React.js 小书

<-- 返回首页

前端组件化（三）：抽象出公共组件类

- 作者：胡子大哈
- 原文链接： http://huziketang.com/books/react/lesson4

为了让代码更灵活，可以写更多的组件，我们把这种模式抽象出来，放到一个 `Component` 类当中：

```
class Component {
  setState (state) {
    const oldEl = this.el
    this.state = state
    this.el = this._renderDOM()
    if (this.onStateChange) this.onStateChange(oldEl, this.el)
  }

  _renderDOM () {
    this.el = createDOMFromString(this.render())
    if (this.onClick) {
      this.el.addEventListener('click', this.onClick.bind(this), false)
    }
    return this.el
  }
}
```

这个是一个组件父类 `Component`，所有的组件都可以继承这个父类来构建。它定义的两个方法，一个是我们已经很熟悉的 `setState`；一个是私有方法 `_renderDOM`。`_renderDOM` 方法会调用 `this.render` 来构建 DOM 元素并且监听 `onClick` 事件。所以，组件子类继承的时候只需要实现一个返回 HTML 字符串的 `render` 方法就可以了。

还有一个额外的 `mount` 的方法，其实就是把组件的 DOM 元素插入页面，并且在 `setState` 的时候更新页面：

```
const mount = (component, wrapper) => {
  wrapper.appendChild(component._renderDOM())
  component.onStateChange = (oldEl, newEl) => {
    wrapper.insertBefore(newEl, oldEl)
    wrapper.removeChild(oldEl)
  }
}
```

这样的话我们重新写点赞组件就会变成：

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

  onClick () {
    this.setState({
      isLiked: !this.state.isLiked
    })
  }

  render () {
    return `
      <button class='like-btn'>
        <span class='like-text'>${this.state.isLiked ? '取消' : '点赞'}</span>
        <span>👍</span>
      </button>
    `
  }
}

mount(new LikeButton(), wrapper)
```

这样还不够好。在实际开发当中，你可能需要给组件传入一些自定义的配置数据。例如说想配置一下点赞按钮的背景颜色，如果我给它传入一个参数，告诉它怎么设置自己的颜色。那么这个按钮的定制性就更强了。所以我们可以给组件类和它的子类都传入一个参数 `props`，作为组件的配置参数。修改 `Component` 的构造函数为：

```
...
  constructor (props = {}) {
    this.props = props
  }
...
```

继承的时候通过 `super(props)` 把 `props` 传给父类，这样就可以通过 `this.props` 获取到配置参数：

```
class LikeButton extends Component {
  constructor (props) {
    super(props)
    this.state = { isLiked: false }
  }

  onClick () {
    this.setState({
      isLiked: !this.state.isLiked
    })
```

```
  }

  render () {
    return `
      <button class='like-btn' style="background-color: ${this.props.bgColor}"
        <span class='like-text'>
          ${this.state.isLiked ? '取消' : '点赞'}
        </span>
        <span>👍</span>
      </button>
    `
  }
}

mount(new LikeButton({ bgColor: 'red' }), wrapper)
```

这里我们稍微修改了一下原有的 `LikeButton` 的 `render` 方法，让它可以根据传入的参数 `this.props.bgColor` 来生成不同的 `style` 属性。这样就可以自由配置组件的颜色了。

只要有了上面那个 `Component` 类和 `mount` 方法加起来不足40行代码就可以做到组件化。如果我们需要写另外一个组件，只需要像上面那样，简单地继承一下 `Component` 类就好了：

```
class RedBlueButton extends Component {
  constructor (props) {
    super(props)
    this.state = {
      color: 'red'
    }
  }

  onClick () {
    this.setState({
      color: 'blue'
    })
  }

  render () {
    return `
      <div style='color: ${this.state.color};'>${this.state.color}</div>
    `
  }
}
```

简单好用，现在可以灵活地组件化页面了。`Component` 完整的代码可以在这里找到 reactjs-in-40。

总结

我们用了很长的篇幅来讲一个简单的点赞的例子，并且在这个过程里面一直在优化编写的方式。最后抽离出来了一个类，可以帮助我们更好的做组件化。在这个过程里面我们学到了什么？

组件化可以帮助我们解决前端结构的复用性问题，整个页面可以由这样的不同的组件组合、嵌套构成。

一个组件有自己的显示形态（上面的 HTML 结构和内容）行为，组件的显示形态和行为可以由数据状态（state）和配置参数（props）共同决定。数据状态和配置参数的改变都会影响到这个组件的显示形态。

当数据变化的时候，组件的显示需要更新。所以如果组件化的模式能提供一种高效的方式自动化地帮助我们更新页面，那也就可以大大地降低我们代码的复杂度，带来更好的可维护性。

好了，课程结束了。你已经学会了怎么使用 React.js 了，因为我们已经写了一个——当然我是在开玩笑，但是上面这个 `Component` 类其实和 React 的 `Component` 使用方式很类似。掌握了这几节的课程，你基本就掌握了基础的 React.js 的概念。

接下来我们开始正式进入主题，开始正式介绍 React.js。你会发现，有了前面的铺垫，下面讲的内容理解起来会简单很多了。

> 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：React.js 基本环境安装

上一节：前端组件化（二）：优化 DOM 操作

4 条评论，4 人参与。

5

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

SprayLee • 7月5日 16:26

厉害了

顶 • 回复 • 分享»

• 6月20日 19:56

第三段代码 似乎少了 super();

顶 • 回复 • 分享»

蒋启钲 • 4月13日 13:50

非常赞

1顶 • 回复 • 分享»

Ahonn_ • 4月2日 20:46

这部分很赞，从普通的例子开始讲如何以及为何组件化。 比起直接开始讲 React 怎么用好很多。

顶 • 回复 • 分享»

友言?