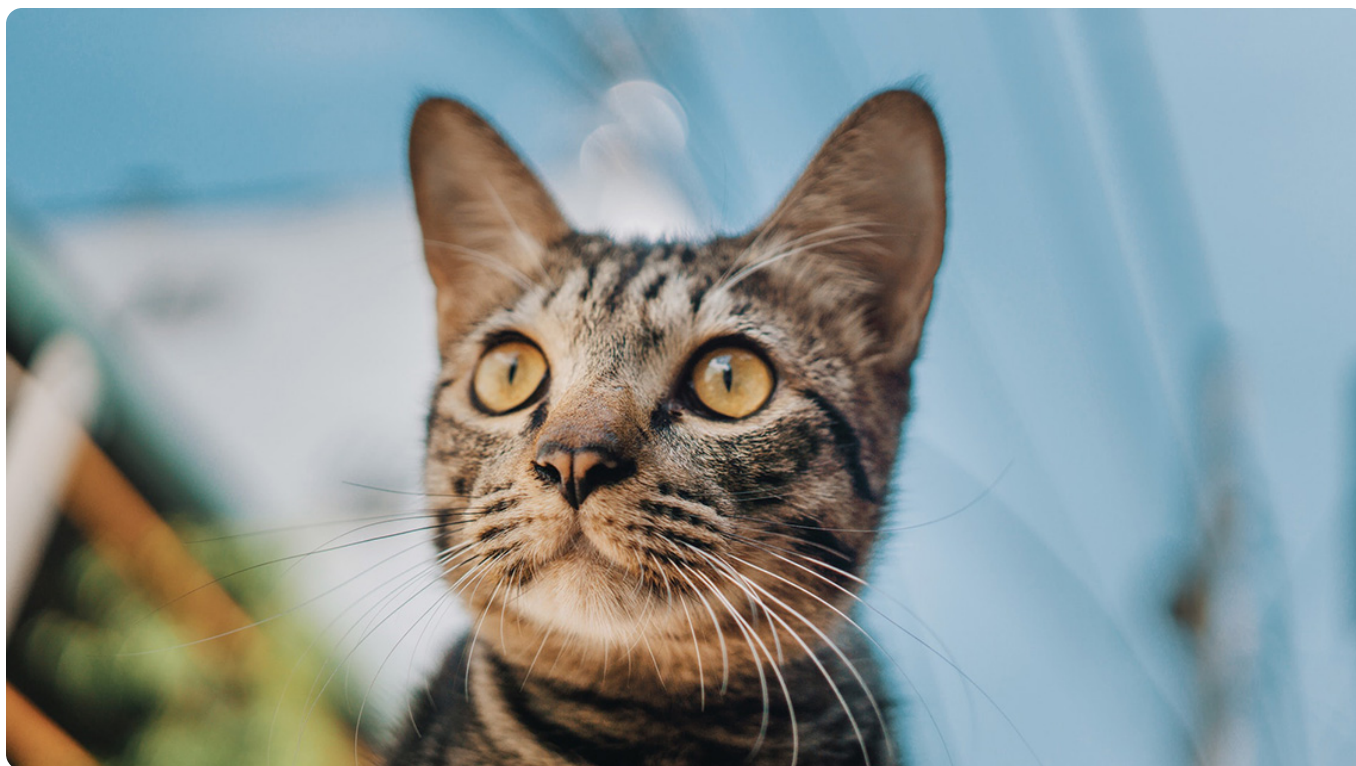


明确你的前端学习路线与方法

winter 2019-01-17



你好，我是 winter。今天我们一起聊聊前端的学习路线与方法。

在“开篇词”中，我和你简单回顾了前端行业的发展，到现在为止，前端工程师已经成为研发体系中的重要岗位之一。可是，与此相对的是，我发现极少或者几乎没有大学的计算机专业愿意开设前端课程，更没有系统性的教学方案出现。大部分前端工程师的知识，其实都是来自于实践和工作中零散的学习。

这样的现状就引发了一系列的问题。

首先是前端的基础知识，常常有一些工作多年的工程师，在看到一些我认为很基础的 JavaScript 语法的时候，还会惊呼“居然可以这样”。是的，基础知识的欠缺会让你束手束脚，更限制你解决问题的思路。

其次，技术上存在短板，就会导致前端开发者的上升通道不甚顺畅。特别是一些小公司的程序员，只能靠自己摸索，这样就很容易陷入重复性劳动的陷阱，最终耽误自己的职业发展。

除此之外，前端工程师也会面临技术发展问题带来的挑战。前端社区高度活跃，前端标准也在快速更新，这样蓬勃发展对技术来说无疑是好事，但是副作用也显而易见，它使得前端工程师的学习压力变得很大。

