tweet</button>
```

如果你在原生的HTML中写`disabled="true"`或者`disabled="false"`。在原生的HTML中，你需要移除`disabled`属性来启用按钮。但是React不是原生的HTML，它遵循的是下面的原则：

- 如果你在JSX中使用`disabled={true}`，它会编译成`<button ... disabled>`
- 如果你在JSX中使用`disabled={false}`，`disabled`属性会从`button`标签中移除。

这个同样适用于其它布尔类型的属性，例如`checked`，暂时这还不是正式的官方文档的写法，但是它应该会很快被添加进去。

现在的代码示例：JSBin。

小结
在进入下一个步骤之前，请牢记React和jQuery的区别：

- 在jQuery中，每发生一个事件，就需要改变一次DOM
- 在React中，每发生一个事件，只会改变`state`“状态”，通过`render`来映射现在的状态

步骤七：jQuery实现提示文本中剩余字数功能

下一个功能就是文本中剩余字数的提示：

功能说明：

- 字符字数会这样显示`140 - the number of characters entered.`

我们会首先利用jQuery实现这样的功能，之后再利用React。

接着我们上一步的jQuery代码，React代码暂时放到一边。从现在开始，每一个小章节，我都会给出新的代码。这意味，在你看完每一步之后，都可以自己好好阅读这些代码。

JSBin代码示范

首先，在HTML两种通过添加`span`标签来增加字符字数，请看下面的示范：

```
<textarea ...></textarea><br>
<span>140</span>
<button ...>Tweet</button>
```

在JS代码中添加以下：

```
$("textarea").on("input", function() {
  $("span").text(140 - $(this).val().length);
  ...
});
```

好了，尝试输入字符，你会发现提示字符数会发生改变。

查看完整的JSBin示例

步骤八：React实现提示文本中剩余字数功能

怎么用React实现跟上面一样的功能呢？也许，你可以自己尝试一下。

tips：你可以在JSBin上面尝试自己动手编写。

在这一步中，你不需要操作HTML。建议你在JSBin中关闭它。

代码编写建议：

- 没有必要去改变`getInitialState()`或者`handleChange()`
- 可以尝试在`render()`中操作`this.state.text.length`

参考代码：
在`render()`的`<br/>`后面添加

```
<span>{140 - this.state.text.length}</span>
```

这是到现在为止的JSBin代码链接

感觉很简单？不明白为什么React比jQuery优秀这么多？好吧，下一步更加复杂了，而那才是React真正的瑰宝。

步骤九："Add Photo"按钮

让我们在这个UI组件上面添加一个"Add Photo"按钮。之后的事情就有些有趣了。

不过，我们不会真的上传图片。下面的，才是我们需要做的。

当你在推特上传图片的时候，它会自动帮你计算剩下的可以输入的字符。而你每次上传一次图片，它将会占用23个你原本可以输入的字符字数。

仿照推特，接下来我们所要做的就是：

- 创建一个"Add Button"按钮
- 点击按钮的时候，可以切换它的状态。如果它的状态是"on"，按钮会显示`✓ Photo Added`
- 如果按钮的状态是"on"，那么可以输入的字符的个数会下降23个
- 同样，如果按钮的状态是"on"，即使没有文本输入，那么"Tweet"按钮也是可以点击的

JSBin中的demo，尝试点击"Add Photo"按钮，然后观察可输入字数以及"Tweet"按钮的变化

步骤十：利用jquery实现跟上一步同样的功能

接着之前写过的jQuery代码继续~

首我们将会同时修改HTML以及js。在这之前，我们使用了一个选择器`$("button")`，但是如果有两个`button`标签的话就不太好用了。

接下来，我们要开始修改HTML如下：

```
...
<button class="js-tweet-button btn btn-primary pull-right" disabled>Tweet</button>
<button class="js-add-photo-button btn btn-default pull-right">Add Photo</button>
...
```

下面来详细解释一下改变了那些内容：

- 添加了叫做"Add Photo"的按钮
- 分别给按钮添加类名`js-tweet-button`以及`js-add-photo-button`，给它们的前缀增加了'js'，是因为这些只会在js中使用，在css中不会使用。

接下来，重新写js文件如下：

