React组件性能优化

作者 hepeguo (/u/922546df4ea0) + 关注
2016.07.27 20:42 字数 1452 阅读 734 评论 0 喜欢 5
(/u/922546df4ea0)

> React: 一个用于构建用户界面的JAVASCRIPT库.

React仅仅专注于UI层；它使用虚拟DOM技术，以保证它UI的高速渲染；它使用单向数据流，因此它数据绑定更加简单；那么它内部是如何保持简单高效的UI渲染呢？

React不直接操作DOM，它在内存中维护一个快速响应的DOM描述，render方法返回一个DOM的描述，React能够计算出两个DOM描述的差异，然后更新浏览器中的DOM。

就是说React在接收到props或者state更新时，React就会通过前面的方式更新UI。就算重新使用 `ReactDOM.render(<Component />, mountNode)` ，它也只是当作props更新，而不是重新挂载整个组件。所以React整个UI渲染是比较快的。

但是，这里面有几个问题

1. 如果更新的props和旧的一样，这个时候很明显UI不会变化，但是React还是要进行虚拟DOM的diff，这个diff就是多余的性能损耗，而且在DOM结构比较复杂的情况，整个diff会花费较长的时间。

2. 既然React总是要进行虚拟DOM的diff，那么它的diff规则是什么？怎么利用？

PureRenderMixin

针对第一个问题React给我们提供了 PureRenderMixin。
如果React组件是纯函数的，就是给组件相同的props和state组件就会展现同样的UI，可以使用这个Minxin来优化React组件的性能。

```
var PureRenderMixin = require('react-addons-pure-render-mixin');
React.createClass({
      mixins: [PureRenderMixin],

      render: function() {
        return <div className={this.props.className}>foo</div>;
      }
});
```

ES6中的用法是

```
import PureRenderMixin from 'react-addons-pure-render-mixin';
class FooComponent extends React.Component {
      constructor(props) {
        super(props);
        this.shouldComponentUpdate = PureRenderMixin.shouldComponentUpdate.bind(this
);
      }

      render() {
        return <div className={this.props.className}>foo</div>;
      }
}
```

PureRenderMixin的原理就是它实现了shouldComponentUpdate，在shouldComponentUpdate内它比较当前的props、state和接下来的props、state，当两者相等的时候返回false，这样组件就不会进行虚拟DOM的diff。

这里需要注意：
PureRenderMixin内进行的仅仅是浅比较对象。如果对象包含了复杂的数据结构，深层次的差异可能会产生误判。仅用于拥有简单props和state的组件。

shouldComponentUpdate

React虽然提供简单的PureRenderMixin来提升性能，但是如果有更特殊的需求时怎么办？如果组件有复杂的props和state怎么办？这个时候就可使用shouldComponentUpdate来进行更加定制化的性能优化。

```
boolean shouldComponentUpdate(object nextProps, object nextState) {
    return nexprops.id !== this.props.id;
}
```

在React组件需要更新之前就会调用这个方法，如果这个方法返回false，则组件不更新；如果返回true，则组件更新。在这个方法内部可以通过nextProps和当前props，nextState和当前state的对比决定组件要不要更新。

如果对比的数据结构比较复杂，层次较深，对比的过程也是会有较大性能消耗，又可能得不偿失。
这个时候**immutable.js (https://facebook.github.io/immutable-js/)**就要登场了，也是fb出品，有人说这个框架的意义不亚于React，但是React光芒太强。它能解决复杂数据在deepClone和对比过程中性能损耗。

注意:shouldComponentUpdate在初始化渲染的时候不会调用，但是在使用forceUpdate方法强制更新的时候也不会调用。

render

PureRenderMixin和shouldComponentUpdate的关注点是UI需不需要更新，而render则更多关注虚拟DOM的diff规则了，如何让diff结果最小化、过程最简化是render内优化的关注点。

React在进行虚拟DOM diff的时候假设：

1、拥有相同类的两个组件将会生成相似的树形结构，拥有不同类的两个组件将会生成不同的树形结构。
2、可以为元素提供一个唯一的标志，该元素在不同的渲染过程中保持不变。

DOM结构

```
renderA: <div />
renderB: <span />
=> [removeNode <div />], [insertNode <span />
```

DOM属性

```
renderA: <div id="before" />
renderB: <div id="after" />
=> [replaceAttribute id "after"]
```

之前插入DOM

```
renderA: <div><span>first</span></div>
renderB: <div><span>second</span><span>first</span></div>
=> [replaceAttribute textContent 'second'], [insertNode <span>first</span>]
```

之前插入DOM，有key的情况

```
renderA: <div><span key="first">first</span></div>
renderB: <div><span key="second">second</span><span key="first">first</span></div>
=> [insertNode <span>second</span>]
```

由于依赖于两个预判条件，如果这两个条件都没有满足，性能将会大打折扣。

1、diff算法将不会尝试匹配不同组件类的子树。如果发现正在使用的两个组件类输出的DOM 结构非常相似，你可以把这两个组件类改成一个组件类。

2、如果没有提供稳定的key（例如通过 Math.random() 生成），所有子树将会在每次数据更新中重新渲染。

总结

使用PureRenderMixin、shouldComponentUpdate来避免不必要的虚拟DOM diff，在render内部优化虚拟DOM的diff速度，以及让diff结果最小化。

使用immutable.js解决复杂数据diff、clone等问题。

参考

immutable.js (https://facebook.github.io/immutable-js/)

reconciliation (https://facebook.github.io/react/docs/reconciliation.html)

pure-render-mixin (https://facebook.github.io/react/docs/pure-render-mixin.html)

前端 (/nb/5320656) 举报文章

hepeguo (/u/922546df4ea0) + 关注
写了 10783 字，被 6 人关注，获得了 16 个喜欢
(/u/922546df4ea0)

如果觉得我的文章对您有用，请随意打赏。您的支持将鼓励我继续创作！

赞赏支持

喜欢 (/sign_in) 5

更多分享
(http://cwb.assets.jianshu.io/notes/images/49583

(/sign_in)后发表评论

评论

智慧如你，不想发表一点想法 (/sign_in)咩~

被以下专题收入，发现更多相似内容

首页投稿 前端那些事儿 程序员 React.j... React.js