

前端框架

不会前端开发的后端不是一个好的后端开发，平时写点小项目可以用得上，先简单了解一下前端这个概念。

前端：HTML（超文本标记语言），CSS（层叠样式表）和JavaScript（脚本语言）。

HTML，通常说的H5，其实按标准来说，HTML4的后续版本不带编号了，并保证向前的兼容性。

CSS的版本3，增加了translate()，能完成以前一定需要js才能做到的动画，同时增加了flex弹性盒子（响应式设计，提供一种更加有效的方式来对一个容器中的子元素进行排列、对齐和分配空白空间，以往使用float）。

jQuery

jQuery是一个快速、简洁的JavaScript框架，是继Prototype之后又一个优秀的JavaScript代码库（或JavaScript框架）。

jQuery设计的宗旨是“write Less, Do More”，即倡导写更少的代码，做更多的事情。

敲黑板：简单来说就是封装了一部分的函数，简化了原生js的写法，在实际使用时，jQuery对DOM树进行操控，即：首先有一个写好的html页面，再对其修改操作，如写按钮事件函数，点击隐藏，切换，页面跳转等。

jQuery库包含以下功能：

- HTML元素选取
- HTML元素操作
- CSS操作
- HTML事件函数
- JavaScript特效和动画
- HTML DOM遍历和修改
- AJAX
- Utilities

除此之外，jQuery还提供了大量的插件。它兼容各种主流浏览器，如IE 6.0+、FF 1.5+、Safari 2.0+、Opera 9.0+等。

这个曾经也是现在依然最流行的web前端js库，可是现在无论是国内还是国外他的使用率正在渐渐被其他的js库所代替，随着浏览器厂商对HTML5规范统一遵循以及ECMA6在浏览器端的实现，jQuery的使用率将会越来越低。

敲黑板（思考）：**为什么说jQuery的使用率越来越低？**

一、JS更新带来的冲击

1. 快速选取DOM节点

对于大部分使用jQuery的开发工程师来说，能够快速选取DOM节点，这个无疑是一个重要的原因，但是就目前情况来说，这个优势显然已经荡然无存了，为什么呢？跟大家说两个API，这两个API已经非常多的人在用了，就是document.querySelector和document.querySelectorAll方法。这两个方法可以通过传入css选择器形式的字符串，就可以匹配到预期的DOM节点。以下是目前两个API的兼容情况：

浏览器兼容性

Update compatibility data on GitHub

	Desktop						Mobile						
	Chrome	Edge	Firefox	Safari	Opera	Internet Explorer	Android	Chrome	Edge	Firefox	Safari	Opera	Internet Explorer
Basic support	1	Yes	3.5	8	10	3.2	Yes	Yes	Yes	Yes	10	3.2	7

浏览器兼容性

Update compatibility data on GitHub

	Desktop						Mobile						
	Chrome	Edge	Firefox	Safari	Opera	Internet Explorer	Android	Chrome	Edge	Firefox	Safari	Opera	Internet Explorer
Basic support	1	Yes	3.5	8	10	3.2	Yes	Yes	Yes	Yes	10	3.2	7

从图中可以看到，这两个API已经很好的兼容各个浏览器。
Vue中也是使用此API进行元素获取的：

```

export function query (el: string | Element): Element {
  if (typeof el === 'string') {
    const selected = document.querySelector(el)
    if (!selected) {
      process.env.NODE_ENV !== 'production' && warn(
        'Cannot find element: ' + el
      )
      return document.createElement('div')
    }
    return selected
  } else {
    return el
  }
}

```

知乎 @Lemonade

所以说jQuery快速选择DOM节点的优势已经不存在了。

2. 方便操作DOM元素的API

可以方便操作DOM元素的API，比如addClass、removeClass、toggleClass。现在原生JS也得到了支持，这个API叫做classList。

Desktop	Mobile						
Feature	Chrome	Edge	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)	
Basic support	8	12	3.6 (1.9.2)	10 [1]	11.50	5.1	
toggle() method's second argument	24	12	24 (24)	未实现[2]	15	7	
Multiple arguments for add() & remove()	24	12	26 (26)	未实现	15	7	
replace()	61	?	49 (49)	未实现	未实现	未实现	






知乎 @Lemonade

虽然说IE兼容的不太完美，但是最基本该实现也都实现了。

3. 动画

现在CSS3动画技术已经非常的成熟，已经完全可以取代jQuery做的动画，而且还能比jQuery的animate方法实现更复杂的动画，兼容性好，性能消耗小，何乐而不为呢？举个例子吧，比方说如果实现背景颜色过度，CSS3可以完美的实现，但是jQuery就不行。并且现在已经出现了很多优秀的CSS3动画库，大名鼎鼎的Animate.css库大家肯定都有耳闻吧。

浏览器支持

IE	Firefox	Chrome	Safari	Opera
				

Internet Explorer 10、Firefox 以及 Opera 支持 animation 属性。

Safari 和 Chrome 支持替代的 -webkit-animation 属性。

注释：Internet Explorer 9 以及更早的版本不支持 animation 属性。

知乎 @Lemonade

那么现在出现的新概念 **Virtual DOM (虚拟DOM)**，就可以解决这个问题。其实**Virtual DOM**就是对真实DOM节点的描述，通过改变Virtual DOM来以最小变动来改变真实DOM (Virtual DOM不一定真的比jQuery性能更好)。

