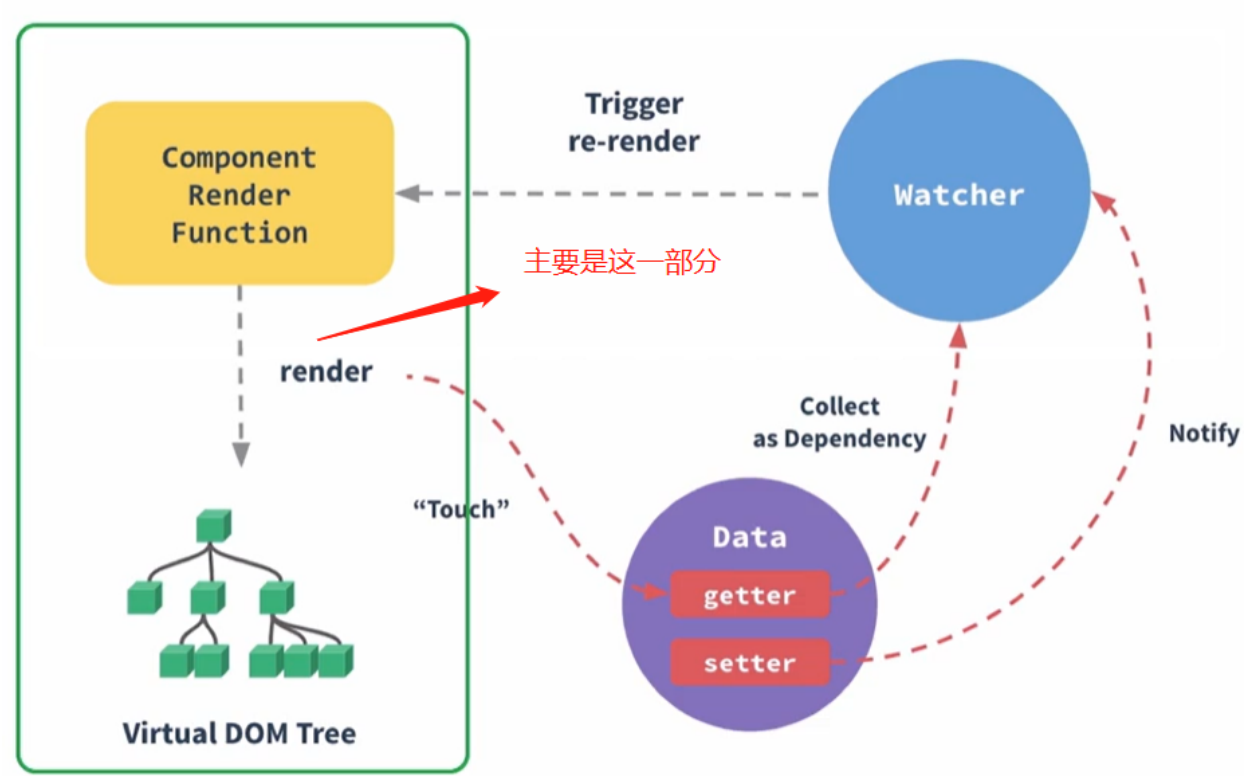
课程目标



上两章主要讲的是依赖收集和依赖通知更新，以及使用异步更新队列解决模板多次渲染的问题，这一节主要讲进入渲染页面的部分，vue根据前面的变更通知，生成一个新的Virtual Dom树,然后将新的Virtual Dom和之前的Virtual Dom树进行diffpatch操作，然后再进行页面渲染。

