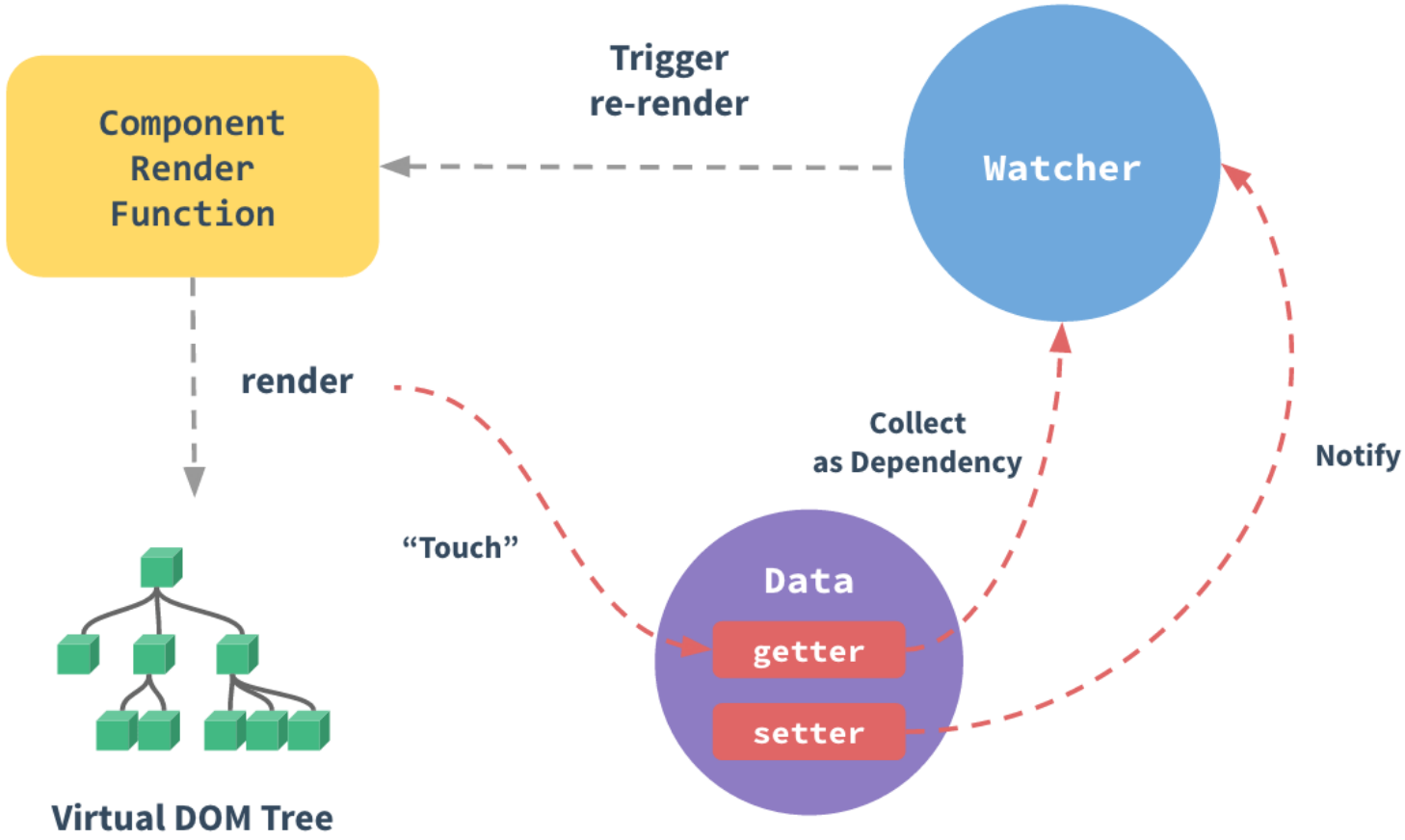
真传X

IT前沿技术在线大学

www.zhenchuanx.com

Vuejs 底层原理



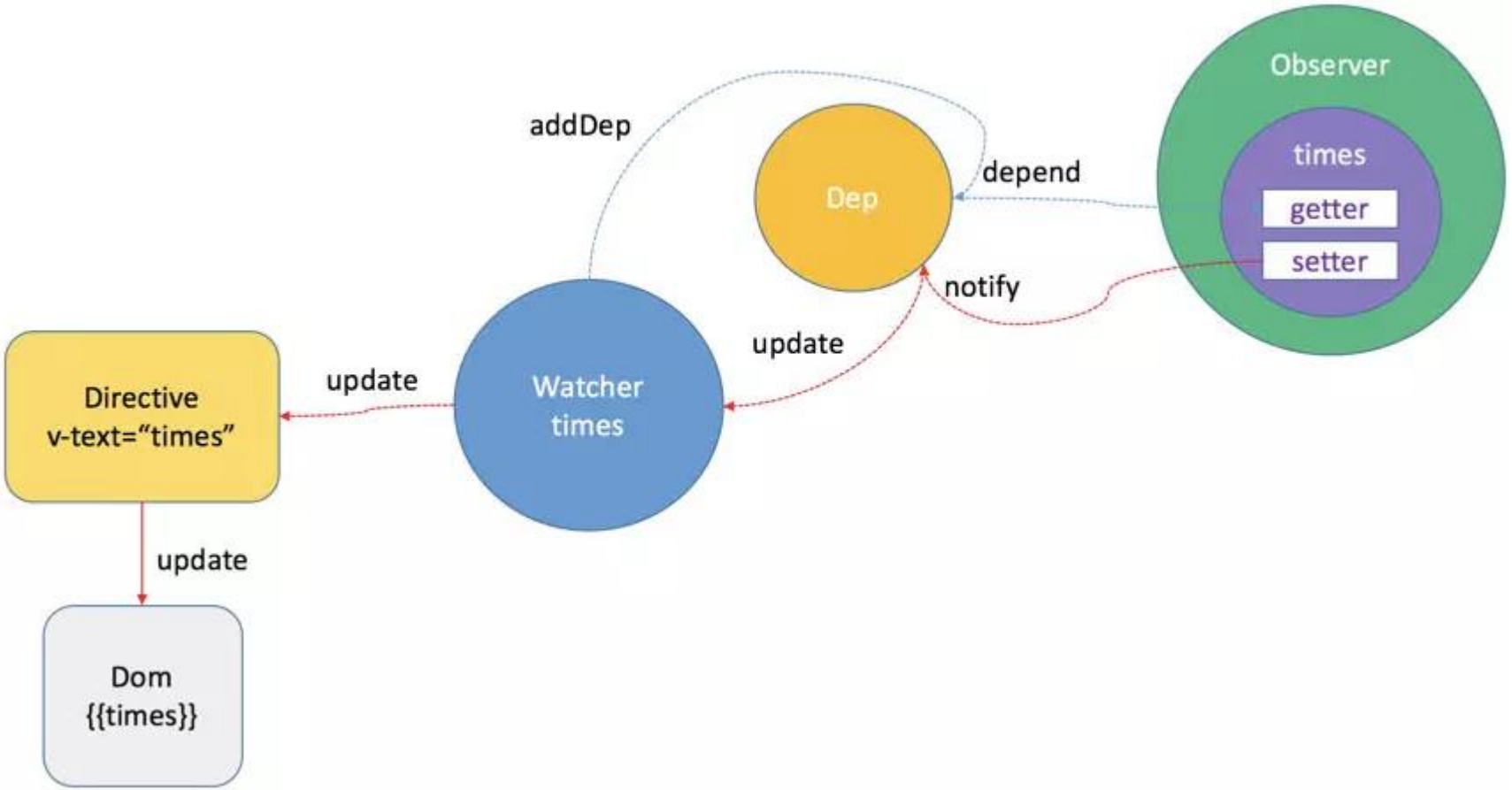
Observer：能够对数据对象的所有属性进行监听，如有变动可拿到最新值并通知订阅者

Compile：对每个元素节点的指令进行扫描和解析，根据指令模板替换数据，以及绑定相应的更新函数

Watcher：作为连接Observer和Compile的桥梁，能够订阅并收到每个属性变动的通知，执行指令绑定的相应回调函数，从而更新视图

简单的 vue demo

https://github.com/FE-star/2019.03/blob/master/vue-demo.html



https://segmentfault.com/a/1190000016434836

Observer：

Observer的核心是通过Obeject.defineProperty()来监听数据的变动，这个函数内部可以定义setter和getter，每当数据发生变化，就会触发setter。

这时候Observer就要通知订阅者，订阅者就是Watcher。

Compile:

Compile主要做的事情是解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图。

Watcher:

Watcher订阅者作为Observer和Compile之间通信的桥梁，主要做的事情是：

1. 在自身实例化时往属性订阅器(dep)里面添加自己

2. 自身必须有一个update()方法

3.待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调

为什么用 virtual dom?

操作 dom 耗时大，可以通过virtual dom达到以下优化

- 在内存中构建 js dom(virtaul dom) – 内存操作
- 通过 js dom 比较，计算出新旧js dom 最少变化指令(patch)
- 运行指令，更新真实dom

传统的 diff 算法

计算treeA转换成treeB的最少操作

算法原理：两棵树互相迭代匹配

传统解法并不友好，算法复杂度：O(n^3)

