Vue源码：<https://vue-js.com/learn-vue/>

1. 变化侦测篇  
   vue最大的特点之一就是数据驱动视图。那么什么是数据驱动视图呢？在这里，我们可以把数据理解为状态，而视图就是用户可直观看到页面。页面不可能是一成不变的，它应该是动态变化的，而它的变化也不应该是无迹可寻的，它或者是由用户操作引起的，亦或者是由后端数据变化引起的，不管它是因为什么引起的，我们统称为它的状态变了，它由前一个状态变到了后一个状态，页面也就应该随之而变化，所以我们就可以得到如下一个公式：  
    UI = render(state)  
   上述公式中：状态state是输入，页面UI输出，状态输入一旦变化了，页面输出也随之而变化。我们把这种特性称之为数据驱动视图  
   那Vue是怎么知道state变化了呢？换句话说，数据变化了是怎么通知给Vue呢？那么，这就引出了Vue中的变化侦测  
   变化侦测它在目前的前端三大框架中均有涉及。在Angular中是通过脏值检查流程来实现变化侦测；在React是通过对比虚拟DOM来实现变化侦测  
   总结：  
    通过object.defineproperty()方法。其中有一个get和一个set。在读取数据时会触发get方法。在修改数据的时候会触发set方法。实现了对object数据的可观测。封装了一个observer方法，将object中的所有属性包括字属性都转为了setter和getter的形式来侦测变化（observer方法：给value新增了一个\_ob\_的属性，值是该value的observer实例，这个操作相当于为value打上标记。表示它已经被转为响应式了，避免重复操作，然后，object类型的数据会调用walk方法将每一个属性转换成getter/setter的形式来侦测变化。然后会调用defineReactive方法，该方法将object下的所有子属性也变为了setter和getter的形式）数据可观测之后，问题又来了。假如观察到数据发生变化，那么应该通知谁去改变，不可能数据变了，让视图重新刷新一遍。  
   因此将依赖该数据的地方全部收集起来。简称依赖收集，接下来的问题是，什么时候收集依赖，收集起来后，什么时候更新。上一个方法使得每一个object变为getter和setter，只要读取数据就会触发getter，修改数据就会触发setter，因此在getter中进行依赖收集，在setter中通知依赖更新。而收集起来的依赖需要保存到一个地方，因此依赖管理器Dep应运而生，（Dep中先定义了一个存放依赖的数组subs，然后定义了几个实例对依赖进行添加，删除和通知等操作。）。在vue中还给依赖定义了一个watcher实例。在之后的数据变化中我们不直接去通知依赖而是通知依赖对应的watcher实例，由watcher去通知真正的视图，或者执行用户的一些逻辑.
2. 虚拟DOM篇
3. 模板编译篇
4. 实例方法篇
5. 全局API篇
6. 生命周期篇
7. 指令篇
8. 过滤器篇
9. 内置组件篇