# [一名【合格】前端工程师的自检清单](https://segmentfault.com/a/1190000018992363)

* [javascript](https://segmentfault.com/t/javascript/blogs)

* [前端](https://segmentfault.com/t/%E5%89%8D%E7%AB%AF/blogs)

* [IMG_256 程序员](https://segmentfault.com/t/%E7%A8%8B%E5%BA%8F%E5%91%98/blogs)

* [知识体系](https://segmentfault.com/t/%E7%9F%A5%E8%AF%86%E4%BD%93%E7%B3%BB/blogs)

 7.2k 次阅读  ·  读完需要 47 分钟

556

## 开篇

前端开发是一个非常特殊的行业，它的历史实际上不是很长，但是知识之繁杂，技术迭代速度之快是其他技术所不能比拟的。

winter在他的《重学前端》课程中提到：

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。

这样是一个非常真实的现状，实际上很多前端开发者都是自学甚至转行过来的，前端入门简单，学习了几个API以后上手做项目也很简单，但是这往往成为了限制自身发展的瓶颈。

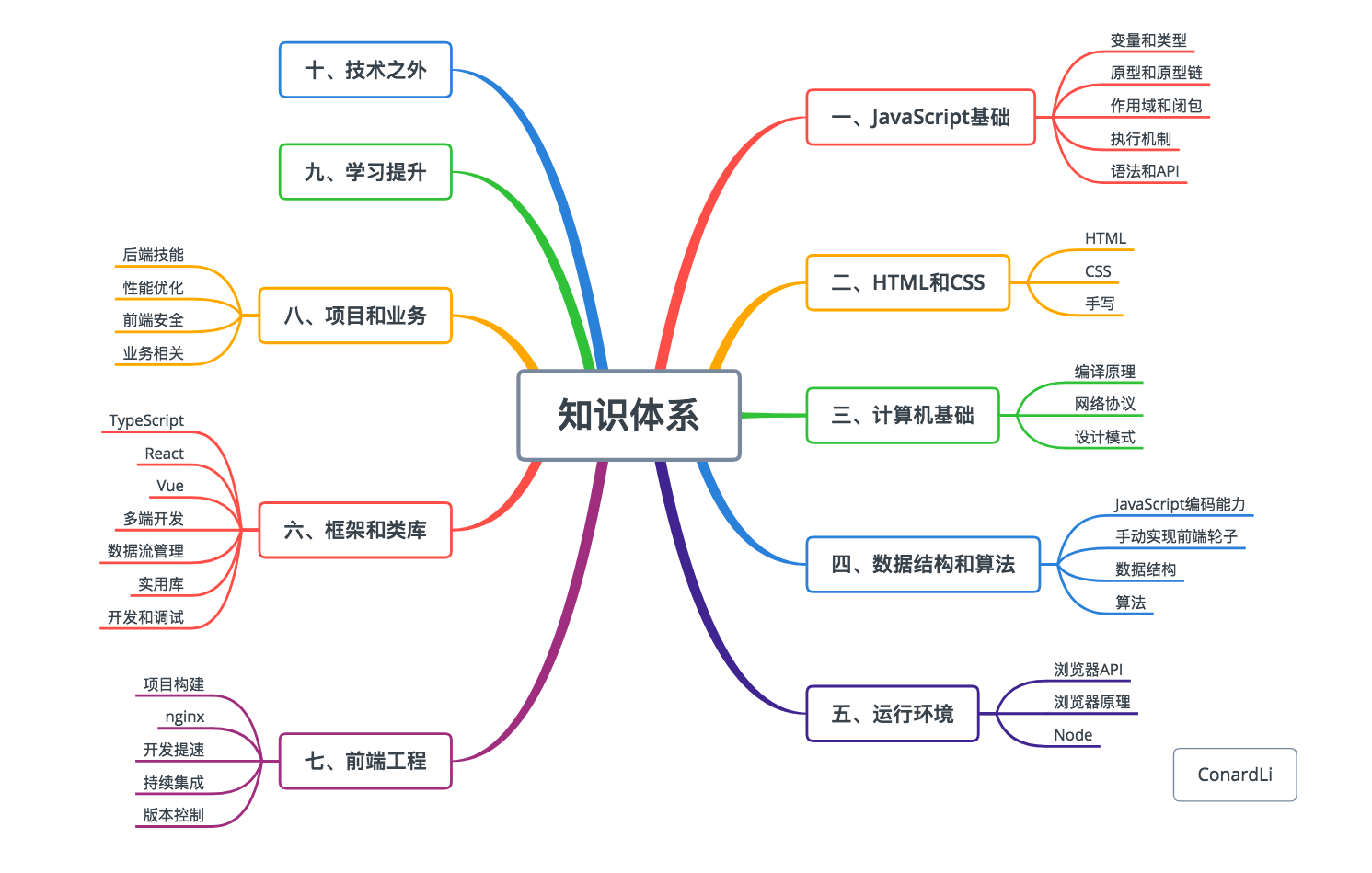
只是停留在会用阶段是远远不够的，我们还需要不断探索和深入。现在市面上并不缺少学习教程，技术文章，如果盲目的学习你会发现看过以后的知识留存率会很低，而且发现没有了解到的知识越来越多，这会让人产生焦虑。