```
$("textarea").on("input", function() {
  $("span").text(140 - $(this).val().length);

  if ($(this).val().length > 0) {
    $(".js-tweet-button").prop("disabled", false);
  } else {
    $(".js-tweet-button").prop("disabled", true);
  }
});
```

来详细解析一下有哪些变化：

- (重要)因为要给Tweet按钮增加`disabled`属性，所以我在第一行移除了`$("button").prop("disabled", true);`
- 用`$(".js-tweet-button")`替换了`$("button")`，这样就可以跟`.js-add-photo-button`区分开来

增加按钮的功能

我们开始来增加新的功能：

- 点击`Add Photo`按钮来改变其状态。如果按钮的状态是"on"，那么就让其显示`✓ Photo Added`

为了达到这样的效果，让我们来增加下面的代码：

```
$("textarea").on("input", function() {
  ...
});

$(".js-add-photo-button").on("click", function() {
  if ($(this).hasClass("is-on")) {
    $(this)
      .removeClass("is-on")
      .text("Add Photo");
  } else {
    $(this)
      .addClass("is-on")
      .text("✓ Photo Added");
  }
});
```

我们使用类名`is-on`来检测按钮的状态，你可以点击或者仔细校对代码来理解它的原理。

递减字符计数

接下里，让我们来实现这样的需求：

- 如果"Add Photo"按钮的状态是"on"，那么可输入字符数会减少23

为了完成这样的需求，编辑代码如下：

```
if ($(this).hasClass("is-on")) {
  $(this)
    .removeClass("is-on")
    .text("Add Photo");
  $("span").text(140 - $("textarea").val().length);
} else {
  $(this)
    .addClass("is-on")
    .text("✓ Photo Added");
  $("span").text(140 - 23 - $("textarea").val().length);
}
```

每次点击的时候，我们都会改变`span`中的文本内容。如果按钮的状态是"on",那么可输入字符数就从117（可输入字符总数140-图片所占的字符数23）开始计数。

你可以点击"Add Photo"来检查上面的代码是否起作用

处理可输入字符提示的功能

上面的还远没有结束。即使"Add Photo"按钮的状态是"on"，你在文本框里面输入字符，提示的剩余可输入字符并不会同步发生改变。

为了解决这个问题，我们需要修改一下有关textarea的代码如下：

```
$("textarea").on("input", function() {
  if ($(".js-add-photo-button").hasClass("is-on")) {
    $("span").text(140 - 23 - $(this).val().length);
  } else {
    $("span").text(140 - $(this).val().length);
  }

  if (...) {
    ...
});
```

现在，你可以点击"Add Photo"之后，在文本框里面输入一些字符检查一下功能是否实现。

我知道这已经花费了好长一段时间了...

请坚持下去，现在jQuery的代码看起来已经有点让人迷糊了，不过没关系，多看几遍你应该就能理解了。

最后一个功能的完善

最后一个需要完善的功能是：

- 如果"Add Photo"的状态是"on"，即使没有文本输入，"Tweet"按钮也应该可以使用。

为了完成这个需求，我们需要修改"Add Photo"按钮的click事件：

```
$(".js-add-photo-button").on("click", function() {
  if ($(this).hasClass("is-on")) {
    ...
    if ($("textarea").val().length === 0) {
      $(".js-tweet-button").prop("disabled", true);
    }
  } else {
    ...
    $(".js-tweet-button").prop("disabled", false);
  }
});
```

代码解析：

- 如果"Add Photo"按钮的状态从"on"改变成"off"(`if`语句)，我们需要检查是否有字符输入，如果没有，禁用按钮
- 如果"Add Photo"按钮的状态从"off"改变成"on"(`else`语句)，我们需要检查是否有字符输入，如果有，按钮启用

打断一下

我们还没有结束，下面的几步会发现代码的一个bug，你可以自己先尝试解决一下：

- 点击"Add Photo"按钮，使其状态从"off"改变成"on"
- 输入一些字符

- 删除所有的字符
- 你会发现，因为"Add Photo"按钮的状态是"on",所以"Tweet"按钮应该仍然是可以启用的，但是现状并不是这样的

这意味我们关于文本框的代码少了某些逻辑，为了解决这个bug，我们需要再加一个`if`判断

