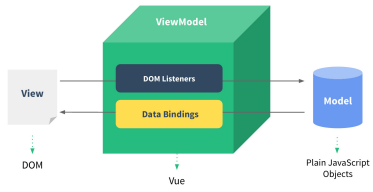
Vue.js

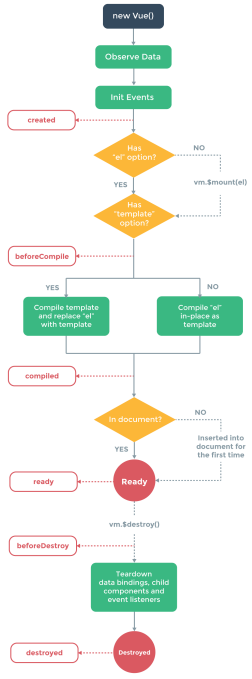
MVVM



Vue.js 的核心是一个响应的数据绑定系统，它让数据与 DOM 保持同步非常简单

Vue.js 拥抱数据驱动的视图概念

生命周期

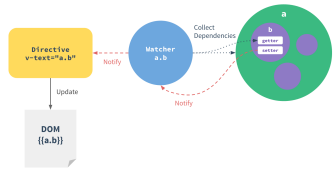


组件

```
Vue.component('my-component', MyComponent)

<div id="example">
  <my-component></my-component>
</div>
// 定义
var MyComponent = Vue.extend({
  template: '<div>A custom component!</div>'
})

// 注册
Vue.component('my-component', MyComponent)

// 创建根实例
new Vue({
  el: '#example'
})
```

父子组件通信

深入响应式原理



把一个普通对象传给 Vue 实例作为它的 data 选项，Vue.js 将遍历它的属性，用 Object.defineProperty 将它们转为 getter/setter。这是 ES5 特性，不能打补丁实现，这便是为什么 Vue.js 不支持 IE8 及更低版本

getter/setters让 Vue.js 追踪依赖，在属性被访问和修改时通知变化

模板中每个指令/数据绑定都有一个对应的 watcher 对象，在计算过程中它把属性记录为依赖

当依赖的 setter 被调用时，会触发 watcher 重新计算 ，也就会导致它的关联指令更新 DOM

vm.$set('b', 2)

Vue.set(data, 'c', 3)

异步更新队列

Vue.js 默认异步更新 DOM。每当观察到数据变化时，Vue 就开始一个队列，将同一事件循环内所有的数据变化缓存起来。如果一个 watcher 被多次触发，只会推入一次到队列中。等到下一次事件循环，Vue 将清空队列，只进行必要的 DOM 更新。在内部异步队列优先使用 MutationObserver，如果不支持则使用 setTimeout(fn, 0)

为了在数据变化之后等待 Vue.js 完成更新 DOM，可以在数据变化之后立即使用 Vue.nextTick(callback)