实际上，除了坚持学习的强大的自驱力，你还需要一个很简单的学习方法。那就是：建立自己的知识体系。它能帮助你更系统性的学习，同时你也时刻能知道自己哪些地方是不足的。

我会把我工作和学习中接触到的知识全部归纳到我的知识体系中，其中不仅仅包括我已经学过的，还有很多我没有来得及学习的。

这不仅仅是我的知识体系，更是我时刻提醒自己的自检清单。

下面我会把我的自检清单分享给大家，你可以按照清单上的知识检测自己还有哪些不足和提升，我也建议大家建自己的知识体系，这样工作或者学习甚至面试时，你能快速定位到知识清单中的点，如果你有哪些我没归纳到的点，欢迎在评论区告诉我。



## 一、JavaScript基础

前端工程师吃饭的家伙，深度、广度一样都不能差。

### 变量和类型

* 1.JavaScript规定了几种语言类型
* 2.JavaScript对象的底层数据结构是什么
* 3.Symbol类型在实际开发中的应用、可手动实现一个简单的Symbol
* 4.JavaScript中的变量在内存中的具体存储形式
* 5.基本类型对应的内置对象，以及他们之间的装箱拆箱操作
* 6.理解值类型和引用类型
* 7.null和undefined的区别
* 8.至少可以说出三种判断JavaScript数据类型的方式，以及他们的优缺点，如何准确的判断数组类型
* 9.可能发生隐式类型转换的场景以及转换原则，应如何避免或巧妙应用
* 10.出现小数精度丢失的原因，JavaScript可以存储的最大数字、最大安全数字，JavaScript处理大数字的方法、避免精度丢失的方法

### 原型和原型链

* 1.理解原型设计模式以及JavaScript中的原型规则
* 2.instanceof的底层实现原理，手动实现一个instanceof
* 4.实现继承的几种方式以及他们的优缺点
* 5.至少说出一种开源项目(如Node)中应用原型继承的案例
* 6.可以描述new一个对象的详细过程，手动实现一个new操作符
* 7.理解es6 class构造以及继承的底层实现原理

### 作用域和闭包

* 1.理解词法作用域和动态作用域
* 2.理解JavaScript的作用域和作用域链
* 3.理解JavaScript的执行上下文栈，可以应用堆栈信息快速定位问题
* 4.this的原理以及几种不同使用场景的取值
* 5.闭包的实现原理和作用，可以列举几个开发中闭包的实际应用
* 6.理解堆栈溢出和内存泄漏的原理，如何防止
* 7.如何处理循环的异步操作
* 8.理解模块化解决的实际问题，可列举几个模块化方案并理解其中原理

### 执行机制

* 1.为何try里面放return，finally还会执行，理解其内部机制
* 2.JavaScript如何实现异步编程，可以详细描述EventLoop机制
* 3.宏任务和微任务分别有哪些
* 4.可以快速分析一个复杂的异步嵌套逻辑，并掌握分析方法
* 5.使用Promise实现串行
* 6.Node与浏览器EventLoop的差异
* 7.如何在保证页面运行流畅的情况下处理海量数据

### 语法和API

* 1.理解ECMAScript和JavaScript的关系
* 2.熟练运用es5、es6提供的语法规范，
* 3.熟练掌握JavaScript提供的全局对象（例如Date、Math）、全局函数（例如decodeURI、isNaN）、全局属性（例如Infinity、undefined）
* 4.熟练应用map、reduce、filter 等高阶函数解决问题
* 5.setInterval需要注意的点，使用settimeout实现setInterval
* 6.JavaScript提供的正则表达式API、可以使用正则表达式（邮箱校验、URL解析、去重等）解决常见问题
* 7.JavaScript异常处理的方式，统一的异常处理方案