我们就拿 JavaScript 标准来说，ES6 中引入的新特性超过了过去十年的总和，新特性带来的实践就更多了，仅仅是一个 Proxy 特性的引入，就支持了 VueJS 从 2.0 到 3.0 的内核原理完全升级。

缺少系统教育 + 技术快速革新，在这样的大环境下，前端工程师保持自学能力就显得尤其重要了。

那么，前端究竟应该怎么学呢？我想，我可以简单分享一下自己的经验。

学习路径与学习方法

首先是 **0 基础入门**的同学，你可以读几本经典的前端教材，比如《JavaScript 高级程序设计》《精通 CSS》等书籍，去阅读一些参考性质的网站也是不错的选项，比如 [MDN](#)。

如果你至少已经**有了 1 年以上的工作经验**，希望在技术上有一定突破，那么，这个专栏就可以是你技术进阶的一个选项了。

在这个专栏中，我希望传达的不仅仅是具体的知识点，还有体系架构和学习方法。我希望达到三个目标：

带你摸索出适合自己的前端学习方法；

帮助你建立起前端技术的知识架构；

让你理解前端技术背后的核心思想。

在开始具体的知识讲解之前，这篇文章中，我想先来谈两个前端学习方法。

第一个方法：建立知识架构

第一个方法是建立自己的知识架构，并且在这个架构上，不断地进行优化。

我们先来讲讲什么叫做知识架构？我们可以把它理解为知识的“目录”或者索引，它能够帮助我们**把零散的知识组织起来，也能够帮助我们发现一些知识上的盲区。**

当然，知识的架构是有优劣之分的，最重要的就是逻辑性和完备性。

我们来思考一个问题，如果我们要给 JavaScript 知识做一个顶层目录，该怎么做呢？

如果我们把一些特别流行的术语和问题，拼凑起来，可能会变成这样：

类型转换；

this 指针；

闭包；

作用域链；

原型链；

.....

这其实不是我们想要的结果，因为这些知识点之间，没有任何逻辑关系。它们既不是并列关系，又不是递进关系，合在一起，也就没有任何意义。这样的知识架构，无法帮助我们去发现问题和理解问题。

如果让我来做，我会这样划分：

文法

语义

运行时

为什么这样分呢，因为对于任何计算机语言来说，必定是“用规定的文法，去表达特定语义，最终操作运行时的”一个过程。

这样，JavaScript 的任何知识都不会出现在这个范围之外，这是知识架构的完备性。我们再往下细分一个层级，就变成了这个样子：

文法

◦ 词法

◦ 语法

语义

运行时

- 类型
- 执行过程

我来解释一下这个划分。

文法可以分成词法和语法，这来自编译原理的划分，同样是完备的。语义则跟语法具有一一对应关系，这里暂时不区分。

对于运行时部分，这个划分保持了完备性，**我们都知道：程序 = 算法 + 数据结构，那么，对运行时来说，类型就是数据结构，执行过程就是算法。**

当我们再往下细分的时候，就会看到熟悉的概念了，词法中有各种直接量、关键字、运算符，语法和语义则是表达式、语句、函数、对象、模块，类型则包含了对象、数字、字符串等.....

这样逐层向下细分，知识框架就初见端倪了。在顶层和大结构上，我们通过逻辑来保持完备性。如果继续往下，就需要一些技巧了，我们可以寻找一些线索。

比如在 JavaScript 标准中，有完整的文法定义，它是具有完备性的，所以我们可以根据它来完成，我们还可以根据语法去建立语义的知识架构。实际上，因为 JavaScript 有一份统一的标准，所以相对来说不太困难。

如果是浏览器中的 API，那就困难了，它们分布在 w3c 的各种标准当中，非常难找。但是我们要想找到一些具有完备性的线索，也不是没有办法。我喜欢的一个办法，就是用实际的代码去找：for in 遍历 window 的属性，再去找它的内容。

我想，学习的过程，实际上就是知识架构不断进化的过程，通过知识架构的自然延伸，我们可以更轻松地记忆一些原本难以记住的点，还可以发现被忽视的知识盲点。

建立知识架构，同样有利于面试，没人能够记住所有的知识，当不可避免地谈到一个记不住的知识，如果你能快速定位到它在知识架构中的位置，把一些相关的点讲出来，我想，这也能捞回不少分。（关于前端具体的知识架构，我会在 02 篇文章中详细讲解。）