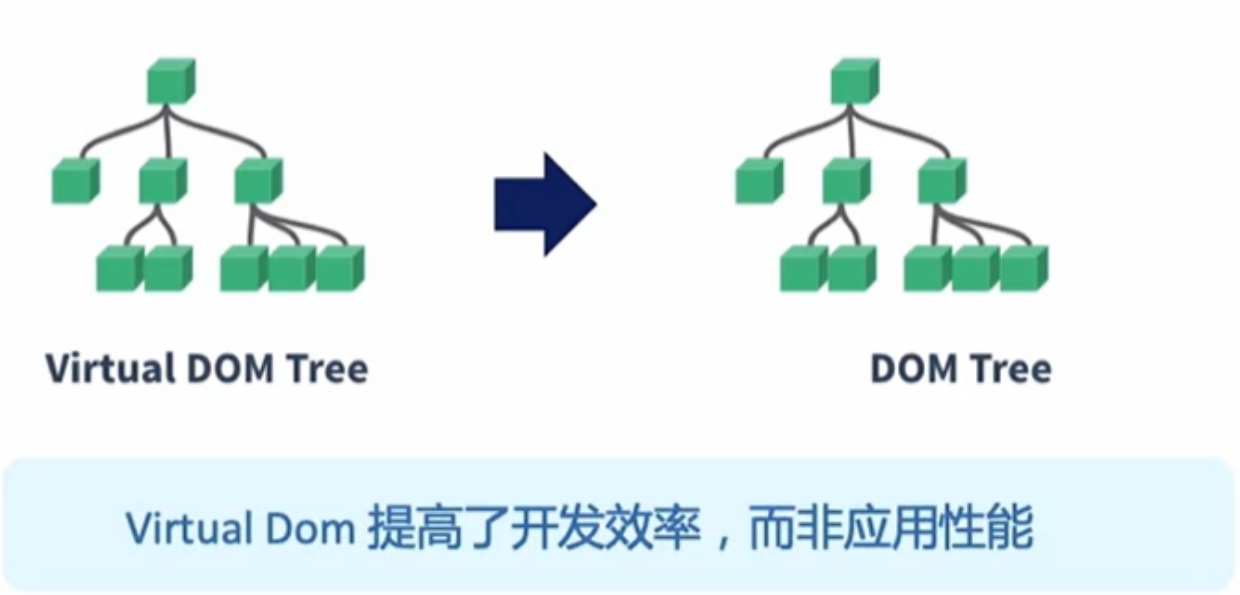
Virtual Dom

virtual Dom称之为虚拟dom，用普通的js来描述DOM结构，因为不是真实的DOM，所以称之为虚拟dom。

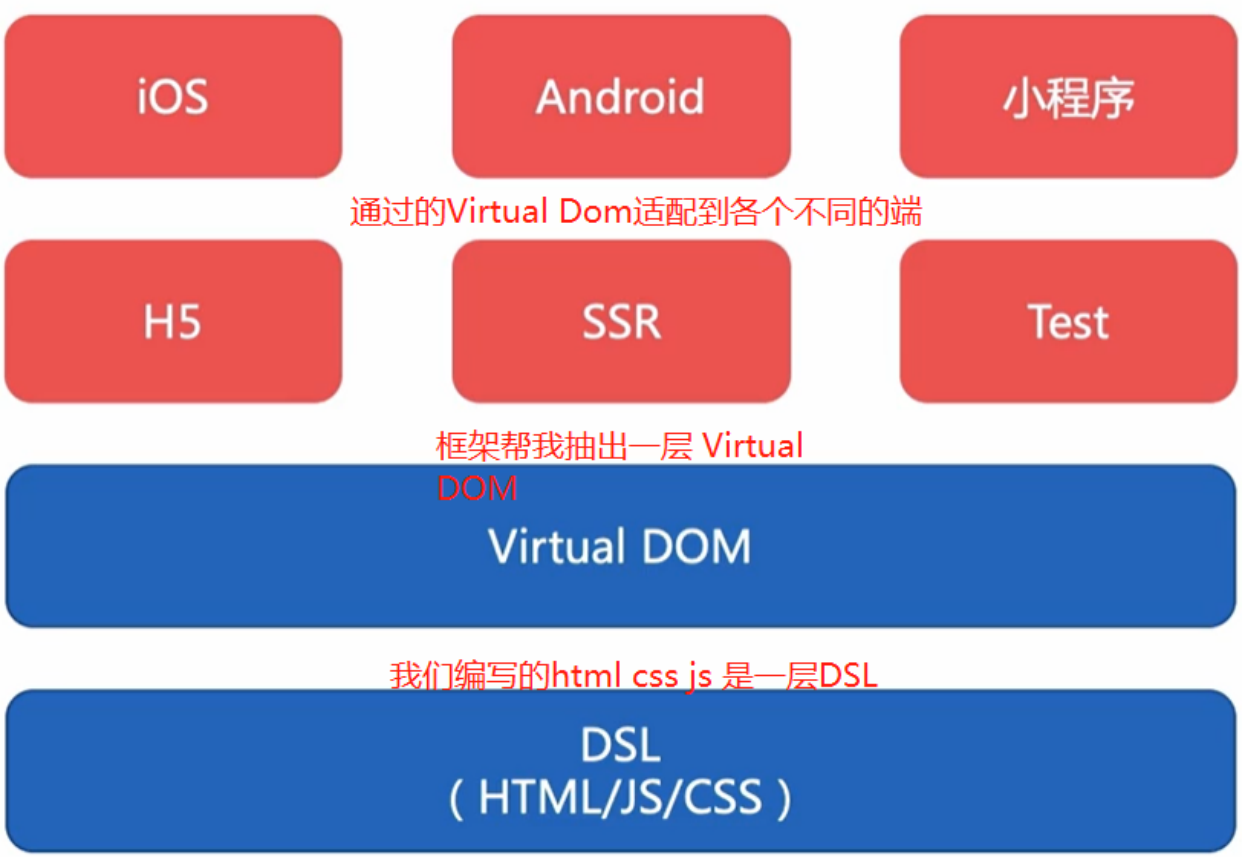
为什么要使用Virtual Dom呢?