## 二、HTML和CSS

### HTML

* 1.从规范的角度理解HTML，从分类和语义的角度使用标签
* 2.常用页面标签的默认样式、自带属性、不同浏览器的差异、处理浏览器兼容问题的方式
* 3.元信息类标签(head、title、meta)的使用目的和配置方法
* 4.HTML5离线缓存原理
* 5.可以使用Canvas API、SVG等绘制高性能的动画

### CSS

* 1.CSS盒模型，在不同浏览器的差异
* 2.CSS所有选择器及其优先级、使用场景，哪些可以继承，如何运用at规则
* 3.CSS伪类和伪元素有哪些，它们的区别和实际应用
* 4.HTML文档流的排版规则，CSS几种定位的规则、定位参照物、对文档流的影响，如何选择最好的定位方式，雪碧图实现原理
* 5.水平垂直居中的方案、可以实现6种以上并对比它们的优缺点
* 6.BFC实现原理，可以解决的问题，如何创建BFC
* 7.可使用CSS函数复用代码，实现特殊效果
* 8.PostCSS、Sass、Less的异同，以及使用配置，至少掌握一种
* 9.CSS模块化方案、如何配置按需加载、如何防止CSS阻塞渲染
* 10.熟练使用CSS实现常见动画，如渐变、移动、旋转、缩放等等
* 11.CSS浏览器兼容性写法，了解不同API在不同浏览器下的兼容性情况
* 12.掌握一套完整的响应式布局方案

### 手写

* 1.手写图片瀑布流效果
* 2.使用CSS绘制几何图形（圆形、三角形、扇形、菱形等）
* 3.使用纯CSS实现曲线运动（贝塞尔曲线）
* 4.实现常用布局（三栏、圣杯、双飞翼、吸顶），可是说出多种方式并理解其优缺点

## 三、计算机基础

关于编译原理，不需要理解非常深入，但是最基本的原理和概念一定要懂，这对于学习一门编程语言非常重要

### 编译原理

* 1.理解代码到底是什么，计算机如何将代码转换为可以运行的目标程序
* 2.正则表达式的匹配原理和性能优化
* 3.如何将JavaScript代码解析成抽象语法树(AST)
* 4.base64的编码原理
* 5.几种进制的相互转换计算方法，在JavaScript中如何表示和转换

### 网络协议

* 1.理解什么是协议，了解TCP/IP网络协议族的构成，每层协议在应用程序中发挥的作用
* 2.三次握手和四次挥手详细原理，为什么要使用这种机制
* 3.有哪些协议是可靠，TCP有哪些手段保证可靠交付
* 4.DNS的作用、DNS解析的详细过程，DNS优化原理
* 5.CDN的作用和原理
* 6.HTTP请求报文和响应报文的具体组成，能理解常见请求头的含义，有几种请求方式，区别是什么
* 7.HTTP所有状态码的具体含义，看到异常状态码能快速定位问题
* 8.HTTP1.1、HTTP2.0带来的改变
* 9.HTTPS的加密原理，如何开启HTTPS，如何劫持HTTPS请求
* 10.理解WebSocket协议的底层原理、与HTTP的区别

### 设计模式

* 1.熟练使用前端常用的设计模式编写代码，如单例模式、装饰器模式、代理模式等
* 2.发布订阅模式和观察者模式的异同以及实际应用
* 3.可以说出几种设计模式在开发中的实际应用，理解框架源码中对设计模式的应用

## 四、数据结构和算法

据我了解的大部分前端对这部分知识有些欠缺，甚至抵触，但是，如果突破更高的天花板，这部分知识是必不可少的，而且我亲身经历——非常有用！

### JavaScript编码能力

* 1.多种方式实现数组去重、扁平化、对比优缺点
* 2.多种方式实现深拷贝、对比优缺点
* 3.手写函数柯里化工具函数、并理解其应用场景和优势
* 4.手写防抖和节流工具函数、并理解其内部原理和应用场景
* 5.实现一个sleep函数