https://grfia.dlsi.ua.es/ml/algorithms/references/editsurvey_bille.pdf

React 是怎么做的

https://blog.csdn.net/qq_26708777/article/details/78107577

https://www.jianshu.com/p/3ba0822018cf

Diff 策略优化思路：

全局匹配 - > 局部匹配

最优操作指令-> 相对优操作指令

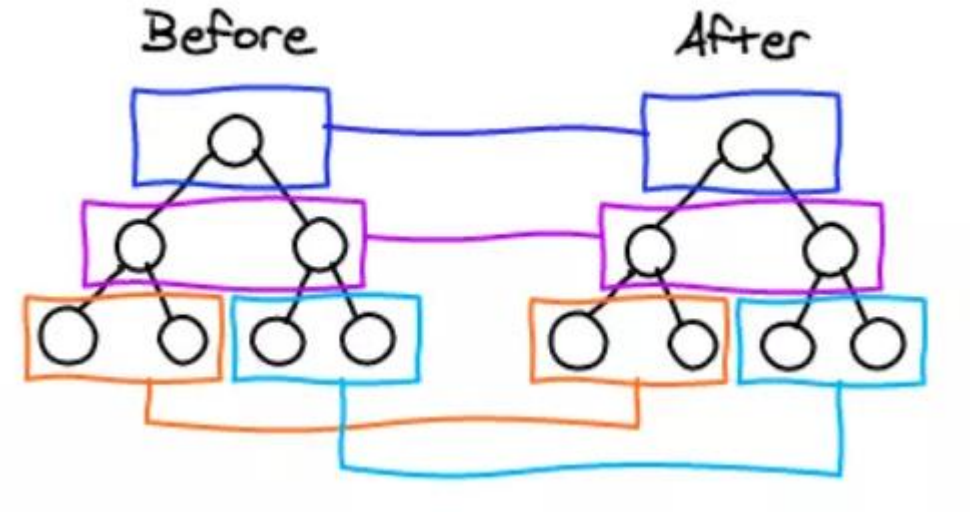
Diff 策略优化思路：

1. tree diff

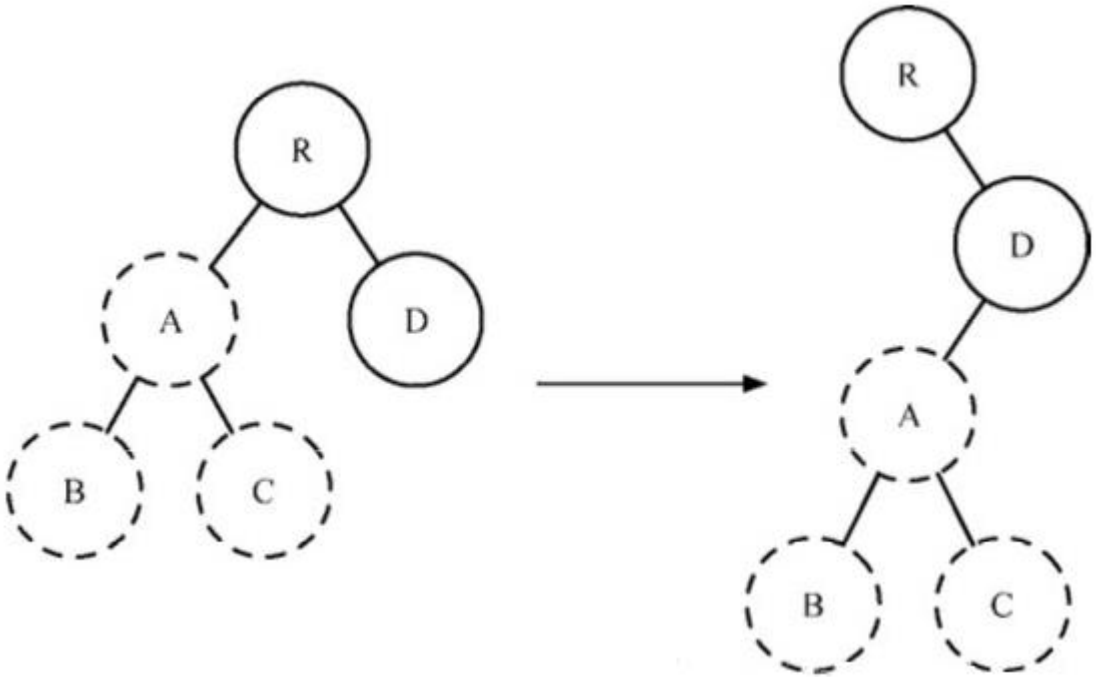
