一

1.首先进行Vue的初始化，初始化Vue的实例成员以及静态成员。

2.当初始化结束之后，开始调用构造函数，在构造函数中调用this.\_init()，这个方法相当于 我们整个Vue的入口。

3.在\_init()中调用this.$mount()，共有两个this.$mount()。

4.接下来调用mountComponent()

5.然后定义了updateComponent()

6.再接下来就是创建Watcher对象，在创建Watcher时，传递了updateComponent这个函数，这个函数最终是在Watcher内部调用的。在Watcher创建完之后还调用了get方法，在get方法中，会调用updateComponent()。

7.然后触发了生命周期的钩子函数mounted,挂载结束，最终返回Vue实例。

二

当一个Vue实例创建时，vue会遍历data选项的属性，用Object.defineProperty将它们转为getter/setter并且在内部追踪相关依赖，在属性被访问和修改时通知变化。每个组件实例都有相应的watcher程序实例，它会在组件渲染的过程中把属性记录为依赖，之后当依赖项的setter被调用时，会通知watcher重新计算，从而致使它关联的组件得以更新。

三

可以减少 dom 的操作，减少 diff 和渲染所需要的时间，提升了性能。

四

1.缓存公共的 mount 函数，并重写浏览器平台的 mount

2.判断是否传入了 render 函数，没有的话，是否传入了 template ，没有的话，则获取 el 节点的 outerHTML 作为 template

3.调用 baseCompile 函数

4.解析器(parse) 将模板字符串的模板编译转换成 AST 抽象语法树

5.优化器(optimize) - 对 AST 进行静态节点标记，主要用来做虚拟DOM的渲染优化

6.通过 generate 将 AST 抽象语法树转换为 render 函数的 js 字符串

7.将 render 函数 通过 createFunction 函数 转换为 一个可以执行的函数

8.将 最后的 render 函数 挂载到 option 中

9.执行 公共的 mount 函数