### 2).比较两棵虚拟DOM树的差异

在用JS对象表示DOM结构后，当页面状态发生变化而需要操作DOM时，我们可以先通过虚拟DOM计算出对真实DOM的最小修改量，然后再修改真实DOM结构(因为真实DOM的操作代价太大)。

。针对前端的具体情况：我们很少跨级别的修改DOM节点，通常是修改节点的属性、调整子节点的顺序、添加子节点等。因此，我们只需要对同级别节点进行比较，避免了diff算法的复杂性。对同级别节点进行比较的常用方法是深度优先遍历：

<2>如何记录节点之间的差异

由于我们对DOM树采取的是同级比较，因此节点之间的差异可以归结为4种类型：

修改节点属性, 用PROPS表示修改节点文本内容, 用TEXT表示替换原有节点, 用REPLACE表示调整子节点，包括移动、删除等，用REORDER表示

在深度优先遍历的过程中，每个节点都有一个编号，如果对应的节点有变化，只需要把相应变化的类别记录下来即可。下面是具体实现：

注意：虚拟DOM只是实现MVVM的一种方案，或者说是视图更新的一种策略。没有虚拟DOM比MVVM更好一说。

在Web开发中，需要将数据的变化实时反映到UI上，这时就需要对DOM进行操作，但是复杂或频繁的DOM操作通常是性能瓶颈产生的原因，为此，React引入了虚拟DOM（Virtual DOM）的机制。

1. 什么是虚拟DOM？
2. 虚拟DOM VS 直接操作原生DOM？
3. 虚拟DOM VS MVVM？
4. 对React虚拟DOM的误解？

## **一、什么是虚拟DOM？**

在React中，render执行的结果得到的并不是真正的DOM节点，结果仅仅是轻量级的JavaScript对象，我们称之为virtual DOM。

虚拟DOM是React的一大亮点，具有batching(批处理)和高效的Diff算法。这让我们可以无需担心性能问题而”毫无顾忌”的随时“刷新”整个页面，由虚拟 DOM来确保只对界面上真正变化的部分进行实际的DOM操作。在实际开发中基本无需关心虚拟DOM是如何运作的，但是理解其运行机制不仅有助于更好的理解React组件的生命周期，而且对于进一步优化 React程序也会有很大帮助。

## **二、虚拟DOM VS 直接操作原生DOM？**

如果没有 Virtual DOM，简单来说就是直接重置 innerHTML。这样操作，在一个大型列表所有数据都变了的情况下，还算是合理，但是，当只有一行数据发生变化时，它也需要重置整个 innerHTML，这时候显然就造成了大量浪费。

比较innerHTML 和Virtual DOM 的重绘过程如下：

innerHTML: render html string + 重新创建所有 DOM 元素

Virtual DOM: render Virtual DOM + diff + 必要的 DOM 更新

和 DOM 操作比起来，js 计算是非常便宜的。Virtual DOM render + diff 显然比渲染 html 字符串要慢，但是，它依然是纯 js 层面的计算，比起后面的 DOM 操作来说，依然便宜了太多。当然，曾有人做过验证说React的性能不如直接操作真实DOM，代码如下：

functionRaw(){

vardata=\_buildData(),

html="";

**...**

**for**(vari=0;i<data.length;i++){

varrender=template;

render=render.replace("{{className}}","");

render=render.replace("{{label}}",data[i].label);

html+=render;

}

**...**

container.innerHTML=html;

**...**

}

该测试用例中虽然构造了一个包含1000个Tag的String，并把它添加到DOM树中，但是只做了一次DOM操作。然而，在实际开发过程中，这1000个元素更新可能分布在20个逻辑块中，每个逻辑块中包含50个元素，当页面需要更新时，都会引起DOM树的更新，上述代码就近似变成了如下格式：

functionRaw(){

vardata=\_buildData(),

html="";

...

for(vari=0;i<data.length;i++){

varrender=template;

render=render.replace("{{className}}","");

render=render.replace("{{label}}",data[i].label);

html+=render;

if(!(i%50)){

container.innerHTML=html;

}

}

...

}

这样来看，React的性能就远胜于原生DOM操作了。

而且，DOM 完全不属于Javascript (也不在Javascript 引擎中存在).。Javascript 其实是一个非常独立的引擎，DOM其实是浏览器引出的一组让Javascript操作HTML文档的API而已。在即时编译的时代，调用DOM的开销是很大的。而Virtual DOM的执行完全都在Javascript 引擎中，完全不会有这个开销。

React.js 相对于直接操作原生DOM有很大的性能优势， 很大程度上都要归功于virtual DOM的batching 和diff。batching把所有的DOM操作搜集起来，一次性提交给真实的DOM。diff算法时间复杂度也从 ***[标准的的Diff算法](http://grfia.dlsi.ua.es/ml/algorithms/references/editsurvey_bille.pdf)*** 的O(n^3)降到了O(n)。这里留到下一次博客单独讲。

## **四、对React虚拟DOM的误解？**

React 从来没有说过 “React 比原生操作 DOM 快”。React给我们的保证是，在不需要手动优化的情况下，它依然可以给我们提供过得去的性能。

React掩盖了底层的 DOM 操作，可以用更声明式的方式来描述我们目的，从而让代码更容易维护。下面还是借鉴了知乎上的回答：没有任何框架可以比纯手动的优化 DOM 操作更快，因为框架的 DOM 操作层需要应对任何上层 API 可能产生的操作，它的实现必须是普适的。针对任何一个 benchmark，我都可以写出比任何框架更快的手动优化，但是那有什么意义呢？在构建一个实际应用的时候，你难道为每一个地方都去做手动优化吗？出于可维护性的考虑，这显然不可能。

<http://blog.csdn.net/qian_xiaoqian/article/details/53068860>

<http://blog.csdn.net/qq_26943485/article/details/56676803>

<https://www.zhihu.com/question/29504639?sort=created>

<http://blog.csdn.net/jiangbo_phd/article/details/51758148>

http://kb.cnblogs.com/page/129756/