第二个方法：追本溯源

第二个方法，我把它称作追本溯源。

有一些知识，背后有一个很大的体系，例如，我们对比一下 CSS 里面的两个属性：

opacity；

display。

虽然都是“属性”，但是它们背后的知识量完全不同，opacity 是个非常单纯的数值，表达的意思也很清楚，而 display 的每一个取值背后都是一个不同的布局体系。我们要讲清楚 display，就必须关注正常流（Normal Flow）、关注弹性布局系统以及 grid 这些内容。

还有一些知识，涉及的概念本身经历了各种变迁，变得非常复杂和有争议性，比如 MVC，从 1979 年至今，概念变化非常大，MVC 的定义几乎已经成了一段公案，我曾经截取了 MVC 原始论文、MVP 原始论文、微软 MSDN、Apple 开发者文档，这些内容里面，MVC 画的图、箭头和解释都完全不同。

这种时候，就是我们做一些考古工作的时候了。追本溯源，其实就是关注技术提出的背景，关注原始的论文或者文章，关注作者说的话。

操作起来也非常简单：翻翻资料（一般 wiki 上就有）找找历史上的文章和人物，再顺藤摸瓜翻出来历史资料就可以了，如果翻出来的是历史人物（幸亏互联网的历史不算悠久），你也可以试着发封邮件问问。

这个过程，可以帮助我们理解一些看上去不合理的东西，有时候还可以收获一些趣闻，比如 JavaScript 之父 Brendan Eich 曾经在 Wikipedia 的讨论页上解释 JavaScript 最初想设计一个带有 prototype 的 scheme，结果受到管理层命令把它弄成像 Java 的样子（如果你再挖的深一点，甚至能找到他对某位“尖头老板”的吐槽）。

根据这么一句话，我们再去看看 scheme，看看 Java，再看看一些别的基于原型的语言，我们就可以理解为什么 JavaScript 是现在这个样子了：函数是一等公民，却提供了 new this instanceof 等特性，甚至抄来了 Java 的 getYear 这样的 Bug。

结语

今天我带你探索了前端的学习路径，并提出了两个学习方法：你要试着建立自己的知识架构，除此之外，还要学会追本溯源，找到知识的源头。

这个专栏中，我并不奢望通过短短的 40 篇专栏，事无巨细地把前端的所有知识都罗列清楚，这本身是 MDN 这样的参考手册的工作。但是，我希望通过这个专栏，把前端技术背后的设计原理和知识体系讲清楚，让你能对前端技术产生整体认知，这样才能够在未来汹涌而来的新技术中保持领先的状态。

在你的认识中，前端知识的结构是怎样的？欢迎留言告诉我，我们一起讨论。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言(206)



米斯特菠萝

问题一提出来不知道怎么回答，看评论区才反应过来，我的认识中前端就是html负责骨架,css负责外表和js负责行为

2019-01-17



134



新哥

前端不是切页面的吗？

2019-01-17



2



71



CC

在阅读这篇文章之前，我对于前端知识结构的划分是基于「语言」：

前端 = HTML + CSS + JavaScript

每种语言下，又混杂了基础知识以及相关的工具。

比如我之前认为的 JavaScript 包括：

- * JavaScript 基础
- * Package Manager (NPM, Yarn)
- * Build tools (NPM Scripts, ESLint, Webpack...)
- * Frameworks (React, Vue.js, Angular)
- * Testing tools (Jest ...)

现在知道自己的认识有两个误区：

1. 这个知识架构没有完备性，这样划分，总有在这个结构之外的知识；
2. 学习的时候会疲于奔命，总想“完整”的学完所有的知识，却走了弯路。

感谢 winter 老师。

作者回复: 我觉得你不妨思考下，如果我要讲Package Manager、Build tools，你猜我会放到哪个部分？

2019-01-17

💬 2

👍 62



周群华

建议正式开始的时候能够图文并茂 而不只是纯粹的文字 还有音频和看文字区别不大

2019-01-17

💬 1

👍 55



芳玥

我所认为的前端的话，基础系列html+css+js，进阶一点就css3+js高级应用+前端框架+构建工具。工作快近2年，最近觉得焦虑。是自己太水，看看面试要求，反省自己，一看只会基础的但又不精。现在跟着课程体系，想建立好自己的知识体系，把握19年，不至于荒废时光。