```
$("textarea").on("input", function() {
  ...
  if ($(this).val().length > 0 || $(".js-add-photo-button").hasClass("is-on")) {
    ...
  } else {
    ...
  }
});
```

我们增加了一个判断去检查"Tweet"按钮是否应该禁用

你可以多试几次，这次应该没有问题了。

步骤十一：关于jQuery代码如此让我困惑的反思

这是完整jQuery代码

再一次查看你的jQuery代码，很让人困惑，不是吗？如果你一直这样写下去，你就不得不借助大量的注释来帮助你记住之前的代码做了哪些。更进一步，这些代码也不利于你之后重构你的项目。

那么问题来了：为什么这么快利用jQuery所写的代码就变得这么“丑陋”?

这就不得不谈到我们之前说起的jQuery的风格。

再看这幅图:

当只有一个事件而且处理对象只有一个DOM的时候，利用jquery写出的代码是非常简单的。然而，当多个事件多个处理对

象的时候，利用jQuery写的代码就有点恶心了。

想象一下，如果有更多的事件，上面的图片中会出现更多的箭头，与此同时，代码变得更加不好管理了。

理论上，你可以减轻这些缺陷通过重构出复用的代码，但是你仍然不得不每次增加新功能的时候，绞尽脑汁处理好它们之间的关系。

现在，让我们来看看在React中做同样的事情是多么简单！

步骤十二：利用React实现"Add Photo"按钮功能

接着你之前的代码开始

首先，在JSX中添加"Add Photo"按钮：

```
<button ...>Tweet</button>
<button className="btn btn-default pull-right">Add Photo</button>
```

给按钮添加一个事件，使其可以改变按钮的显示内容从`Add Photo`到`✓ Photo Added`。温习一下React的代码风格：

我们将会：

- 创建一个状态变量用来追踪"Add Photo"按钮的状态是"on"还是"off"
- 使用`render()`中的状态来决定按钮使显示`Add Photo`还是`✓ Photo Added`
- 在click事件发生时改变其状态

对于第一步，我们会构造一个`getInitialState`方法，在其中设置一对键值对来记录图片是否上传：

```
getInitialState: function() {
  return {
```

```
    text: "",
    photoAdded: false
  };
},
```

我们会对JSX中的"Add Photo"button标签做一些改变。我们可以通过下面的语句使其实现如果`this.state.photoAdded`是true，那么button上面就会显示"Photo Added":

```
<button className="btn btn-default pull-right">
  {this.state.photoAdded ? "✓ Photo Added" : "Add Photo" }
</button>
```

对于第三步，我们会在JSX中给文本框增加一个点击事件:

```
<button className="btn btn-default pull-right"
  onClick={this.togglePhoto}>
  {this.state.photoAdded ? "✓ Photo Added" : "Add Photo" }
</button>
```

并且增加一个方法`this.state.photoAdded`用来改变button按钮的状态:

```
togglePhoto: function(event) {
  this.setState({ photoAdded: !this.state.photoAdded });
},
```

现在，你可以尝试点击一下，你会发现"Add Photo"显示的内容会随着你的点击而改变。

可输入字符提示

我们将会接着实现下面的功能:

- 如果"Add Photo"按钮的状态是"on"，可输入字符串将会减少23个

现在，可输入字符串在`render()`中是这样控制的：

```
<span>{140 - this.state.text.length}</span>
```

现在，可输入字符数也依赖`this.state.photoAdded`，所以我们需要添加`if-else`语句

然而，在JSX中，你不能直接在`{...}`写入`if-else`语句，你可以使用三元运算符，但是那样的话在这个项目里面也太长了。

正常最简单的做法就是写一个方法来判断状态，我们来尝试一下。

首先，重新编辑`span`标签的代码：

```
<span>{ this.remainingCharacters() }</span>
```

并且，这样定义方法：

```
remainingCharacters: function() {
  if (this.state.photoAdded) {
    return 140 - 23 - this.state.text.length;
  } else {
    return 140 - this.state.text.length;
  }
},
```

现在，"Add photo"的状态应该可以影响可输入剩余字数了。

问题：在`render()`中为什么`{ this.remainingCharacters() }`后面有`()`，而`{ this.handleChange }`以及`{ this.togglePhoto }`没有呢?

好问题，让我们再看一次`render()`：

