React.js 小书

<-- 返回首页

state vs props

- 作者：胡子大哈
- 原文链接： http://huziketang.com/books/react/lesson12

我们来一个关于 `state` 和 `props` 的总结。

`state` 的主要作用是用于组件保存、控制、修改自己的可变状态。`state` 在组件内部初始化，可以被组件自身修改，而外部不能访问也不能修改。你可以认为 `state` 是一个局部的、只能被组件自身控制的数据源。`state` 中状态可以通过 `this.setState` 方法进行更新，`setState` 会导致组件的重新渲染。

`props` 的主要作用是让使用该组件的父组件可以传入参数来配置该组件。它是外部传进来的配置参数，组件内部无法控制也无法修改。除非外部组件主动传入新的 `props`，否则组件的 `props` 永远保持不变。

`state` 和 `props` 有着千丝万缕的关系。它们都可以决定组件的行为和显示形态。一个组件的 `state` 中的数据可以通过 `props` 传给子组件，一个组件可以使用外部传入的 `props` 来初始化自己的 `state`。但是它们的职责其实非常明晰分明：**`state` 是让组件控制自己的状态，`props` 是让外部对组件自己进行配置**。

如果你觉得还是搞不清 `state` 和 `props` 的使用场景，那么请记住一个简单的规则：尽量少地用 `state`，尽量多地用 `props`。

没有 `state` 的组件叫无状态组件（stateless component），设置了 state 的叫做有状态组件（stateful component）。因为状态会带来管理的复杂性，我们尽量多地写无状态组件，尽量少地写有状态的组件。这样会降低代码维护的难度，也会在一定程度上增强组件的可复用性。前端应用状态管理是一个复杂的问题，我们后续会继续讨论。

React.js 非常鼓励无状态组件，在 0.14 版本引入了函数式组件——一种定义不能使用 `state` 组件，例如一个原来这样写的组件：

```
class HelloWorld extends Component {
  constructor() {
    super()
  }

  sayHi () {
    alert('Hello World')
```

```
  }

  render () {
    return (
      <div onClick={this.sayHi.bind(this)}>Hello World</div>
    )
  }
}
```

用函数式组件的编写方式就是：

```
const HelloWorld = (props) => {
  const sayHi = (event) => alert('Hello World')
  return (
    <div onClick={sayHi}>Hello World</div>
  )
}
```

以前一个组件是通过继承 `Component` 来构建，一个子类就是一个组件。而用函数式的组件编写方式是一个函数就是一个组件，你可以和以前一样通过 `<HellWorld />` 使用该组件。不同的是，函数式组件只能接受 `props` 而无法像跟类组件一样可以在 `constructor` 里面初始化 `state`。你可以理解函数式组件就是一种只能接受 `props` 和提供 `render` 方法的类组件。

但本书全书不采用这种函数式的方式来编写组件，统一通过继承 `Component` 来构建组件。

> 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：渲染列表数据

上一节：配置组件的 props

2 条评论，2 人参与。

0

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

念旧。 • 6月29日 23:58

你可以理解函数式组件就是一种只能接受 props 和提供 render 方法的类组件 ,函数组件能用render方法??

顶 • 回复 • 分享»

亿丰克微 • 4月24日 16:04

如果 将 this.props.data 赋值给 this.state.data 然后在父组件 更新传入的 data 组件中的 this.state.data 仍然不变

顶 • 回复 • 分享»

友言?