2. compment diff

3. element diff

tree diff



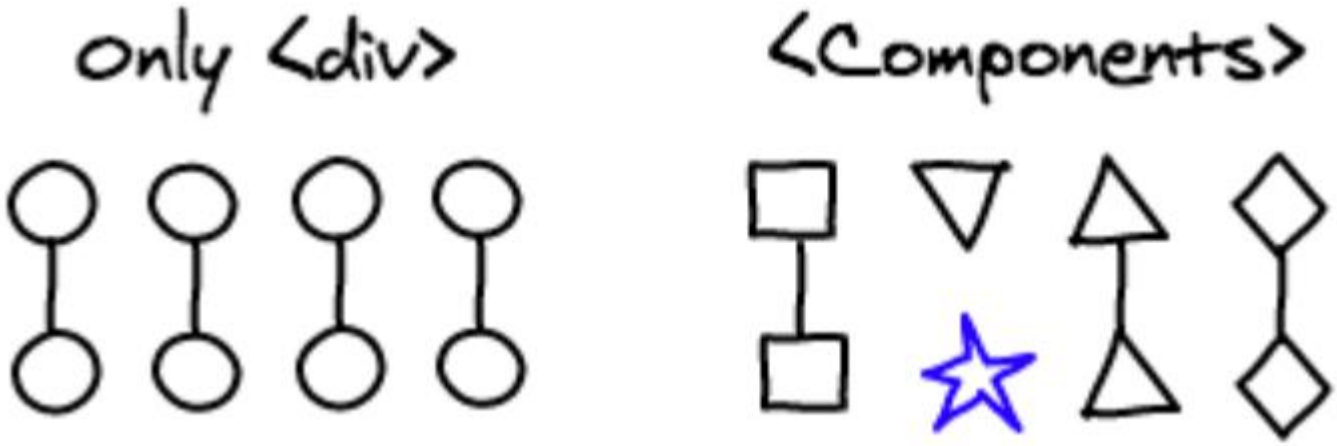
Web UI 中DOM节点跨层级的移动操作特别少，可以忽略不计



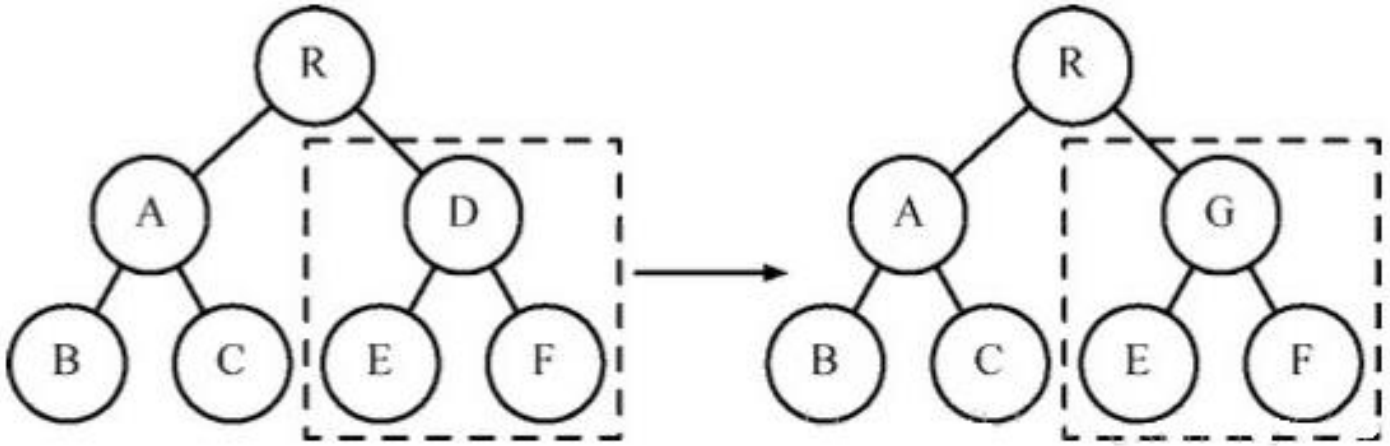
操作逻辑

createA-->createB-->createC-->deleteA

component diff

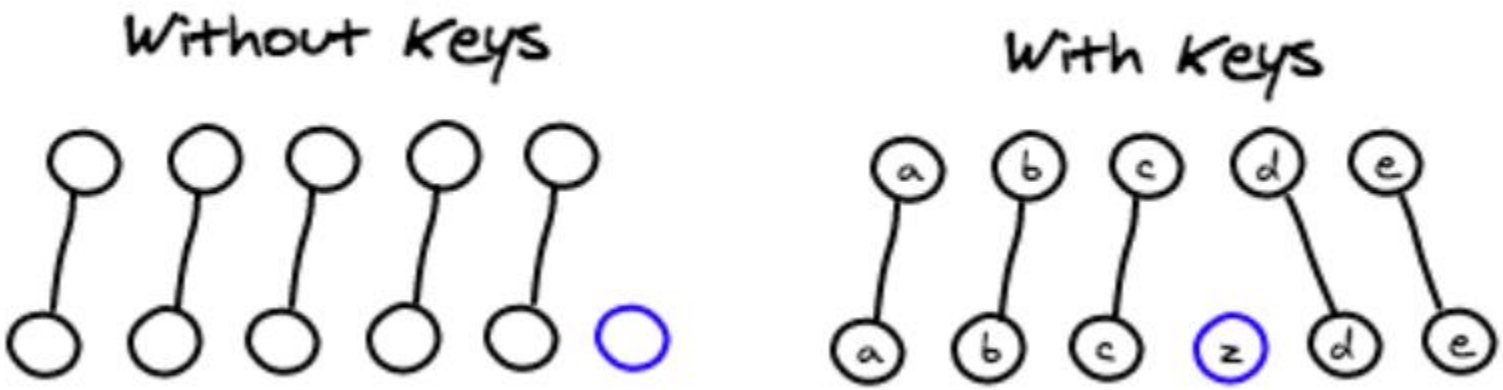


如果是同一类型的组件，按照原策略继续比较 virtual DOM tree。
如果不是，则将该组件判断为 dirty component，从而替换整个组件下的所有子节点。