操作Dom是很耗性能的一件事情，我们可以考虑通过js对象来模拟DOM对象，毕竟操作JS对象比操作DOM省时的多。

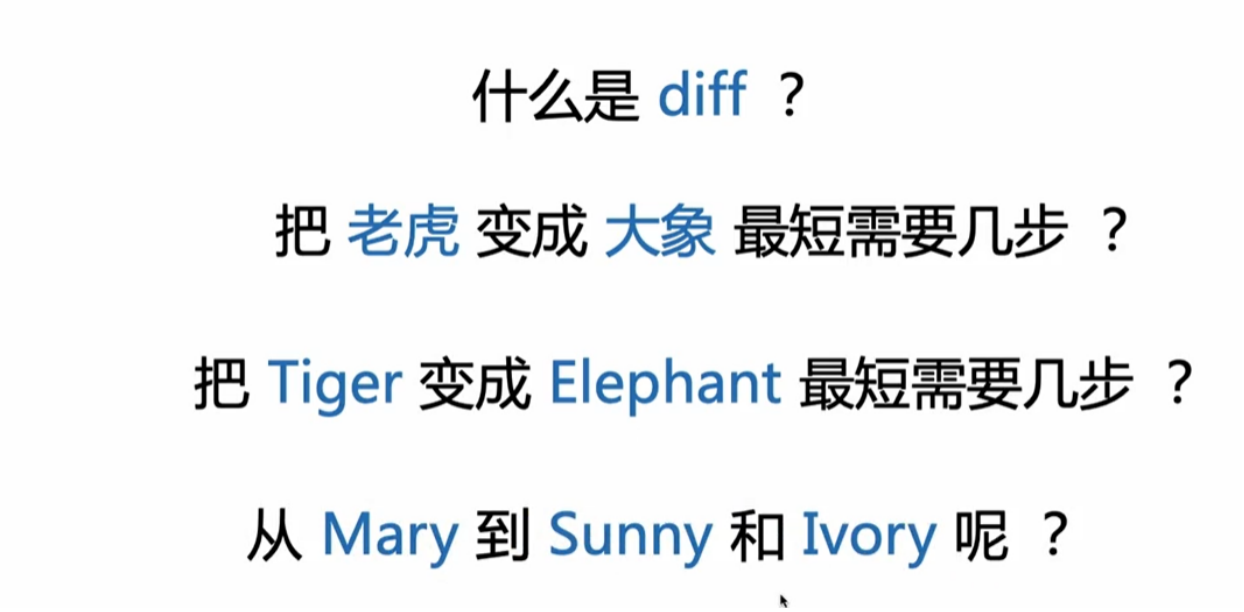
其实这是一种错误的说法，操作js对象只是省时，只是相对提高了开发效率，而非应用性能提升。