Vue

Vue (读音 /vju:/, 类似于 view) 是一套用于构建用户界面的渐进式框架。

与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用，通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。

Vue是一个兴起的前端js库，是一个精简的MVVM。从技术角度讲，Vue.js 专注于 MVVM 模型的 ViewModel 层。

它通过双向数据绑定把 View 层和 Model 层连接了起来，通过对数据的操作就可以完成对页面视图的渲染。

当然还有很多其他的MVVM框架如Angular，React都是大同小异，本质上都是基于MVVM的理念。然而Vue以他独特的优势简单，快速，组合，紧凑，强大而迅速崛起。

敲黑板：为什么说“通过对数据的操作就可以完成对页面视图的渲染”？

Vue.js是一种基于MVVM方式的框架，专注于MVVM模型的ViewModel层，通过双向数据绑定的方式将Model层和View层连接起来。

而在Vue中实现**双向数据绑定**的原理是：采用数据劫持结合发布者-订阅者的方式，通过Object.defineProperty()来劫持各个属性的setter，getter，在数据变动时，发布消息给订阅者，触发相应的监听回调。

通俗的讲，就是利用observe监听Model层的数据变化；利用Compile来编译解析模板指令，最终利用Watcher搭起Observer和Compile之间的通信桥梁，达到数据变化 (model)-》视图更新(view)；视图变化(view)-》数据(model) 变更的双向绑定效果。

敲黑板：可以说操作DOM的事儿，就留给框架去做了。这比传统jQuery开发效率高，代码可维护性高，可扩展性强、性能好。

jQuery操作思想

jQuery是使用选择器 (\$) 选取DOM对象，对其进行赋值、取值、事件绑定等操作，其实和原生的HTML的区别只在于可以更方便的选取和操作DOM对象，

而数据和界面是在一起的。比如需要获取label标签的内容：\$("lable").val();，它还是依赖DOM元素的值。

Vue操作思想

Vue基于一种MVVM模式，使用数据驱动的方式，通过Vue对象将数据和View完全分离开来了。对数据进行操作不再需要引用相应的DOM对象，可以说数据和View是分离的，他们通过Vue对象这个vm实现相互的绑定。

jQuery应用场景

jquery侧重样式操作，比如一些H5的动画页面；需要js来操作页面样式的页面。

敲黑板：jQuery的编程思想是首先编写HTML和CSS的页面展示再操作DOM树，而框架是首先考虑页面的功能，再进行前端的展示，编程思想正好相反。

Vue应用场景

Vue侧重数据绑定，比如复杂数据操作的后台页面；表单填写页面。

敲黑板：二者也是可以结合起来一起使用的，vue侧重数据绑定，jquery侧重样式操作，动画效果等，则会更加高效率的完成业务需求。

Vue带来了哪些改变？

我是一名后端开发，刚开始入门时接触js然后jQuery，感觉它更像是一把剪刀，简单而犀利，通常是配合一些框架来完成一些静态页面开发的工作。

因为jQuery的诸多局限性导致前端工程师的发展受到了很多的限制，只能做一些表面性的工作，并不能实现前后端分离开发。

而近期出现的Vue，它给前端带来了无限的可能和改变。

改变一：真正意义上的前端工程师

之前开发都是前端做静态页面，把页面给到后台程序员改成jsp、php、asp等等...一顿乱改，一顿塞变量，做完以后页面样式乱七八糟，最后你再调整css。说白了你会html，css就行了，基本没什么门槛，可以这么说。

有了Vue和Node的前端工程化以后，前端工程师能做的事情越来越多，后台人员只需要抛过来一个Api，剩下的就可以都交给前端了。

改变二：服务端渲染VS客户端渲染

传统的jsp、php或是模板渲染也好，都是服务端渲染，就是客户端一个请求，服务器直接把整个页面返回给你，简单粗暴。（Spring Boot是通过模板引擎，

由服务端完成的渲染工作)

但是vue开发是前后端分离开发，通过api进行交互，客户端请求服务器返回json数据，由客户端进行渲染。

不仅减轻了服务器的压力速度更快而且渲染更加优雅，代码更容易维护。

改变三：渲染优雅，代码易维护

jQuery是通过DOM来控制数据，不仅笨重而且渲染数据特别麻烦，而Vue是通过数据来控制状态，通过控制数据来控制渲染，变量可以直接写在标签中，渲染更加优雅。

因为前端代码和后台代码都是分开的，所以项目更容易维护，开发效率更高。

改变四：项目工程化，结合npm直接安装第三方库

Vue让前端项目更加工程化，同时也规范了前端工程师的代码，而node和npm的加入才是vue能蓬勃发展的重要原因。

Node为Vue提供了本地server和模块化开发的思路，npm更能安装Vue项目需要的模块，配合Vue使用，比如Moment.js Element ui vuex等等，这些第三方库让Vue有了无限的可能。

敲黑板（补充下）：传统开发jQuery是命令式编程，现代框架开发是函数式编程。现代框架开发，可以使用Webpack（当然使用jQuery也可以使用Webpack），**可以使用人家提供的现成的脚手架，比方说create-react-app, vue-cli。极大提高了开发的效率，并且可以使用最新的ES6、ES7语法进行开发，在编码体验上，就提高了一个档次。**