2019-01-17

💬

👍 43





YUANWOW

大家都觉得HTML容易，非常简单
就算不会，坐下来花一天时间看看MDN几乎都能写出来了
我们都知道写HTML要语义化，但是HTML5的语义我感觉非常复杂
有多少人能保证写出来的网页完完全全遵循了正确的语义？
能达到屏幕阅读器理解的程度？
我们想写语义正确的网页，但是也不想被语义复杂性所拖累（本来读的也不是文科，div一分钟能写出来

的东西在语义上要纠结好久)

程老师，我的问题是，作为一名前端工程师，平常工作的时候写HTML究竟要语义化到哪种程度呢？

作者回复: 这是个好问题，在语义化部分我有讲这个问题，很快你就能看到了。

2019-01-17  1  42



Jerry

粗鄙认为前端知识架构可以分为渲染层和网络层两方面，不论前端如何发展变化，比如react，vue，ng，还是flutter，小程序等等，本质上脱离不了这两方面，另外前端工程师在接触一个新语言或新框架，最难不是学习语言本身（官方文档可以解决），反而棘手的是如何去工程化（更好组织代码，打包工程）和去优化，这是做前端两年的感受

作者回复: 渲染层和网络层这个分法听起来是有一定思考了，不过可以再想想，是不是还有别的层？

另外就是，如何定义渲染层呢？

2019-01-17   31



nbili

如果让自己描述前端的知识结构，大致会按照下面这个roadmap来。

<https://github.com/kamranahmedse/developer-roadmap/blob/master/readme.md>

winter老师第一种建立知识架构的方式，更像是做了进一步抽象。

作者回复: 要自己整理才行哦。

2019-01-17   23



把那个产品经理祭天

对于科班出身的学技术都是这一套思路吧！

感觉 winter 叔这篇主要是站在非科班出身或半路出家的前端从业者的角度来看的，确实我刚开始接触前端这一陌生领域来说就是这样，没有知识体系，都是知识点拼接的一张记忆网络，很多点觉得不可思

议，难以记忆。实际是上不了解本质，不了解来龙去脉，所以很多知识在脑海中就不够形象。
期待接下来 winter 的思路！

2019-01-17

  22



reece

- 1.1，代码打包，如何打包代码发送到服务器。
 - 1.1.1，资源加载
 - 1.1.2，webpack打包
 - 1.1.3，摇树优化

- 1.2，网络传输，因为传统上是浏览器发起页面请求，然后服务器端发送html页面。
 - 1.2.1，资源加载
 - 1.2.2，首屏页面加载优化

- 1.3，浏览器端的页面渲染。
 - 1.3.1，浏览器如何渲染一张html页面
 - 1.3.2，渲染性能优化

- 1.4代码编写
 - 1.4.1，html页面架构
 - 1.4.1.1
 - ...
 - 1.4.2，css页面样式
 - 1.4.2.1，
 -
 - 1.4.3，js交互
 - 1.4.3.1，
 -

作者回复: 不错，比较认真了，但是你有没有考虑过除了打包、传输还有什么？它们跟代码编写是一个维度的概念吗？

2019-01-17

  19



Jake

首先特别认同大佬现在的图形化学习的重要性，比如谷歌出的flutter，flutter的野心在于他想统一所有端的开发模式，他的核心思想就是抛开一切html, css, js, xml等的束缚(flutter也在计划支持web端了)，用户界面即画布，用户想要得到的视觉，画给用户就可以了，而现在无论是web端还是ios，安卓其实都只是图形学的其中一个分支，想要彻底攻克其本质，其实对于底层语言的学习，学习计算机绘图原理可

能是大前端的终极的学习目标。

前端发展到现在，无论是jquery时代，还是现在三大框架，还是说vue in all的时代，归根结底这些工具的产生都是为了解决工作效率等问题。结合现在互联网行业的发展来看，后期的前端生存之道可能更多的是思考如何把前端当成一个应用来做，不仅仅局限于网页。而应用可能要思考的是易用性，可扩展性，可维护性，甚至可移植性，那么这一切都需要夯实的基础来支撑。

2019-02-14



16



Destroy、

说实在，我现在知识体系都是乱的。都是东看一点西看一点拼凑起来的。

2019-01-17



13



棚头傀儡

我是黑马出身，emmm我的知识体系是初级前端要会哪些，中级要会哪些，又比如原型链和this之类的是中级必会的

2019-01-17



13



_1024

17年大学毕业之后开始接触前端，刚接触的时候觉得JavaScript特别简单，容易上手，但用了半年之后，发现不行了，就像 winter 老师所说的，经常会有“XX，还有这种用法”的感叹，于是，在 18 年，我自己把 JavaScript 的基础恶补了一遍，于是把一些概念开始聚合起来，有了一些比较清晰的脉络，学习的主要内容也是阮一峰老师的一些教程，在去年一年，也形成了自己的知识体系，有了一些落地的笔记，当然也还需要很多时间去修补