Virtual DOM 是分层思想的体现。



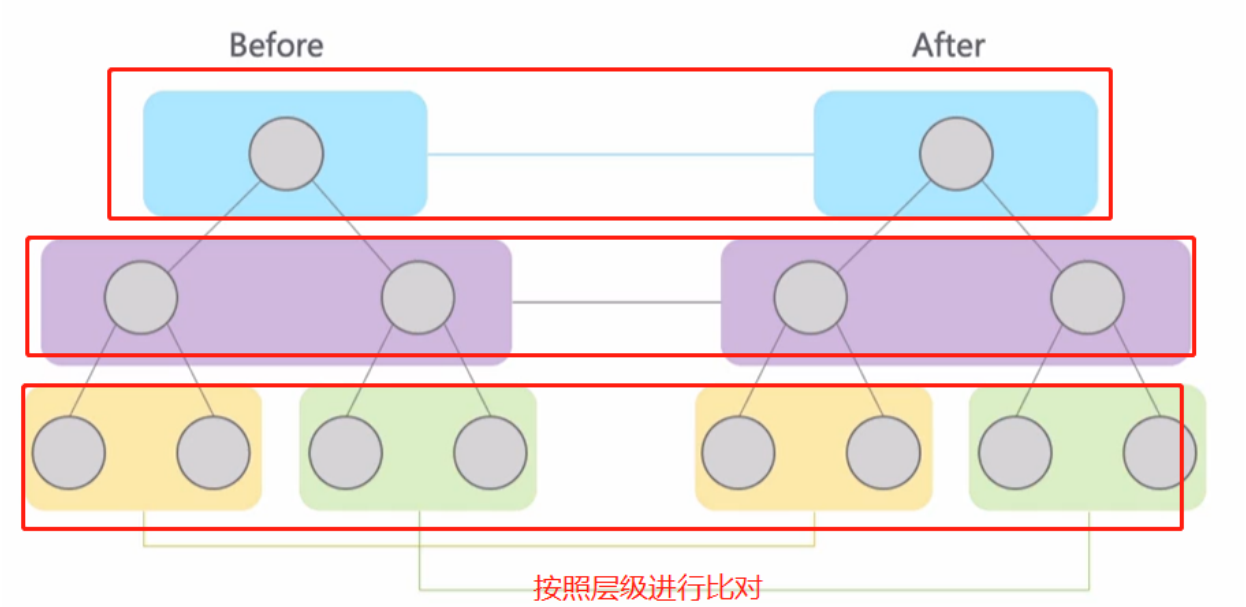
Diff&Patch



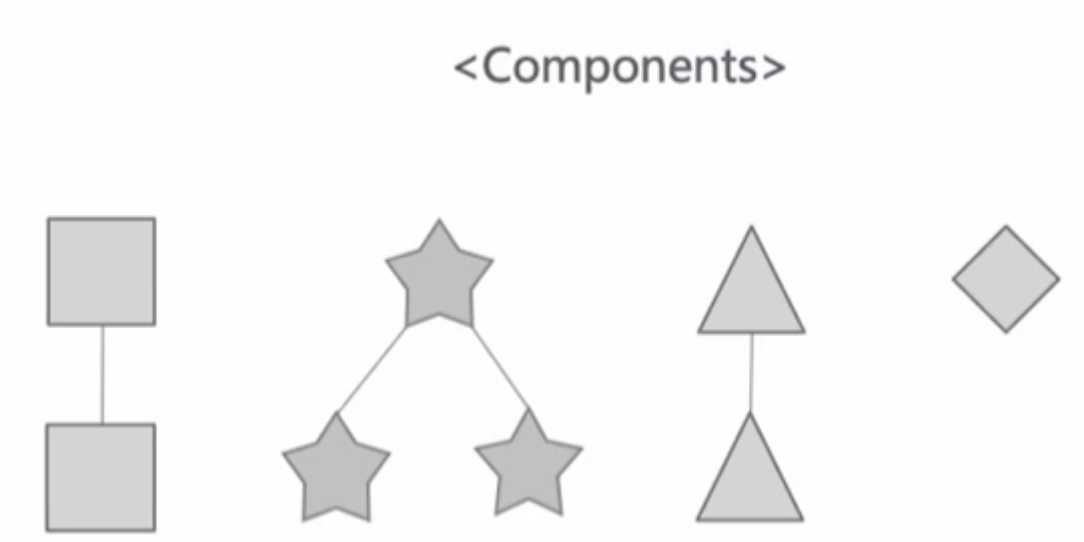
在这里我们可以看出diff是字符串变更的比较和比对。不仅如此 diff算法，还用与生物学RNA的序列比对，也是使用了diff算法。下面我们来说一下 vue当中的diff策略。