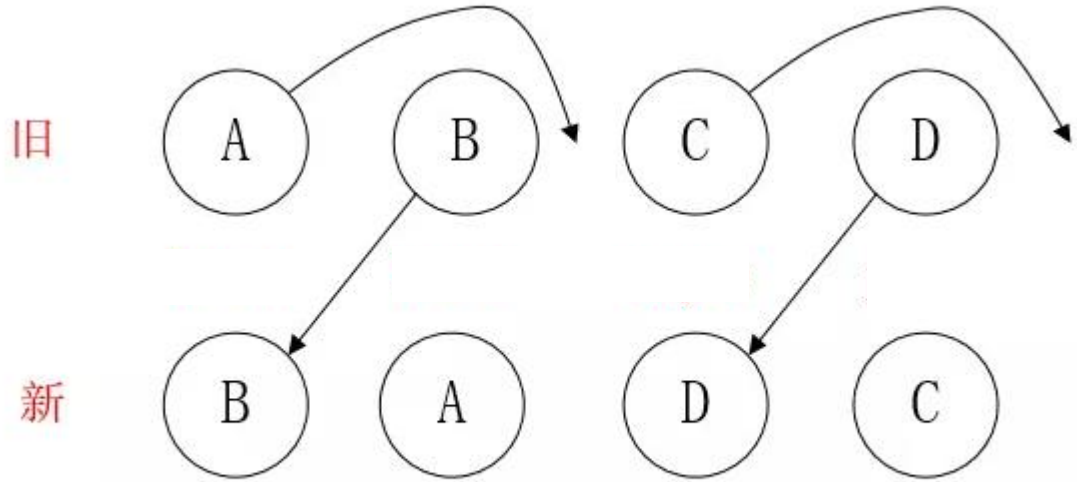


D 与 G 不是同一个组件，直接标记替换

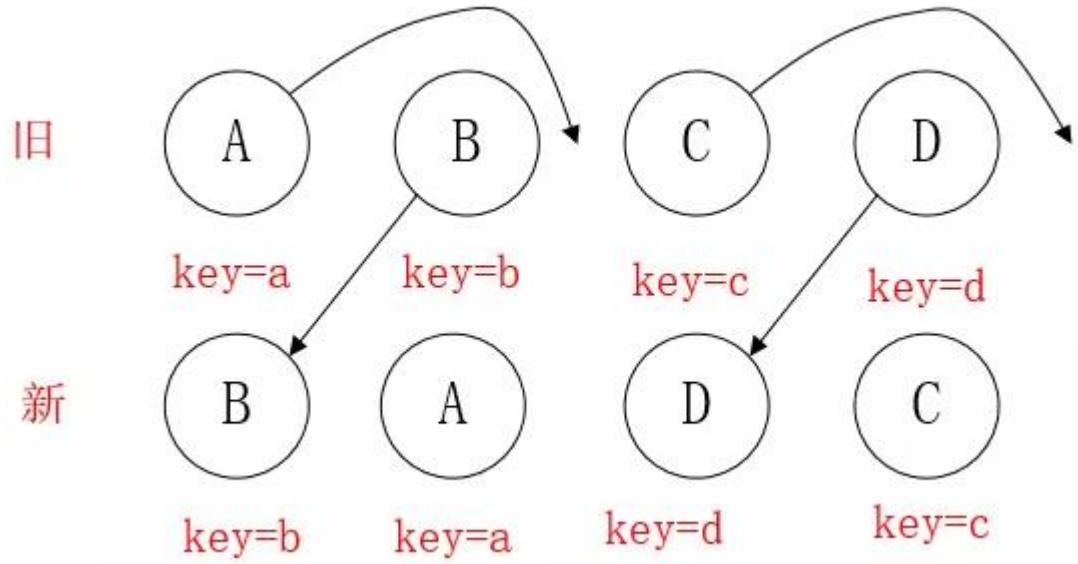
element diff



当节点处于同一层级时，diff提供三种节点操作：**删除、插入、移动**。通过唯一id进行区分他们操作类型



旧的diff 算法：两个数组迭代，判断出哪些节点变化，算法复杂度 O(n^2)



新的diff 算法：利用hashtable 记录每个节点和他们的index 位置， 一轮迭代判断变化并且修改对应 index值，算法复杂度 o(n)

