Vue和Angular的本质区别

Vue.js的数据观测实现原理和Angular有着本质的不同。了解Angular的读者可能知道，Angular的数据观测采用的是脏检查（dirty checking）机制。每一个指令都会有一个对应的用来观测数据的对象，叫做watcher；一个作用域中会有很多个watcher。每当界面需要更新时，Angular会遍历当前作用域里的所有watcher，对它们一一求值，然后和之前保存的旧值进行比较。如果求值的结果变化了，就触发对应的更新，这个过程叫做digest cycle。

脏检查有两个问题：

1.任何数据变动都意味着当前作用域的每一个watcher需要被重新求值，因此当watcher的数量庞大时，应用的性能就不可避免地受到影响，并且很难优化。

2.当数据变动时，框架并不能主动侦测到变化的发生，需要手动触发digest cycle才能触发相应的DOM 更新。Angular通过在DOM事件处理函数中自动触发digest cycle部分规避了这个问题，但还是有很多情况需要用户手动进行触发。

Vue.js采用的则是基于依赖收集的观测机制。从原理上来说，和老牌MVVM框架Knockout是一样的。依赖收集的基本原理是：

1.将原生的数据改造成“可观察对象”。一个可观察对象可以被取值，也可以被赋值。

2.在watcher的求值过程中，每一个被取值的可观察对象都会将当前的watcher注册为自己的一个订阅者，并成为当前watcher的一个依赖。

3.当一个被依赖的可观察对象被赋值时，它会通知所有订阅自己的watcher重新求值，并触发相应的更新。

4.依赖收集的优点在于可以精确、主动地追踪数据的变化，不存在上述提到的脏检查的两个问题。但传统的依赖收集实现，比如Knockout，通常需要包裹原生数据来制造可观察对象，在取值和赋值时需要采用函数调用的形式，在进行数据操作时写法繁琐，不够直观；同时，对复杂嵌套结构的对象支持也不理想。

Vue.js利用了ES5的Object.defineProperty方法，直接将原生数据对象的属性改造为getter和setter(这是ES5的特性，需要js解释引擎的支持，无法通过各种打shim补丁来实现。这也是为什么Vue不支持IE8及以下版本的原因)，在这两个函数内部实现依赖的收集和触发，而且完美支持嵌套的对象结构。对于数组，则通过包裹数组的可变方法（比如push）来监听数组的变化。这使得操作Vue.js的数据和操作原生对象几乎没有差别[注:在添加/删除属性，或是修改数组特定位置元素时，需要调用特定的函数，如obj.$add(key, value)才能触发更新。这是受ES5的语言特性所限。在操作对象类型数据的时候一定要注意这点，否则无法实现响应。