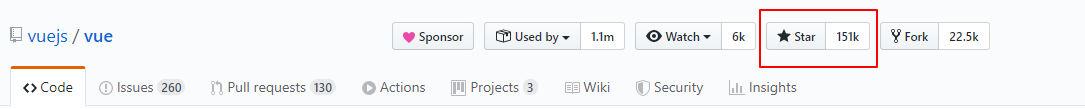
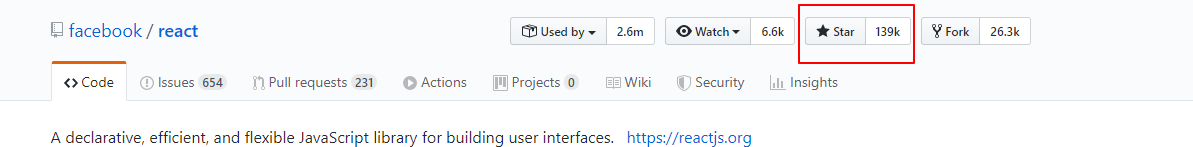
# Vue当前的影响力

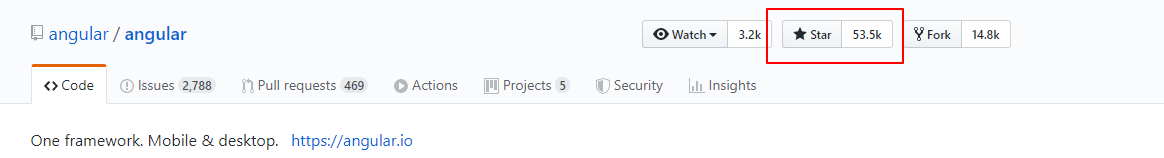
1.vue是一个渐进式框架，核心是视图渲染，其次是组件，再后面可以根据应用的大小，增加路由机制，也可以使用vue的全家桶，状态管理和构建工具。可以根据项目情况自行选择。

2.vue 2.0比1.0新增了很多特性，但API基本没有变化，学习曲线平缓。

3.vue 使用者非常多，社区活跃







# 二．Vue.js 源码目录设计

# **Vue.js 源码目录设计**

Vue.js 的源码都在 src 目录下，其目录结构如下。

src

├── compiler # 编译相关

├── core # 核心代码

├── platforms # 不同平台的支持

├── server # 服务端渲染

├── sfc # .vue 文件解析

├── shared # 共享代码

## **compiler**

compiler 目录包含 Vue.js 所有编译相关的代码。它包括把模板解析成 ast 语法树，ast 语法树优化，代码生成等功能。

编译的工作可以在构建时做（借助 webpack、vue-loader 等辅助插件）；也可以在运行时做，使用包含构建功能的 Vue.js。显然，编译是一项耗性能的工作，所以更推荐前者——离线编译。

## **core**

core 目录包含了 Vue.js 的核心代码，包括内置组件、全局 API 封装，Vue 实例化、观察者、虚拟 DOM、工具函数等等。

这里的代码可谓是 Vue.js 的灵魂，也是我们之后需要重点分析的地方。

## **platform**

Vue.js 是一个跨平台的 MVVM 框架，它可以跑在 web 上，也可以配合 weex 跑在 natvie 客户端上。platform 是 Vue.js 的入口，2 个目录代表 2 个主要入口，分别打包成运行在 web 上和 weex 上的 Vue.js。

我们会重点分析 web 入口打包后的 Vue.js，对于 weex 入口打包的 Vue.js。

## **server**

Vue.js 2.0 支持了服务端渲染，所有服务端渲染相关的逻辑都在这个目录下。注意：这部分代码是跑在服务端的 Node.js，不要和跑在浏览器端的 Vue.js 混为一谈。

服务端渲染主要的工作是把组件渲染为服务器端的 HTML 字符串，将它们直接发送到浏览器，最后将静态标记"混合"为客户端上完全交互的应用程序。

## **sfc**

通常我们开发 Vue.js 都会借助 webpack 构建， 然后通过 .vue 单文件的编写组件。

这个目录下的代码逻辑会把 .vue 文件内容解析成一个 JavaScript 的对象。

## **shared**

Vue.js 会定义一些工具方法，这里定义的工具方法都是会被浏览器端的 Vue.js 和服务端的 Vue.js 所共享的。

## **总结**

从 Vue.js 的目录设计可以看到，作者把功能模块拆分的非常清楚，相关的逻辑放在一个独立的目录下维护，并且把复用的代码也抽成一个独立目录。

这样的目录设计让代码的阅读性和可维护性都变强，是非常值得学习和推敲的。

# 三．揭开Vue的面纱

## **Vue 的入口**

在 src/platforms/web/runtime/index.js 中：

import Vue from 'core/index'

**import** { initMixin } **from** './init'  
**import** { stateMixin } **from** './state'  
**import** { renderMixin } **from** './render'  
**import** { eventsMixin } **from** './events'  
**import** { lifecycleMixin } **from** './lifecycle'  
**import** { warn } **from** '../util/index'  
  
**function** Vue (options) {  
 **if** (process.env.NODE\_ENV !== 'production' &&  
 !(**this instanceof** Vue)  
 ) {  
 warn('Vue is a constructor and should be called with the `new` keyword')  
 }  
 **this**.\_init(options)  
}  
  
initMixin(Vue)  
stateMixin(Vue)  
eventsMixin(Vue)  
lifecycleMixin(Vue)  
renderMixin(Vue)  
  
**export default** Vue

在这里，我们看到了 Vue 就是一个用 Function 实现的类，我们只能通过 new Vue 去实例化它。

为何 Vue 不用 ES6 的 Class 去实现呢？我们往后看这里有很多 xxxMixin 的函数调用，并把 Vue 当参数传入，它们的功能都是给 Vue 的 prototype 上扩展一些方法，Vue 按功能把这些扩展分散到多个模块中去实现，而不是在一个模块里实现所有，这种方式是用 Class 难以实现的。这么做的好处是非常方便代码的维护和管理，这种编程技巧也非常值得我们去学习。