渲染的整个过程

首次渲染

1.创建虚拟DOM

2. 遍历虚拟DOM，并创建对应的真实DOM

3. 将创建好的真实DOM一次性append到容器中

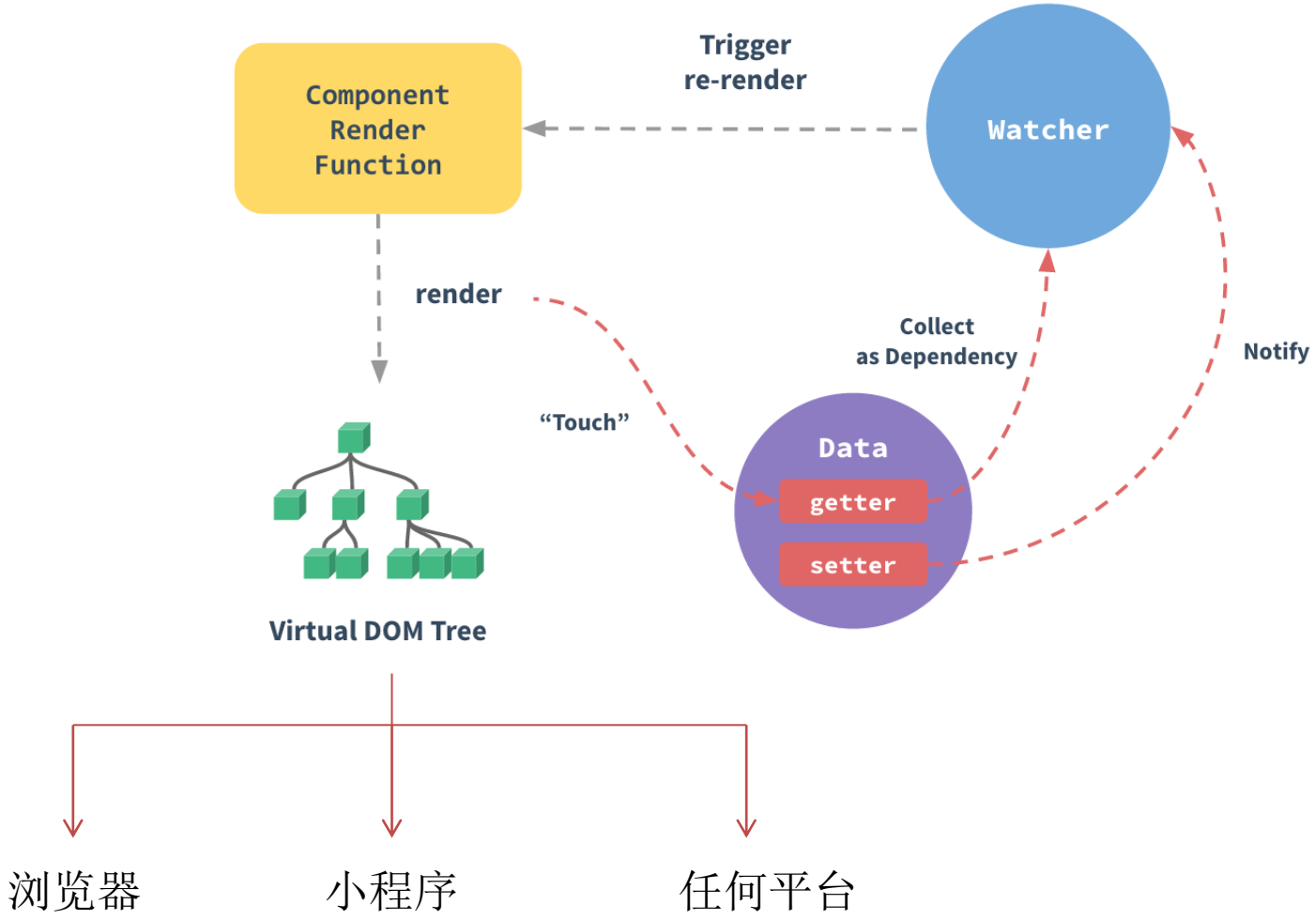
再次渲染

1.对比新旧两个虚拟DOM

2. 生成差异

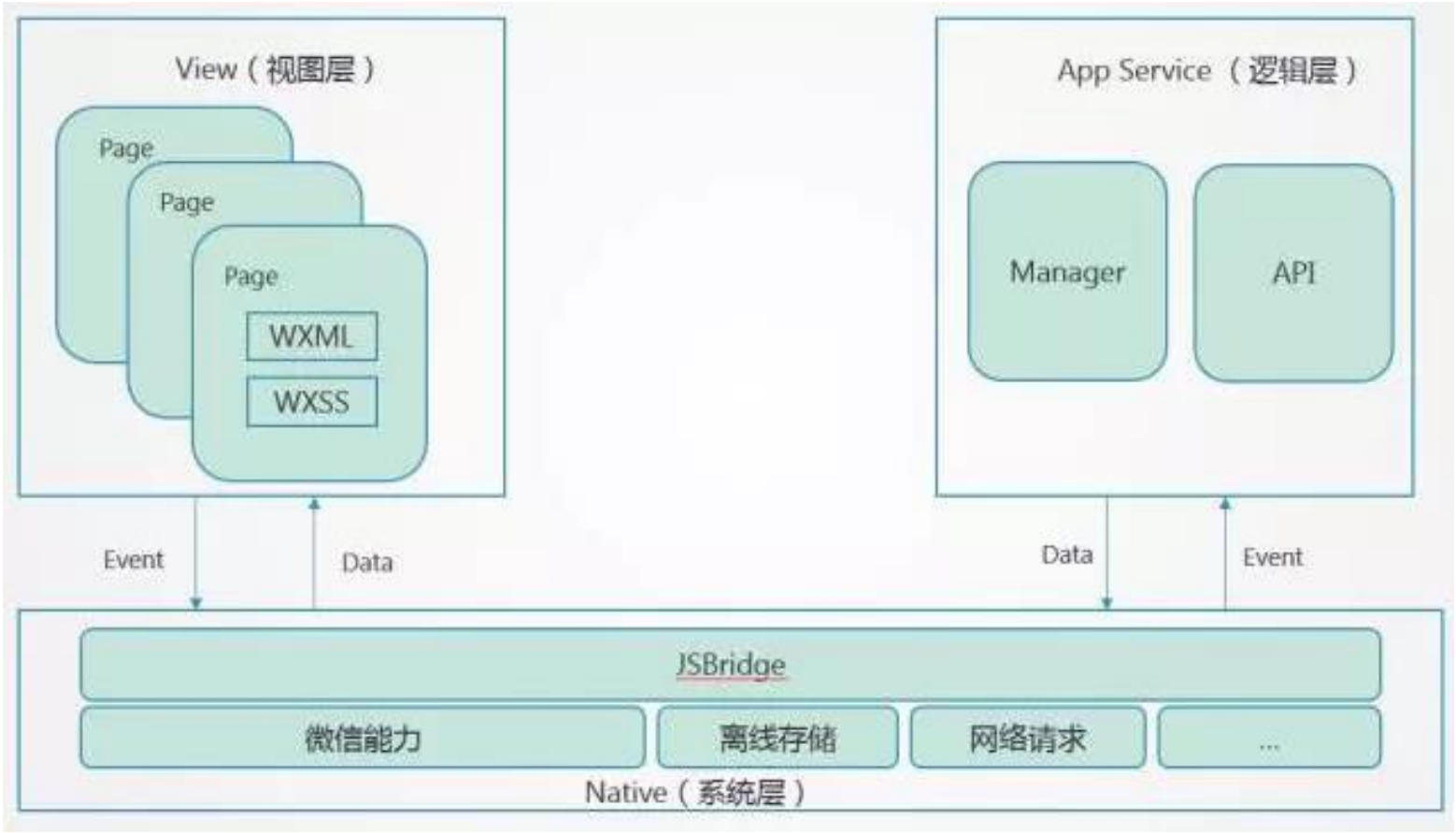
3. 一次性执行所有差异

Vue 未来是平台无关性

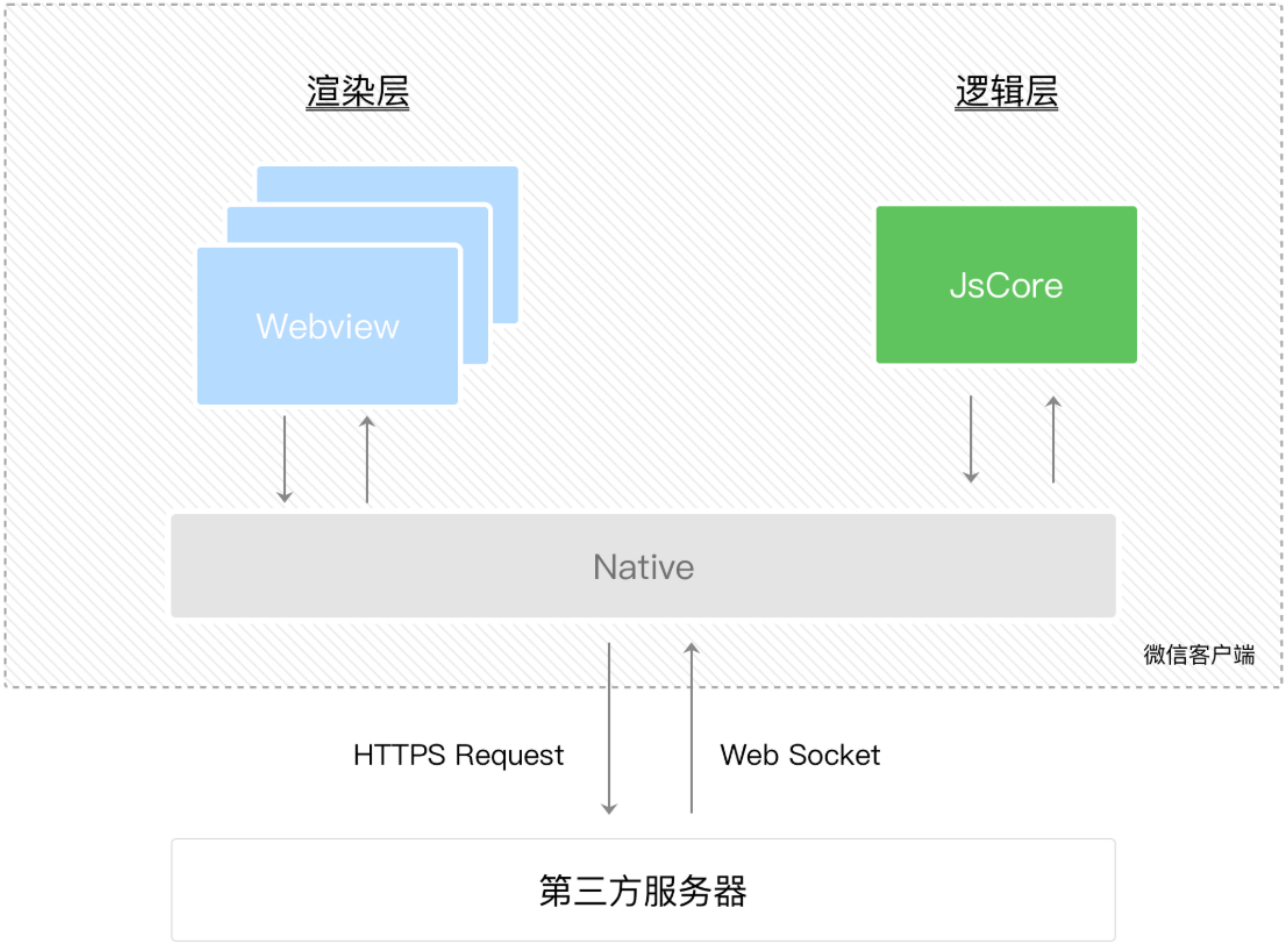


小程序初探

查看demo



1、WAService.js 框架JS库，提供逻辑层基础的API能力
2、WAWebview.js 框架JS库，提供视图层基础的API能力
3、WAConsole.js 框架JS库，控制台
4、app-config.js 小程序完整的配置，包含我们通过app.json里的所有配置，综合了默认配置型
5、app-service.js 我们自己的JS代码，全部打包到这个文件
6、page-frame.html 小程序视图的模板文件，所有的页面都使用此加载渲染，且所有的WXML都拆解为JS实现打包到这里
7、pages 所有的页面，这个不是我们之前的wxml文件了，主要是处理WXSS转换，使用js插入到header区域



为什么 Js 代码不能操作dom 原因？

使用原生渲染，提高体验：

1. Web 渲染。
2. Native 原生渲染。
3. Web 与 Native 两者掺杂，也即我们常说的 Hybrid 渲染。

安全限制

需要阻止开发者使用一些浏览器提供的，诸如跳转页面、操作 DOM、动态执行脚本的开放性接口，例如

1. Xss 攻击
2. Csrf
3. Iframe 广告

基于以上两个原因：

小程序的渲染层和逻辑层分别由 2 个线程管理：

1. 渲染层的界面使用了 WebView 进行渲染

2. 逻辑层采用 JsCore 线程运行 JS 脚本

逻辑层：
创建一个单独的线程去执行 JavaScript，在这个环境下执行的都是有关小程序业务逻辑的代码(ios 用 js core , android 腾讯x5)

渲染层：
界面渲染相关的任务全都在 WebView 线程里执行，通过逻辑层代码去控制渲染哪些界面。一个小程序存在多个界面，所以渲染层存在多个 WebView 线程

