Vue是一个渐进式框架。数据和DOM会建立起关联，所有的东西都是响应式的。但是只有在data中的数据才是响应式的。Vue的核心是双向绑定和虚拟DOM。

每个Vue实例在创建时都会经历一系列的初始化过程，例如需要设置数据监听、编译模板、实例挂载到DOM、数据更新时更新DOM。

模板语法：

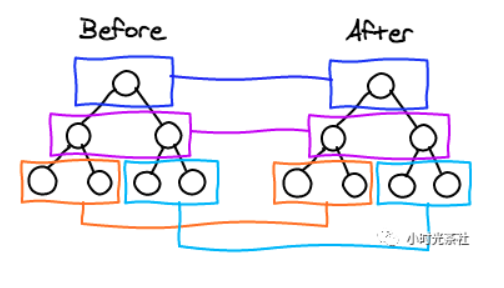
允许开发者声明式地将DOM绑定至底层Vue实例的数据。在底层实现上，Vue将模板编译成虚拟DOM的渲染函数。结合响应式系统，Vue能够智能地计算出最少需要重新渲染多少组件，并把DOM操作次数减到做少。

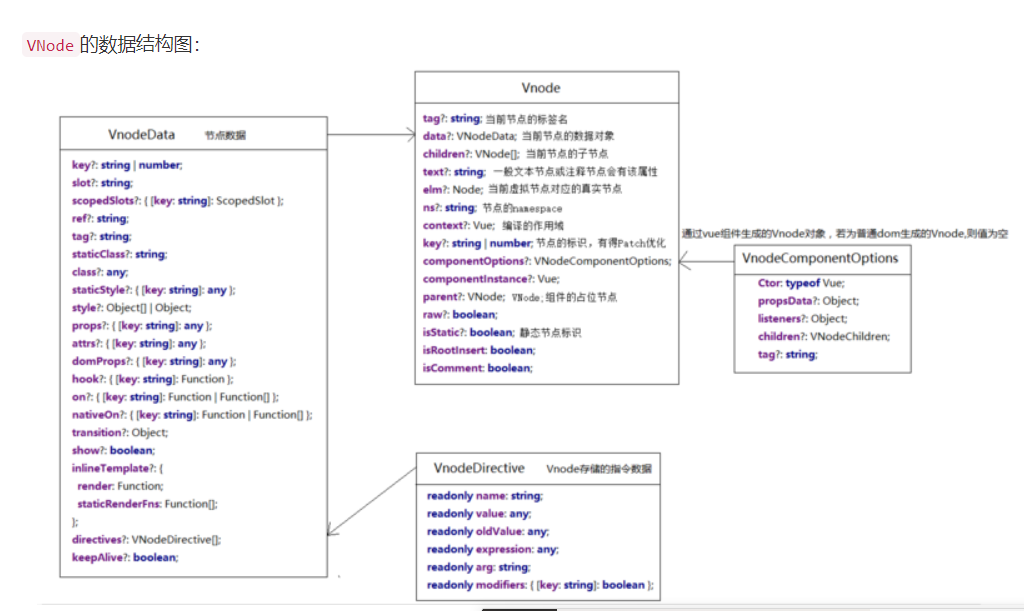
虚拟DOM？

VDOM是树状结构，其节点都是VNODE，VNODE和浏览器DOM中的NODE一一对应，通过VNODE的elm属性可以访问到对应的真实DOM。

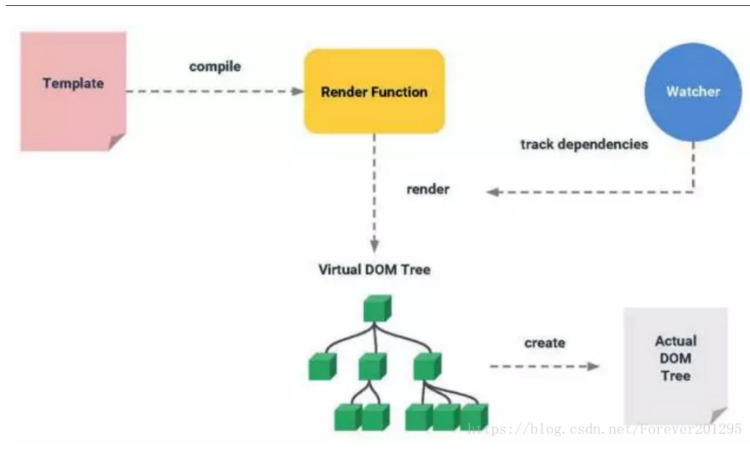
因为VDOM是纯粹的JS对象，所以操作它会很高效，但是VDOM的变更最终会转换成DOM操作，为了实现高效的DOM操作，一套高效的虚拟DOM diff算法是很有必要的。

Vue的diff算法仅在同级的vnode间做diff，递归地进行同级vnode的diff，最终实现整个DOM树的更新。





生成VNode有两种方式，一种是写一个模板或者指定一个el根元素，另一种就是render函数。

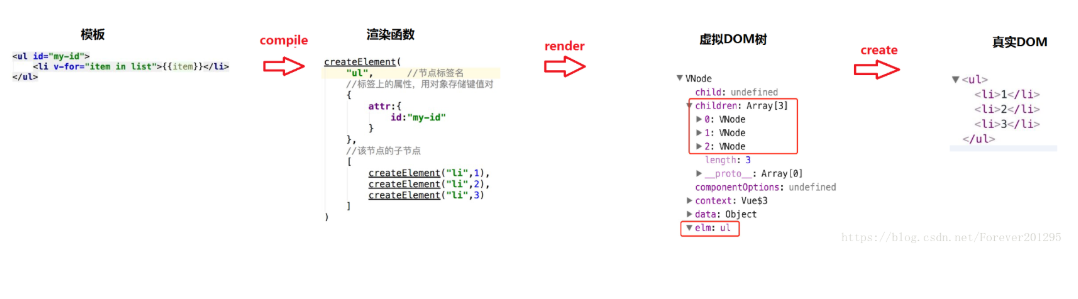


模板最终都会被vue-template-compiler编译成一个Render函数，最后生成一VDOM，VDOM的本质就是在JS和DOM之间做了一个缓存，因为JS直接操作DOM太慢了，而且代价很大。

基本思想就是:

1. 用JS对象表示DOM树的结构，然后用这个DOM树去构建一个真正的DOM树，插到文档当中。
2. 当状态变更的时候，重新构建一个新的对象树。然后用新的树和旧的树对比diff，记录两棵树的差异。
3. 把2所记录的差异应用到步骤1所构建的真正的DOM树上，视图就更新了。

Vue自己能够智能地计算出需要重新渲染多少组件，然后将操作DOM的次数减到最少。



渲染函数简介：

createElement(

// 1.传入你想要生成最外层的根节点

// {String | Object | Function}

// 可以是一个 HTML 标签字符串，组件选项对象，或者

// 解析上述任何一种的一个 async 异步函数，必要参数。

'ul',

// 2.一个包含DOM节点相关属性的数据对象

// {Object}

// 设置 生成DOM 中的属性。一般操作根节点，可选参数。

{

class: {

bg:true

},

style:{

fontSize:"50px"

},

attrs:{

abc:"yyy"

},

domProps:{

/\*innerHTML:"<li>我是里面的html结构</li>" \*/

}

},

// 3.生成子节点

// {String | Array}

// 子节点由 `createElement()` 构建而成，

// 也可以使用字符串来生成“文本节点”。可选参数。

[

createElement("li",1),

createElement("li",2),

createElement("li",3)

]

)