<https://docs.itellboy.wang/es5/>

共勉

2019-02-13



12



st

以前认知是html cds js,后来认识一些大佬，和他们同一个项目，日常的点滴，渐渐规范起来，才明白，这只是非常浅显的认知。

首先，前端项目的搭建要合理清晰，

其次，组件的复用性，最好的组件方式就是只负责UI层面的渲染，不涉及业务，业务抽离出来做成servi

ce等；而且组件最好细小化，只负责一小块，

然后，页面尽量语义化，scss,sass尽量少，能全局的放在全局，比如Mixin, normalize,当设计变来变去，也要和设计师沟通协商调整，不可能他今天一个样式，明天换一个，

接着，组件，service什么的写完了，确保自己代码测试覆盖率，记得写unit test，可以向别人下保票，自己写的99%是没问题的，如果别人想来challenger你代码，也会理直气壮有理有据。

尽量减少自己的代码量，html css js文件，
业界有很多成熟的框架，结合自己的业务，结合项目要求，能用的就用，时间精力思考广度，远远优于自己去手写一个，不然很容易会出现各种各样的问题和bug，

用别人的框架，有时间就了解一下原理，看看源码啥的，知其然也要知其所以然，面试也有帮助，加身功能理解，以后业务方面也会更清晰，

然后呢，肯定任何人写的代码都有值得优化的，要有这个思想存在，想着变得更好，

还有其他的细节吧，先这样啦，个人体会就是这样啦，也是在每天进步，观察，思考，实践，层层递进，

2019-04-21

💬 3

👍 11



大蓝

前端5年，正如文章讲浪费了学习的时间的小公司前端，学前端完全是野路子，前面三年都惧怕前端杂乱和混杂的方向，各种源码解读，各种原型解读，到最后都看不下去～没有清晰的方向，今年发觉并非自己笨，而是前端这条路子自己走的太坎坷了，没有名师，没有团队，凭个人智力 摸索打滚，也只是坐井观天

2019-01-21

💬

👍 8



远恒之义

刚看完了老师的直播，学到了很多的东西。
又重新复习了一遍今天的内容：学习路径和学习方法。

学习路径就是看书和建设性网站，购买老师的专栏，我觉得看不错的视频教程也是一种路径。

学习方法就是搭建知识架构和追本溯源。
老师讲解的学习方法都是三个维度（what、how、why）一步步由浅入深，从概念到方法再到收获，逻辑清晰同时具有完备性。

我是做 iOS 开发，知识架构是通过印象笔记中 iOS 功能知识点划分的笔记本加标签共同构建的目录和索引。学习了老师的方法之后，后面重新思考再梳理一下有逻辑具有完备性的知识体系吧。

2019-01-17



👍 8



岑中归月

工作一年多，我对前端的知识架构的认识就是：顶层目录html+css+javascript，二级目录是把这三个里面的知识点生硬并且毫无关联的列出来，三级目录是继续划分，，，，，看了老师的讲解，觉得这样划分的知识架构毫无逻辑可言，期待老师后面的课程。

2019-01-17



👍 8



王智

学习后端的我也来蹭点经验.无论是前端还是后端,我感觉学习方法都是相通的,甚至有时候我觉得前端的更新以及新技术的出现会更甚于后端,所以学习方法很有必要.我觉得老师说的两个方法都很好,追本溯源了解技术出现的背景和解决的问题,可以更快地了解这个技术以及进行学习,根据自己的问题来学习技术.同样,构建知识架构也是很有必要的,我一直想总结一下后端的知识体系,一直没有思路,等下篇文章看看前端的知识体系来总结一下吧. (◡‿◡)و

加油加油!!!

2019-01-17



👍 8



梁山伯与伏地魔

看了后 我心中浮现的结构大概如下吧：

- 1.html 骨架
- 2.css css3 外观
- 3.js 行为
- 4.浏览器（
 - 1.插件 草料二维码 vue dectools这类
 - 2.http协议
 - 3.工具 network console...
 - 4.window对象等
 - 5.兼容
 - 6.渲染模式）
- 5.工具（
 - 1.编辑器 webstorm vscode hbuilder
 - 2.git/webpack
- 6.框架（vue react...
- 7.插件库/ui库（axios moment 富文本编辑器 element antd

大概就是这个样子吧..
希望可以看到后面的课程取精华去糟粕有所进步

2019-01-18



6

