无状态组件(Stateless Component) 与高阶组件

作者 jacobbubu (/u/8ca9803d6c8a) + 关注
2016.04.13 12:07* 字数 1170 阅读 2888 评论 2 喜欢 16
(/u/8ca9803d6c8a)

无状态组件(Stateless Component) 是 React 0.14 之后推出的，大大增强了编写 React 组件的方便性，也提升了整体的渲染性能。

无状态组件 (Stateless Component)

```
function HelloComponent(props, /* context */) {
  return <div>Hello {props.name}</div>
}
ReactDOM.render(<HelloComponent name="Sebastian" />, mountNode)
```

`HelloComponent` 第一个参数是 `props` ，第二个是 `context` 。最后一句也可以这么写：

```
ReactDOM.render(HelloComponent{ name: "Sebastian" }, mountNode)
```

可以看到，原本需要写“类”定义（ `React.createClass` 或者 `class YourComponent extends React.Component` ）来创建自己组件的定义，现在被精简成了只写一个 `render` 函数。更值得一提的是，由于仅仅是一个无状态函数，React 在渲染的时候也省掉了将“组件类” 实例化的过程。

结合 ES6 的解构赋值，可以让代码更精简。例如下面这个 Input 组件：

```
function Input({ label, name, value, ...props }, { defaultTheme }) {
  const { theme, autoFocus, ...rootProps } = props
  return (
    <label
      htmlFor={name}
      children={label || defaultLabel}
      {...rootProps}
    >
    <input
      name={name}
      type="text"
      value={value || ''}
      theme={theme || defaultTheme}
      {...props}
    />
  )}
Input.contextTypes = {defaultTheme: React.PropTypes.object};
```

这个 Input 组件（仅仅是示例）直接实现了 label/inputText 的组合：

- `defaultTheme` 是从 Context 中解构出来的，如果 `props` 没有设定 `theme` ，就将用 `defaultTheme` 替代。
- `autoFocus` 需要被传递到底层的 inputText 而不能同时遗留给 label，因此会先通过 `{ theme, autoFocus, ...rootProps } = props` 拿出来。

无状态组件用来实现 Server 端渲染也很方便，只要避免去直接访问各种 DOM 方法。

无状态组件与组件的生命周期方法

我们可以看到，无状态组件就剩了一个 `render` 方法，因此也就没有没法实现组件的生命周期方法，例如 `componentDidMount`，`componentWillUnmount` 等。那么如果需要让我们的 `Input` 组件能够响应窗口大小的变化，那么该如何实现呢？这其实还是要引入“有状态的组件”，只不过这个“有状态的组件”可以不仅仅为 "Input" 组件服务。

```
const ExecutionEnvironment = require('react/lib/ExecutionEnvironment')
let viewport = { width: 1366, height: 768 }; // Default size for server-side renderi
ng

function handleWindowResize() {
  if (viewport.width !== window.innerWidth || viewport.height !== window.innerHeight
) {
    viewport = { width: window.innerWidth, height: window.innerHeight }
  }
}

function withViewport(ComposedComponent) {
  return class Viewport extends React.Component {
    state = {
      // Server 端渲染和单元测试的时候可未必有 DOM 存在
      viewport: ExecutionEnvironment.canUseDOM ?
        { width: window.innerWidth, height: window.innerHeight } : viewport
    }
    componentDidMount() {
      // Server 端渲染是不会执行到 `componentDidMount` 的，只会执行到 `componentWillMoun
t`
      window.addEventListener('resize', handleWindowResize)
      window.addEventListener('orientationchange', handleWindowResize)
    }
    componentWillUnmount() {
      window.removeEventListener('resize', handleWindowResize)
      window.removeEventListener('orientationchange', handleWindowResize)
    }
    render() {
      return <ComposedComponent {...this.props} viewport={this.state.viewport}/>
    }
  }
}
```

专业的实现参看 https://github.com/kriasoft/react-decorators (https://github.com/kriasoft/react-decorators)

那么，下面我们就可以创建出一个有机会响应窗口大小变化的 Input 组件：

```
const SizeableInput = withViewport(Input)
ReactDOM.render(<SizeableInput name="username" label="Username" {...props} />, mount
Node)
```

`withVieowrt` 作为一个 "高阶组件" (http://www.jianshu.com/p/4780d82e874a) 可不仅仅是为了 Input 服务的。它可以为你需要的任何组件添加上 `viewport` 属性，当窗口大小变化时，触发重绘。

如果你用过 Redux，那么应该也熟悉 "connect decorator" 的用法。"connect decorator" 也是一个高阶组件，因此，你可以继续来“拼凑”：

```
const UserNameInput = connect(
  state => ({ value: state.username })
)(SizeableInput)
```

高阶组件的存在有两个好处：

- 当写着写着无状态组件的时候，有一天忽然发现需要状态处理了，那么无需彻底返工:)
- 往往我们需要状态的时候，这个需求是可以重用的，例如上面的 `withViewport` ，今后可以用来给其他组件（无论是否是无状态组件）添加 `viewport` 属性。

高阶组件加无状态组件，则大大增强了整个代码的可测试性和可维护性。同时不断“诱使”我们写出组合性更好的代码。

无状态组件不支持 "ref"

有一点遗憾的是无状态组件不支持 "ref"。原理很简单，因为在 React 调用到无状态组件的方法之前，是没有一个实例化的过程的，因此也就没有所谓的 "ref"。

`ref` 和 `findDOMNode` 这个组合，实际上是打破了父子组件之间仅仅通过 `props` 来传递状态的约定，是危险且肮脏，需要避免。

无状态组件尚不支持 babel-plugin-react-transform 的 Hot Module Replacement

如果你是用 Webpack 以及 HMR，用 babel-plugin-react-transform (https://github.com/gaearon/babel-plugin-react-transform) 来做 jsx 转换等，那么当你在编辑器中修改无状态组件的源代码的时候，HMR 并不会在浏览器中自动载入修改后的代码。具体问题跟踪请参 https://github.com/gaearon/babel-plugin-react-transform/issues/57 (https://github.com/gaearon/babel-plugin-react-transform/issues/57) 。