### 手动实现前端轮子

* 1.手动实现call、apply、bind
* 2.手动实现符合Promise/A+规范的Promise、手动实现async await
* 3.手写一个EventEmitter实现事件发布、订阅
* 4.可以说出两种实现双向绑定的方案、可以手动实现
* 5.手写JSON.stringify、JSON.parse
* 6.手写一个模版引擎，并能解释其中原理
* 7.手写懒加载、下拉刷新、上拉加载、预加载等效果

### 数据结构

* 1.理解常见数据结构的特点，以及他们在不同场景下使用的优缺点
* 2.理解数组、字符串的存储原理，并熟练应用他们解决问题
* 3.理解二叉树、栈、队列、哈希表的基本结构和特点，并可以应用它解决问题
* 4.了解图、堆的基本结构和使用场景

### 算法

* 1.可计算一个算法的时间复杂度和空间复杂度，可估计业务逻辑代码的耗时和内存消耗
* 2.至少理解五种排序算法的实现原理、应用场景、优缺点，可快速说出时间、空间复杂度
* 3.了解递归和循环的优缺点、应用场景、并可在开发中熟练应用
* 4.可应用回溯算法、贪心算法、分治算法、动态规划等解决复杂问题
* 5.前端处理海量数据的算法方案

## 五、运行环境

我们需要理清语言和环境的关系：

ECMAScript描述了JavaScript语言的语法和基本对象规范

浏览器作为JavaScript的一种运行环境，为它提供了：文档对象模型（DOM），描述处理网页内容的方法和接口、浏览器对象模型（BOM），描述与浏览器进行交互的方法和接口

Node也是JavaScript的一种运行环境，为它提供了操作I/O、网络等API

### 浏览器API

* 1.浏览器提供的符合W3C标准的DOM操作API、浏览器差异、兼容性
* 2.浏览器提供的浏览器对象模型 (BOM)提供的所有全局API、浏览器差异、兼容性
* 3.大量DOM操作、海量数据的性能优化(合并操作、Diff、requestAnimationFrame等)
* 4.浏览器海量数据存储、操作性能优化
* 5.DOM事件流的具体实现机制、不同浏览器的差异、事件代理
* 6.前端发起网络请求的几种方式及其底层实现、可以手写原生ajax、fetch、可以熟练使用第三方库
* 7.浏览器的同源策略，如何避免同源策略，几种方式的异同点以及如何选型
* 8.浏览器提供的几种存储机制、优缺点、开发中正确的选择
* 9.浏览器跨标签通信

### 浏览器原理

* 1.各浏览器使用的JavaScript引擎以及它们的异同点、如何在代码中进行区分
* 2.请求数据到请求结束与服务器进行了几次交互
* 3.可详细描述浏览器从输入URL到页面展现的详细过程
* 4.浏览器解析HTML代码的原理，以及构建DOM树的流程
* 5.浏览器如何解析CSS规则，并将其应用到DOM树上
* 6.浏览器如何将解析好的带有样式的DOM树进行绘制
* 7.浏览器的运行机制，如何配置资源异步同步加载
* 8.浏览器回流与重绘的底层原理，引发原因，如何有效避免
* 9.浏览器的垃圾回收机制，如何避免内存泄漏
* 10.浏览器采用的缓存方案，如何选择和控制合适的缓存方案

### Node

* 1.理解Node在应用程序中的作用，可以使用Node搭建前端运行环境、使用Node操作文件、操作数据库等等
* 2.掌握一种Node开发框架，如Express，Express和Koa的区别
* 3.熟练使用Node提供的API如Path、Http、Child Process等并理解其实现原理
* 4.Node的底层运行原理、和浏览器的异同
* 5.Node事件驱动、非阻塞机制的实现原理

## 六、框架和类库

轮子层出不穷，从原理上理解才是正道

### TypeScript

* 1.理解泛型、接口等面向对象的相关概念，TypeScript对面向对象理念的实现
* 2.理解使用TypeScript的好处，掌握TypeScript基础语法
* 3.TypeScript的规则检测原理
* 4.可以在React、Vue等框架中使用TypeScript进行开发

### React

* 1.React和vue 选型和优缺点、核心架构的区别
* 2.React中setState的执行机制，如何有效的管理状态
* 3.React的事件底层实现机制
* 4.React的虚拟DOM和Diff算法的内部实现
* 5.React的Fiber工作原理，解决了什么问题
* 6.React Router和Vue Router的底层实现原理、动态加载实现原理
* 7.可熟练应用React API、生命周期等，可应用HOC、render props、Hooks等高阶用法解决问题
* 8.基于React的特性和原理，可以手动实现一个简单的React