Diff策略

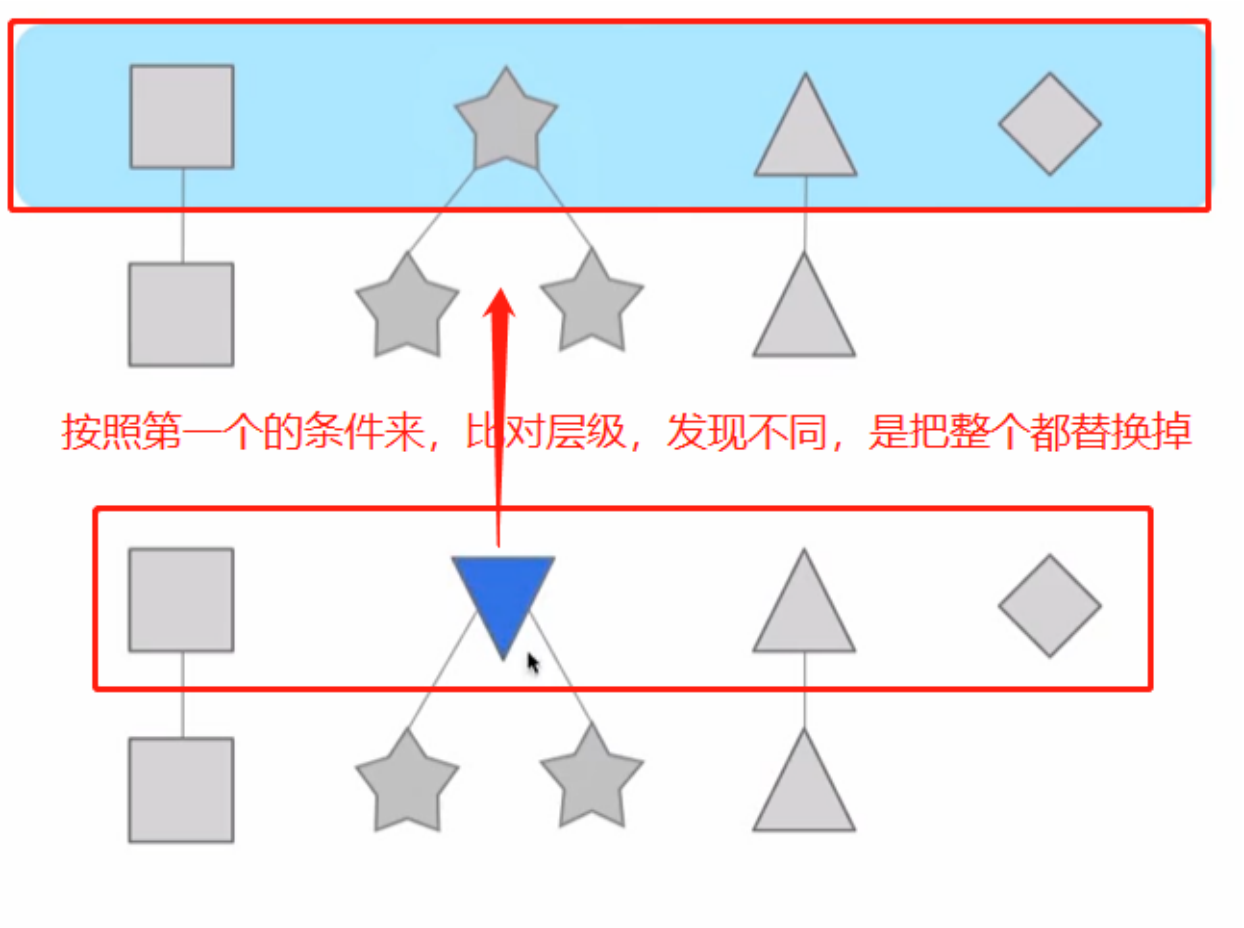
1. 按照tree层级diff(level by level 按照层级来的diff差异)