Native：
逻辑层和渲染层的通信会由 Native （微信客户端）做中转，逻辑层发送网络请求也经由 Native 转发。

小程序 vs vue渲染

vue virtual dom

->用JS对象模拟DOM树

-> 比较两棵虚拟DOM树的差异

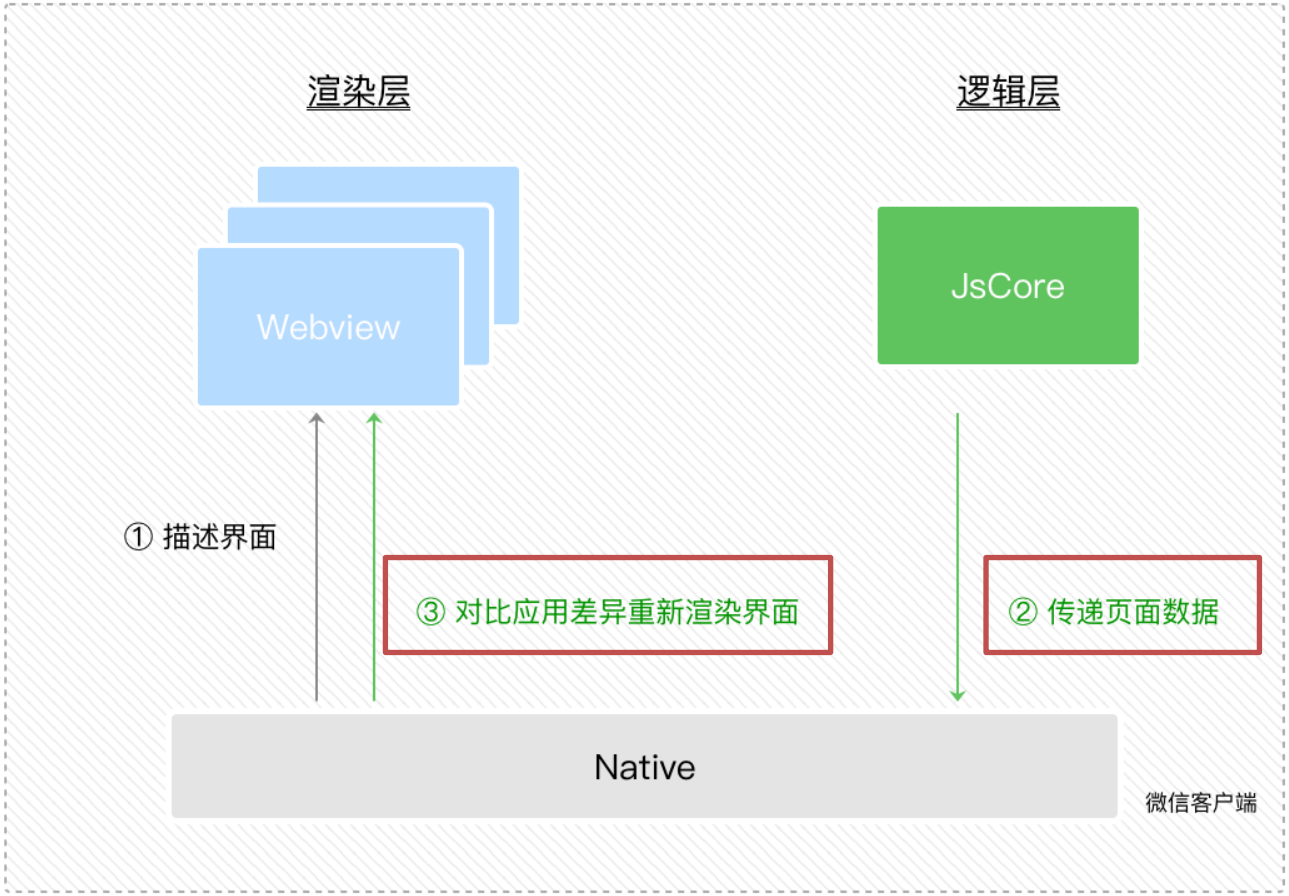
-> 把差异应用到真正的DOM树上

小程序 virtual dom

->用JS对象模拟DOM树

-> 比较两棵虚拟DOM树的差异

-> 把差异应用微信客户端展示()



越来越多业务要跑到小程序又要运行在H5

如何处理？

写两份？

WEPY https://tencent.github.io/wepy/document.html

腾讯团队开源的一款类vue语法规范的小程序框架,借鉴了Vue的语法风格和功能特性,支持了Vue的诸多特征

MpVue http://mpvue.com/mpvue/#-html

美团团队开源的一款使用 Vue.js 开发微信小程序的前端框架。使用此框架，开发者将得到完整的 Vue.js 开发体验，同时为 H5 和小程序提供了代码复用的能力。

Taro https://taro.aotu.io/

京东凹凸实验室开源的一款使用 React.js 开发微信小程序的前端框架。

	wepy	mpvue	taro
作者	腾讯	美团	京东
语法风格	类Vue规范	Vue规范	React标准,支持JSX
多端复用	H5,微信,支付宝	H5,小程序	H5,小程序,RN
集中数据管理	Redux or Mbox	Vuex	Redux
上手成本	熟悉Vue且学习wepy	熟悉Vue即可	熟悉React即可
组件化	自定义组件规范	Vue组件规范	React组件规范
脚手架	wepy-cli	vue-cli	taro-cli
构建工具	框架内置构建工具	webpack	webpack
样式规范	Less/Sass/Styus/ PostCss	Less/Sass/PostCss	Less/Sass/PostCss

IT前沿技术在线大学