### Vue

* 1.熟练使用Vue的API、生命周期、钩子函数
* 2.MVVM框架设计理念
* 3.Vue双向绑定实现原理、Diff算法的内部实现
* 4.Vue的事件机制
* 5.从template转换成真实DOM的实现机制

### 多端开发

* 1.单页面应用（SPA）的原理和优缺点，掌握一种快速开发SPA的方案
* 2.理解Viewport、em、rem的原理和用法，分辨率、px、ppi、dpi、dp的区别和实际应用
* 3.移动端页面适配解决方案、不同机型适配方案
* 4.掌握一种JavaScript移动客户端开发技术，如React Native：可以搭建React Native开发环境，熟练进行开发，可理解React Native的运作原理，不同端适配
* 5.掌握一种JavaScript PC客户端开发技术，如Electron：可搭建Electron开发环境，熟练进行开发，可理解Electron的运作原理
* 6.掌握一种小程序开发框架或原生小程序开发
* 7.理解多端框架的内部实现原理，至少了解一个多端框架的使用

### 数据流管理

* 1.掌握React和Vue传统的跨组件通信方案，对比采用数据流管理框架的异同
* 2.熟练使用Redux管理数据流，并理解其实现原理，中间件实现原理
* 3.熟练使用Mobx管理数据流，并理解其实现原理，相比Redux有什么优势
* 4.熟练使用Vuex管理数据流，并理解其实现原理
* 5.以上数据流方案的异同和优缺点，不情况下的技术选型

### 实用库

* 1.至少掌握一种UI组件框架，如antd design，理解其设计理念、底层实现
* 2.掌握一种图表绘制框架，如Echart，理解其设计理念、底层实现，可以自己实现图表
* 3.掌握一种GIS开发框架，如百度地图API
* 4.掌握一种可视化开发框架，如Three.js、D3
* 5.工具函数库，如lodash、underscore、moment等，理解使用的工具类或工具函数的具体实现原理

### 开发和调试

* 1.熟练使用各浏览器提供的调试工具
* 2.熟练使用一种代理工具实现请求代理、抓包，如charls
* 3.可以使用Android、IOS模拟器进行调试，并掌握一种真机调试方案
* 4.了解Vue、React等框架调试工具的使用

## 七、前端工程

前端工程化：以工程化方法和工具提高开发生产效率、降低维护难度

### 项目构建

* 1.理解npm、yarn依赖包管理的原理，两者的区别
* 2.可以使用npm运行自定义脚本
* 3.理解Babel、ESLint、webpack等工具在项目中承担的作用
* 4.ESLint规则检测原理，常用的ESLint配置
* 5.Babel的核心原理，可以自己编写一个Babel插件
* 6.可以配置一种前端代码兼容方案，如Polyfill
* 7.Webpack的编译原理、构建流程、热更新原理，chunk、bundle和module的区别和应用
* 8.可熟练配置已有的loaders和plugins解决问题，可以自己编写loaders和plugins

### nginx

* 1.正向代理与反向代理的特点和实例
* 2.可手动搭建一个简单的nginx服务器、
* 3.熟练应用常用的nginx内置变量，掌握常用的匹配规则写法
* 4.可以用nginx实现请求过滤、配置gzip、负载均衡等，并能解释其内部原理

### 开发提速

* 1.熟练掌握一种接口管理、接口mock工具的使用，如yapi
* 2.掌握一种高效的日志埋点方案，可快速使用日志查询工具定位线上问题
* 3.理解TDD与BDD模式，至少会使用一种前端单元测试框架

### 版本控制

* 1.理解Git的核心原理、工作流程、和SVN的区别
* 2.熟练使用常规的Git命令、git rebase、git stash等进阶命令
* 3.可以快速解决线上分支回滚、线上分支错误合并等复杂问题

### 持续集成

* 1.理解CI/CD技术的意义，至少熟练掌握一种CI/CD工具的使用，如Jenkins
* 2.可以独自完成架构设计、技术选型、环境搭建、全流程开发、部署上线等一套完整的开发流程（包括Web应用、移动客户端应用、PC客户端应用、小程序、H5等等）

## 八、项目和业务

### 后端技能

* 1.了解后端的开发方式，在应用程序中的作用，至少会使用一种后端语言
* 2.掌握数据最终在数据库中是如何落地存储的，能看懂表结构设计、表之间的关联，至少会使用一种数据库