```
render: function() {
  return (
    ...
      <textarea className="..."
                onChange={ this.handleChange }></textarea>
      ...
      <span>{ this.remainingCharacters() }</span>
      ...
      <button className="..."
        onClick={ this.togglePhoto }>
        ...
      </button>
    </div>
  );
```

参考答案:

- 我们写`remainingCharacters()`这个方法来返回一个数，我们需要得到这个数并让它在`span`标签中显示，所以我们需要通过`()`调用`remainingCharacters()`
- 另一方面，`handleChange`以及`togglePhoto`是事件处理方法，只有当交互（改变文本或者点击按钮）实现的时候，我们才需要调用这些方法。于是，我们不需要写`()`，只需要把他们赋值给`onChange`以及`onClick`这样的属性

Tweet按钮的状态

我们仍然有一个需求需要完善:

- 如果"Add Photo"按钮的状态是"on"，但是没有字符输入，"Tweet"按钮应该可以启用:

这很简单，之前"Tweet"按钮的禁用属性是这样设置的:

```
<button ... disabled={this.state.text.length === 0}>...</button>
```

简而言之，要将之前只要字符输入为0的时候，"Tweet"按钮禁用改成只要:

- 字符串输入为0

- "Add Photo"按钮的状态是"off"

两者都满足的时候"Tweet"按钮禁用

所以逻辑就变成了这样：

```
<button ... disabled={this.state.text.length === 0 && !this.state.photoAdded}>...</button>
```

或者，你可以调用`remainingCharacters()`来简化代码。如果未输入字符为140，意味着没有文本输入且"Add Photo"按钮的状态为"off"，从而"Tweet"按钮应该禁用：

```
<button ... disabled={this.remainingCharacters() === 140}>...</button>
```

就是这样的，尝试点击"Add Photo"按钮来检查"Tweet"按钮是否设置正确。

我们完成了

看起来非常简单，这是完整的代码：

```
var TweetBox = React.createClass({
  getInitialState: function() {
    return {
      text: "",
      photoAdded: false
    };
  },
  handleChange: function(event) {
    this.setState({ text: event.target.value });
  },
  togglePhoto: function(event) {
    this.setState({ photoAdded: !this.state.photoAdded });
  },
  remainingCharacters: function() {
    if (this.state.photoAdded) {
      return 140 - 23 - this.state.text.length;
    } else {
```

```
      return 140 - this.state.text.length;
    }
  },
  render: function() {
    return (
      <div className="well clearfix">
        <textarea className="form-control"
                  onChange={this.handleChange}></textarea>
        <br/>
        <span>{ this.remainingCharacters() }</span>
        <button className="btn btn-primary pull-right"
```

步骤十三：关于React代码的思考-为什么如此简单?

使用React，对于"Add Photo"按钮的代码所做的改变是最小的，不需要进行重构，这是为什么呢?

这与React的代码风格思想有很大的关系。在React中，只有当事件发生的时候，`state`才发生改变，之后，React自动调用`render()`来更新UI。

在这个特殊的例子中，它是这样发生的:

state成为了事件以及`render()`之间过渡:

- 每个事件不需要担心哪一部分的DOM发生变化，他们只需要设置`state`就可以了
- 相应的，当你写`render()`的时候，你也只需要担心现在的`state`是什么

与jQuery做比较:

你可以想象一下，当UI有更多交互的时候，会发生什么，没有了中间的过渡层'state'，我们需要花费很大的精力来解决它们之间相互的联系。这就是为什么对于复杂的组件，建议使用React而不是jQuery。

再次说明，你也可以写出简洁的jQuery代码。但是你必须想出良好的代码结构，每次想要增加新功能的时候还需要特别注意是否影响代码的重构。而使用React，就没有这样的烦恼。

步骤十四：最后的功能需求-超出字数高亮显示

最后一个功能就是超过所限制输入的字符，超出的字符高亮显示：

但是，我们不会真的在输入框的内部高亮显示，因为这需要我们把`textarea`改成`contenteditable`，而`contenteditable`在这个示例中处理起来有点复杂。

我们会在输入框的上面添加一个弹出框，并且显示应该删除的超出字符限制的字符：

你可以尝试copy下面乔布斯的一段话：

