React.js 小书

<-- 返回首页

# 2. 前端组件化（一）：从一个简单的例子讲起


- 作者：胡子大哈
- 原文链接： http://huziketang.com/books/react/lesson2



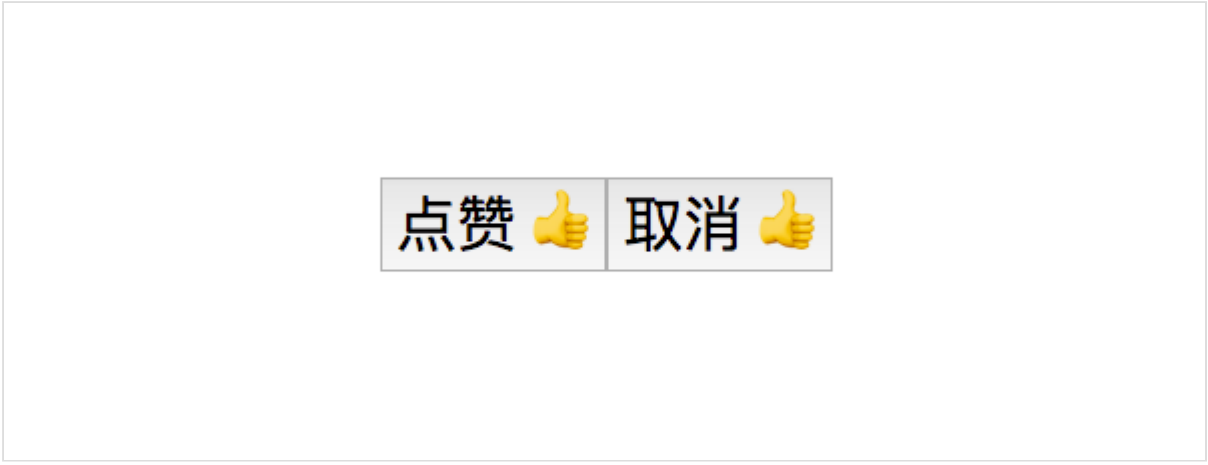
很多课程一上来就给大家如何配置环境、怎么写 React.js 组件。但是本课程还是希望大家对问题的根源有一个更加深入的了解，其实很多的库、框架都是解决类似的问题。只有我们对这些库、框架解决的问题有深入的了解和思考以后，我们才能得心应手地使用它们，并且有新的框架出来也不会太过迷茫——因为其实它们解决都是同一个问题。

这两节课我们来探讨一下是什么样的问题导致了我们需要前端页面进行组件化，前端页面的组件化需要解决什么样的问题。后续课程我们再来看看 React.js 是怎么解决这些问题的。

所以这几节所讲的内容将和 React.js 的内容没有太大的关系，但是如果你能顺利了解这几节的内容，那么后面那些对新手来说很复杂的概念对你来说就是非常自然的事。

## 一个简单的点赞功能

我们会从一个简单的点赞功能讲起。 假设现在我们需要实现一个点赞、取消点赞的功能。




如果你对前端稍微有一点了解，你就顺手拈来：

HTML：

```
<body>
  <div class='wrapper'>
    <button class='like-btn'>
      <span class='like-text'>点赞</span>
      <span>👍</span>
    </button>
  </div>
</body>
```

为了模拟现实当中的实际情况，所以这里特意把这个 `button` 里面的 HTML 结构搞得稍微复杂一些。有了这个 HTML 结构，现在就给它加入一些 JavaScript 的行为：

JavaScript：

```
const button = document.querySelector('.like-btn')
const buttonText = button.querySelector('.like-text')
let isLiked = false
button.addEventListener('click', () => {
  isLiked = !isLiked
  if (isLiked) {
    buttonText.innerHTML = '取消'
  } else {
    buttonText.innerHTML = '点赞'
  }
}, false)
```

功能和实现都很简单，按钮已经可以提供点赞和取消点赞的功能。这时候你的同事跑过来了，说他很喜欢你的按钮，他也想用你写的这个点赞功能。这时候问题就来了，你就会发现这种实现方式很致命：你的同事要把整个 `button` 和里面的结构复制过去，还有整段 JavaScript 代码也要复制过去。这样的实现方式没有任何可复用性。