### 性能优化

* 1.了解前端性能衡量指标、性能监控要点，掌握一种前端性能监控方案
* 2.了解常见的Web、App性能优化方案
* 3.SEO排名规则、SEO优化方案、前后端分离的SEO
* 4.SSR实现方案、优缺点、及其性能优化
* 5.Webpack的性能优化方案
* 6.Canvas性能优化方案
* 7.React、Vue等框架使用性能优化方案

### 前端安全

* 1.XSS攻击的原理、分类、具体案例，前端如何防御
* 2.CSRF攻击的原理、具体案例，前端如何防御
* 3.HTTP劫持、页面劫持的原理、防御措施

### 业务相关

* 1.能理解所开发项目的整体业务形态、业务目标、业务架构，可以快速定位线上业务问题
* 2.能理解所开发项目整体的技术架构、能快读的根据新需求进行开发规划、能快速根据业务报警、线上日志等定位并解决线上技术问题
* 3.可以将自己的想法或新技术在业务中落地实践，尽量在团队中拥有一定的不可替代性

## 九、学习提升

vczh大神在知乎问题【如何能以后达到温赵轮三位大神的水平？】下的回答：

这十几年我一共做了三件事：

* 1、不以赚钱为目的选择学习的内容；
* 2、以自己是否能造出轮子来衡量学习的效果；
* 3、坚持每天写自己的代码，前10年每天至少6个小时，不包含学习和工作的时间。

上面几点可能有点难，第一点我就做不到，但是做到下面绩点还是比较容易的。

关于写博客说明下，能给别人讲明白的知识会比自己学习掌握的要深刻许多

* 1.拥有自己的技术博客，或者在一些博客平台上拥有自己的专栏
* 2.定期的将知识进行总结，不断完善自己的知识体系
* 3.尽量将自己的知识转换成真实的产出，不要仅仅停留在书面理解层面，更重要的是实际应用
* 4.坚持输出自己的代码，不要盲目的扎进公司业

## 十、技术之外

这部分可能比上面九条加起来重要！

* 1.了解互联网人员术语：CEO、CTO、COO、CFO、PM、QA、UI、FE、DEV、DBA、OPS等
* 2.了解互联网行业术语：B2B、B2C、C2C、O2O等
* 3.掌握互联网行业沟通、问答、学习的
* 4.有一定的"PPT"能力
* 5.有一定的理财意识，至少了解储蓄、货币基金、保险、指数基金、股票等基本的理财知识
* 6.掌握在繁重的工作和长期的电脑辐射的情况下保持健康的方法，建立正确的养生知识体系

## 十一、资源推荐

有了知识体系，在阅读一篇技术文章的时候就很容易把它归类，我一直以来就是这样做的。

事实证明，在阅读文章或书籍时，有目的和归类的阅读比"随便看看"后的只是留存率要高很多。

每阅读到一篇好的文章或者书籍，我都会收藏并归类到我的知识体系中。

下面是一些我觉得还不错的文章、博客或者书籍教程等等，分享给大家，资源不多，但都是精品。

学习一门知识，最好先阅读官方文档，把所有的API大概浏览一遍，再继续看大佬们总结的进阶知识，什么东西是搬运过来的，什么是干货，一目了然。

### 语言基础

* [JavaScript高级程序设计（必看）：[https://book.douban.com/subje...](https://book.douban.com/subject/10546125/" \t "https://segmentfault.com/a/_blank)
* [JavaScript：[https://book.douban.com/subje...](https://book.douban.com/subject/5362856/" \t "https://segmentfault.com/a/_blank)
* 现代JavaScript教程：[https://zh.javascript.info/](https://zh.javascript.info/" \t "https://segmentfault.com/a/_blank)
* 阮一峰的ECMAScript 6教程：[http://es6.ruanyifeng.com/](http://es6.ruanyifeng.com/" \t "https://segmentfault.com/a/_blank)
* ECMAScript 6标准：[https://www.ecma-internationa...](https://www.ecma-international.org/ecma-262/6.0/" \t "https://segmentfault.com/a/_blank)
* HTML meta标签总结与属性使用介绍：[https://segmentfault.com/a/11...](https://segmentfault.com/a/1190000004279791)
* CSS编码指导：[https://github.com/chadluo/CS...](https://github.com/chadluo/CSS-Guidelines/blob/master/README.md" \t "https://segmentfault.com/a/_blank)