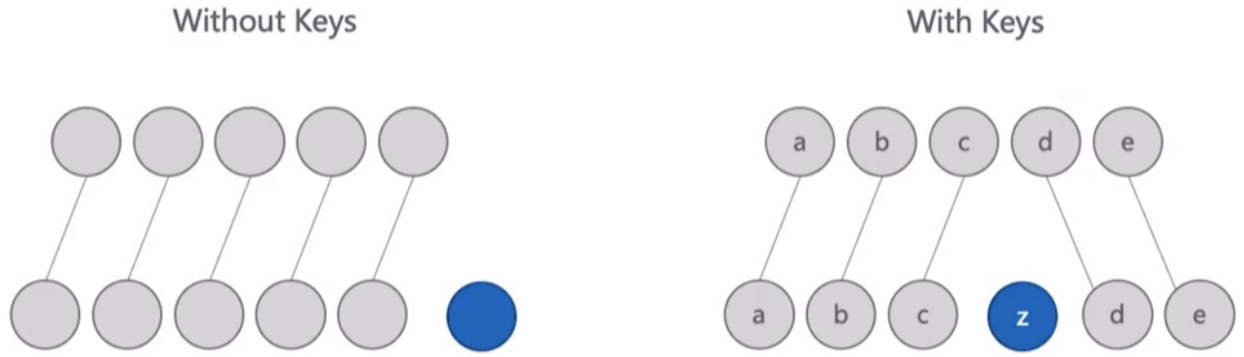
2. 按照类型进行diff(无论 virtual dom节点对应是原生的dom节点还是vue组件。不同类型的节点指数往往差异明显，因此对不同类型的节点的指数进行diff的成本将比较高昂。)



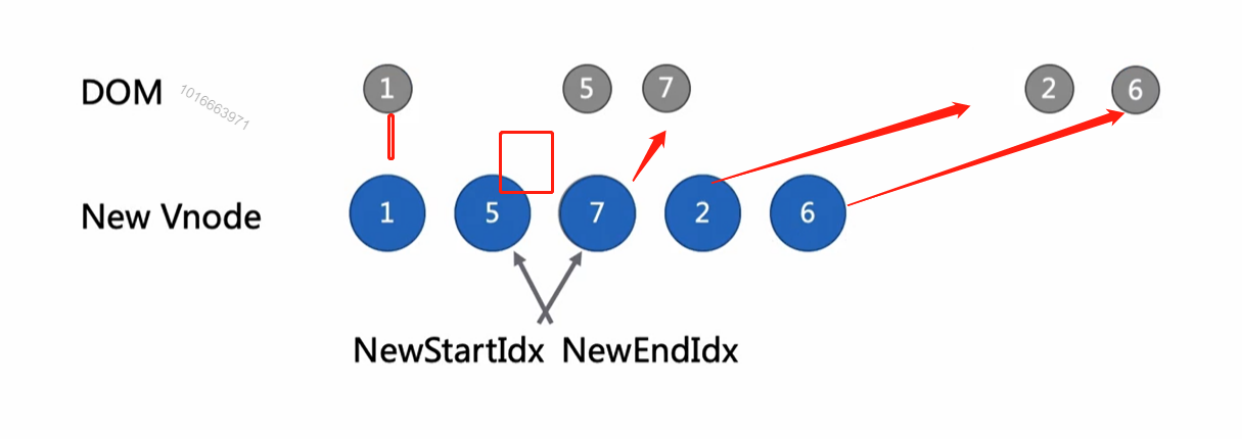
这个按照类型来diff成本很高，还是按照第一个条件按照层级来diff。



3.列表diff(在列表中设立key，提高diff算法的效率。)



真实的diff算法比对，通过的老的dom和虚拟dom进行比对。最终在页面渲染的dom就是虚拟dom排列的dom。



key

设置key 算法的复杂度 O(n)
不设置key 算法的复杂度为 最好情况 O(n) 最坏情况 O(n2)
设置key大大降低了了算法的复杂度。