## 结构复用

现在我们来重新编写这个点赞功能，让它具备一定的可复用。这次我们先写一个类，这个类有 render 方法，这个方法里面直接返回一个表示 HTML 结构的字符串：

```
class LikeButton {
  render () {
    return `
      <button id='like-btn'>
        <span class='like-text'>赞</span>
        <span>👍</span>
      </button>
    `
  }
}
```

然后可以用这个类来构建不同的点赞功能的实例，然后把它们插到页面中。

```
const wrapper = document.querySelector('.wrapper')
const likeButton1 = new LikeButton()
wrapper.innerHTML = likeButton1.render()

const likeButton2 = new LikeButton()
wrapper.innerHTML += likeButton2.render()
```


这里非常暴力地使用了 `innerHTML` ，把两个按钮粗鲁地插入了 `wrapper` 当中。虽然你可能会对这种实现方式非常不满意，但我们还是勉强了实现了结构的复用。我们后面再来优化它。

## 实现简单的组件化

你一定会发现，现在的按钮是死的，你点击它它根本不会有什么反应。因为根本没有往上面添加事件。但是问题来了，`LikeButton` 类里面是虽然说有一个 `button`，但是这玩意根本就是在字符串里面的。你怎么能往一个字符串里面添加事件呢？DOM 事件的 API 只有 DOM 结构才能用。

我们需要 DOM 结构，准确地来说：**我们需要这个点赞功能的 HTML 字符串表示的 DOM 结构。**假设我们现在有一个函数 `createDOMFromString` ，你往这个函数传入 HTML 字符串，但是它会把相应的 DOM 元素返回给你。这个问题就可以解决了。

```
// ::String => ::Document
const createDOMFromString = (domString) => {
  const div = document.createElement('div')
  div.innerHTML = domString
  return div
}
```

先不用管这个函数应该怎么实现，先知道它是干嘛的。拿来用就好，这时候用它来改写一下 `LikeButton` 类：

```
class LikeButton {
  render () {
    this.el = createDOMFromString(`
      <button class='like-button'>
        <span class='like-text'>点赞</span>
        <span>👍</span>
      </button>
    `)
    this.el.addEventListener('click', () => console.log('click'), false)
    return this.el
  }
}
```

现在 `render()` 返回的不是一个 html 字符串了，而是一个由这个 html 字符串所生成的 DOM。在返回 DOM 元素之前会先给这个 DOM 元素上添加事件再返回。

因为现在 `render` 返回的是 DOM 元素，所以不能用 `innerHTML` 暴力地插入 wrapper。而是要用 DOM API 插进去。

```
const wrapper = document.querySelector('.wrapper')

const likeButton1 = new LikeButton()
wrapper.appendChild(likeButton1.render())

const likeButton2 = new LikeButton()
wrapper.appendChild(likeButton2.render())
```

现在你点击这两个按钮，每个按钮都会在控制台打印 `click`，说明事件绑定成功了。但是按钮上的文本还是没有发生改变，只要稍微改动一下 `LikeButton` 的代码就可以完成完整的功能：

```
class LikeButton {
  constructor () {
    this.state = { isLiked: false }
  }

  changeLikeText () {
    const likeText = this.el.querySelector('.like-text')
    this.state.isLiked = !this.state.isLiked
    likeText.innerHTML = this.state.isLiked ? '取消' : '点赞'
  }

  render () {
    this.el = createDOMFromString(`
      <button class='like-button'>
        <span class='like-text'>点赞</span>
        <span>👍</span>
      </button>
```

```
      `)
      this.el.addEventListener('click', this.changeLikeText.bind(this), false)
      return this.el
    }
  }
```

这里的代码稍微长了一些，但是还是很好理解。只不过是在给 `LikeButton` 类添加了构造函数，这个构造函数会给每一个 `LikeButton` 的实例添加一个对象 `state`，`state` 里面保存了每个按钮自己是否点赞的状态。还改写了原来的事件绑定函数：原来只打印 `click`，现在点击的按钮的时候会调用 `changeLikeText` 方法，这个方法会根据 `this.state` 的状态改变点赞按钮的文本。

现在这个组件的可复用性已经很不错了，你的同事们只要实例化一下然后插入到 DOM 里面去就好了。

下一节我们继续优化这个例子，让它更加通用。

> 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：3. 前端组件化（二）：优化 DOM 操作

上一节：1. React.js 简介