### 计算机基础

* 大前端开发者需要了解的基础编译原理和语言知识：[http://fullstack.blog/2017/06...](http://fullstack.blog/2017/06/24/%E5%A4%A7%E5%89%8D%E7%AB%AF%E5%BC%80%E5%8F%91%E8%80%85%E9%9C%80%E8%A6%81%E4%BA%86%E8%A7%A3%E7%9A%84%E5%9F%BA%E7%A1%80%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86%E5%92%8C%E8%AF%AD%E8%A8%80%E7%9F%A5%E8%AF%86" \t "https://segmentfault.com/a/_blank)
* 图解HTTP：[https://book.douban.com/subje...](https://book.douban.com/subject/25863515/" \t "https://segmentfault.com/a/_blank)
* [JavaScript设计模式与开发实践：[https://book.douban.com/subje...](https://book.douban.com/subject/26382780/" \t "https://segmentfault.com/a/_blank)
* 正则表达式30分钟入门教程：[https://link.juejin.im/?targe...](https://link.juejin.im/?target=https://deerchao.net/tutorials/regex/regex.htm" \t "https://segmentfault.com/a/_blank)

### 数据结构和算法

* 数据结构与算法之美：[https://time.geekbang.org/col...](https://time.geekbang.org/column/intro/126" \t "https://segmentfault.com/a/_blank)
* 用动画的形式呈现解LeetCode题目的思路：[https://github.com/MisterBooo...](https://github.com/MisterBooo/LeetCodeAnimation" \t "https://segmentfault.com/a/_blank)
* JavaScript数据结构和算法：[https://github.com/ConardLi/a...](https://github.com/ConardLi/awesome-coding-js" \t "https://segmentfault.com/a/_blank)
* 30-seconds-of-code（里面有很多js代码非常巧妙，我正在将它翻译成中文）：[https://github.com/ConardLi/3...](https://github.com/ConardLi/30-seconds-of-code-Zh-CN" \t "https://segmentfault.com/a/_blank)

### 运行环境

* 《重学前端》中的浏览器原理章节：[https://time.geekbang.org/col...](https://time.geekbang.org/column/article/80240" \t "https://segmentfault.com/a/_blank)
* 图解浏览器的基本工作原理：[https://zhuanlan.zhihu.com/p/...](https://zhuanlan.zhihu.com/p/47407398" \t "https://segmentfault.com/a/_blank)
* 七天学会NodeJS：[https://github.com/nqdeng/7-d...](https://github.com/nqdeng/7-days-nodejs" \t "https://segmentfault.com/a/_blank)
* Node.js模块加载与运行原理：[https://efe.baidu.com/blog/no...](https://efe.baidu.com/blog/nodejs-module-analyze/" \t "https://segmentfault.com/a/_blank)

### 框架和类库

* TypeScript Handbook：[https://zhongsp.gitbooks.io/t...](https://zhongsp.gitbooks.io/typescript-handbook/content/" \t "https://segmentfault.com/a/_blank)
* React.js小书：[http://huziketang.mangojuice....](http://huziketang.mangojuice.top/books/react/" \t "https://segmentfault.com/a/_blank)
* React深入系列：[https://juejin.im/post/5cad39...](https://juejin.im/post/5cad39b3f265da03502b1c0a" \t "https://segmentfault.com/a/_blank)
* Webpack React小书：[https://fakefish.github.io/re...](https://fakefish.github.io/react-webpack-cookbook/index.html" \t "https://segmentfault.com/a/_blank)
* Vue.js技术揭秘：[https://github.com/ustbhuangy...](https://github.com/ustbhuangyi/vue-analysis" \t "https://segmentfault.com/a/_blank)
* Vuex-在Vue中管理状态：[https://sabe.io/tutorials/get...](https://sabe.io/tutorials/getting-started-with-vuex" \t "https://segmentfault.com/a/_blank)
* 你需要Mobx还是Redux？：[https://juejin.im/post/5a7fd7...](https://juejin.im/post/5a7fd72c5188257a766324ae" \t "https://segmentfault.com/a/_blank)
* Underscore源码分析：[https://yoyoyohamapi.gitbooks...](https://yoyoyohamapi.gitbooks.io/undersercore-analysis/content/" \t "https://segmentfault.com/a/_blank)
* 微信小程序开发资源汇总：[https://github.com/justjavac/...](https://github.com/justjavac/awesome-wechat-weapp" \t "https://segmentfault.com/a/_blank)
* 腾讯移动 Web 前端知识库：[https://github.com/AlloyTeam/...](https://github.com/AlloyTeam/Mars" \t "https://segmentfault.com/a/_blank)