```
If you haven't found it yet, keep looking.
Don't settle.
As with all matters of the heart, you'll know when you find it.
And, like any great relationship, it just gets better and better as the years roll on.
```

并且复制进下面的JSBin的链接：高亮显示的代码链接

- 它应该在文本框的上面有一个弹出框，超出字数限制的字符应该红色高亮显示
- 在红色高亮显示的字符之前应该还有10个字符没有高亮

如果我们用jQuery写这个功能，我们的代码会更加的混乱。看下面的图，我们需要重新增加两个箭头

所以我们不会用jQuery来实现这样的一个功能。

我们只会用React来实现它，只需要增加一个箭头就好了：

步骤十五：利用React高亮显示字符

接着之前的React代码

我们会一步一步来做。首先，当输入的字符超过限制字数的时候，会弹出一个简单的弹出框，它里面会有一些简单的文字。

因为它需要条件判断，所以我们需要写一个程序来处理。在文本框之前增加`{ this.overflowAlert() }`：

```
{ this.overflowAlert() }
<textarea className="form-control"
          onChange={this.handleChange}></textarea>
```

现在，这个方法应该返回：

- 如果没有字符串遗留，则返回一个div标签
- 否则，没有任何东西返回

在React中，你可以通过函数方法返回一个JSX标签，并且在其他的方法中使用它。换而言之，你可以尝试做下面的事情：

```
someMethod: function() {
  return (
    <a href="#">Hello World</a>
  );
},
someMethod2: function() {
  return (
    <h1>
      { this.someMethod() }
    </h1>
  );
},
```

在我们的例子中，我们可以在一种情况下返回`( <div> ... </div> )`，另一种情况下，什么也不返回。这样我们的`overflowAlert`方法如下所示：

```
overflowAlert: function() {
  if (this.remainingCharacters() < 0) {
    return (
      <div className="alert alert-warning">
        <strong>Oops! Too Long:</strong>
      </div>
    );
  } else {
    return "";
  }
},
```

注意，我们是通过检查`this.remainingCharacters()`来判断我们是否弹出提示框。

尝试在文本框里面输入超过140个字符（或者一张图片+113个字符），它将会弹出提示框。

对超出的字符进行颜色样式的处理

下面是超出字符的处理方式：

- 在"Oops! Too Long:"和超出的字符之间，存在一个小空格，它在三个'.'之前。我使用了` `
- 提示框中有`this.state.text`第131到第140个字符
- 其余的就是红色高亮显示的字符

下面在JSX中修改代码，在`overflowAlert`添加一个`if`判断，我们会创建两个变量：`beforeOverflowText`以及`overflowText`，我们会在`this.state.text`中使用`.substring()`方法：

```
if (this.remainingCharacters() < 0) {
  var beforeOverflowText = this.state.text.substring(140 - 10, 140);
  var overflowText = this.state.text.substring(140);

  return (
    <div className="alert alert-warning">
      <strong>Oops! Too Long:</strong>
       ...{beforeOverflowText}
      <strong className="bg-danger">{overflowText}</strong>
```

```
      </div>
    );
  }
```

再次copy下面的语句，粘贴进文本框，你就可以看到高亮的文本了，我们基本上以及完成了。

> If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it. And, like any great relationship, it just gets better and better as the years roll on.

如果"Add Photo"按钮的状态是"on"呢?

如果"Add Photo"按钮的状态是"on"，那么字符限制就要减少23，所以我们的`beforeOverflowText`以及`overflowText`需要考虑到这个情况:

```
if (this.state.photoAdded) {
  var beforeOverflowText = this.state.text.substring(140 - 23 - 10, 140 - 23);
  var overflowText = this.state.text.substring(140 - 23);
} else {
  var beforeOverflowText = this.state.text.substring(140 - 10, 140);
  var overflowText = this.state.text.substring(140);
}
```

jsbin完整代码示范

现在，尝试点击"Add Photo"查看显示效果吧~

让我们来看看改变了什么:

步骤十六：接下来呢?

以上就是我的教程，期望你能从中得到:

- jQuery代码和React代码的不同
- 怎么在JSX中写一些基本的React代码

接下来呢?

我的教程里面仅仅只是包含了React中的最好的部分。接下来你需要学的是:

- 怎么编写组件
- 在`state`中使用极广的`props`

幸运的是，官方的教程以及足够好了，你可以好好研读它们。

感谢阅读，如果你有任何的疑问，可以给我发email: shu@chibicode.com

后记

其实距离自己读完这篇文章已经好久了，一直在拖拖拖。作者介绍了React中的瑰宝`state`，确实让我受益很多。前前后后翻译这篇文章快一个月了吧。今天才算真正理解了翻译的不易~

2015年08月27日发布

你可能感兴趣的文章

JavaScript 就要统治世界了? 171 收藏，15.3k 浏览

SegmentFault 2015 Top Rank 296 收藏，7.3k 浏览

SegmentFault 专访 | AlloyTeam 2015 前端技术大会讲师圆桌 8 收藏，914 浏览

讨论区

翻译的很赞，React中的state的确让人耳目一新！

jayki · 2015年08月28日

翻译流畅，辛苦

sdhjl2000 · 2015年08月28日

贴图都没了

zwxajh · 2015年08月29日

回复 zwxajh:
我这边都有啊

墨白 · 2015年08月29日

回复 墨白:
你好, 好像是本机DNS解析的设定的问题, 已经解决了.

zwxajh · 2015年08月29日

不要使用 `class`，使用 `className` 后面的翻译我觉的不合适, 这样会不会好些 因为**JSX**代码最终要翻译成**JS**代码，而 `class` 在最新版的**JS**标准中属于保留字.

iSayme · 2015年08月30日

回复 iSayme:
好的，我看看怎么修改好读一些。有些地方确实时间比较匆忙，感谢指正~

墨白 · 2015年08月30日

写得很棒 我是真正的作者口中的 只会使用jQuery的小白 而且两者比较写得很棒 文章写得很流畅 谢谢分享！

WitNesS · 2015年08月30日

回复 iSayme:
ddddddd

penghuangit · 2015年08月30日

回复 penghuangit:
ddd是什么意思?...

iSayme · 2015年08月30日

回复 iSayme:
已修改，感谢指正。。。另外吐个槽，鄙视只收藏不点赞的

墨白 · 2015年08月31日

回复 墨白:
还好我点赞了~~

iSayme · 2015年08月31日

很不错的文章，感谢分享

johnnyhappy365 · 2015年09月02日

在React中，每发生一个时间，只会改变state“状态”，通过render来映射现在的状态

事件的字写错了?

咕噜噜噜小kefu · 2015年09月04日

回复 咕噜噜噜小kefu:
sorry，太急了，明天改

墨白 · 2015年09月05日

为什么我感觉用angularjs的directive实现起来更方便呢 可能是我熟悉angularjs而不熟悉reactjs吧 哈哈

杨佰 · 2015年09月06日

很漂亮的教程，非常实用，谢谢。

啊六不准 · 2015年09月07日

回复 杨佰:
两家公司的思路不同而已，他们各自都有自己的优势跟劣势

墨白 · 2015年09月07日

回复 墨白:
好吧 完全是个人喜好了！

杨佰 · 2015年09月08日

回复 咕噜噜噜小kefu:
已修改，感谢指正

墨白 · 2015年09月08日

翻译有个小问题，步骤十里那句“The "Tweet" button should still be enabled because the "Add Photo" button is ON, but this isn't the case.”，“but this isn't the case”的前文是此时还没有实现的效果，所以其应该译为“但是现状并不是这样”而不是“但是我们的设想不是这样的”，不然意思就相反了。

Levon · 2015年09月13日

回复 Levon:
感谢提醒，已经改正，公司代码重构，最近忙死了，抱歉这么晚才回复您~

墨白 · 2015年09月21日

喜欢那张jquery和react的关于state的对比图

think2011 · 2015年11月08日

棒！

frabbit · 2015年11月11日

lz 辛苦了 谢谢分享！

雲_旋律 · 2015年11月26日

赞

dreamyboy · 1月18日

请先 登录 后评论

本文隶属于专栏

杂七杂八的感想

关于前端工作的杂七杂八的想法，欢迎一起讨论

墨白
作者

关注专栏

分享扩散:

文章目录