### 前端工程

* 一口（很长的）气了解babel：[https://zhuanlan.zhihu.com/p/...](https://zhuanlan.zhihu.com/p/43249121" \t "https://segmentfault.com/a/_blank)
* Webpack傻瓜式指南：[https://zhuanlan.zhihu.com/p/...](https://zhuanlan.zhihu.com/p/20367175" \t "https://segmentfault.com/a/_blank)
* Webpack原理：[https://segmentfault.com/a/11...](https://segmentfault.com/a/1190000015088834?utm_source=tag-newest)
* 廖雪峰的git教程：[https://www.liaoxuefeng.com/w...](https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000" \t "https://segmentfault.com/a/_blank)
* 图解Git：[https://marklodato.github.io/...](https://marklodato.github.io/visual-git-guide/index-zh-cn.html" \t "https://segmentfault.com/a/_blank)
* 前端开发者必备的Nginx知识：[https://juejin.im/post/5c85a6...](https://juejin.im/post/5c85a64d6fb9a04a0e2e038c" \t "https://segmentfault.com/a/_blank)
* 使用Jenkins进行持续集成：[https://www.liaoxuefeng.com/a...](https://www.liaoxuefeng.com/article/001463233913442cdb2d1bd1b1b42e3b0b29eb1ba736c5e000" \t "https://segmentfault.com/a/_blank)

### 项目和业务

* 常见六大Web安全攻防解析：[https://github.com/ljianshu/B...](https://github.com/ljianshu/Blog/issues/56" \t "https://segmentfault.com/a/_blank)
* 深入理解前端性能监控：[https://juejin.im/post/5caaac...](https://juejin.im/post/5caaacc0e51d452b45296487" \l "heading-5" \t "https://segmentfault.com/a/_blank)
* [[https://book.douban.com/subje...](https://book.douban.com/subject/3132277/" \t "https://segmentfault.com/a/_blank)
* 新人如何快速融入技术实力强的前端团队：[https://juejin.im/post/5cb860...](https://juejin.im/post/5cb860a86fb9a06890705f14" \t "https://segmentfault.com/a/_blank)

### 学习提升

* 印记中文（各种中文开发文档）：[https://www.docschina.org/](https://www.docschina.org/" \t "https://segmentfault.com/a/_blank)
* 前端学习方法：[https://github.com/helloqingf...](https://github.com/helloqingfeng/Awsome-Front-End-learning-resource/tree/master/01-FE-learning-master" \t "https://segmentfault.com/a/_blank)
* 如何在工作内外获得持续的技术成长：[https://juejin.im/post/5cbd74...](https://juejin.im/post/5cbd7477f265da039d32834e" \t "https://segmentfault.com/a/_blank)
* 优秀的前端博客汇总：[https://github.com/foru17/fro...](https://github.com/foru17/front-end-collect" \t "https://segmentfault.com/a/_blank)

另外推荐我一直在关注的几位大佬的个人博客：

* 冴羽的博客：[https://github.com/mqyqingfen...](https://github.com/mqyqingfeng/Blog" \t "https://segmentfault.com/a/_blank)
* 张鑫旭的博客：[https://www.zhangxinxu.com/wo...](https://www.zhangxinxu.com/wordpress/" \t "https://segmentfault.com/a/_blank)
* 左耳朵耗子：[https://coolshell.cn/](https://coolshell.cn/" \t "https://segmentfault.com/a/_blank)

### 技术之外

* 互联网术语大全：[https://www.jianshu.com/p/9a7...](https://www.jianshu.com/p/9a7ca206c1ab" \t "https://segmentfault.com/a/_blank)
* 互联网沟通、问答、学习的艺术：[https://zhuanlan.zhihu.com/p/...](https://zhuanlan.zhihu.com/p/41431775" \t "https://segmentfault.com/a/_blank)
* 经常加班至深夜，怎样才能保持身体健康：[https://www.zhihu.com/questio...](https://www.zhihu.com/question/21790919" \t "https://segmentfault.com/a/_blank)

其实在这个信息发达的时代最不缺的就是资源，如何从众多的资源中获取到真正精华的部分，是非常重要的，资源在于精不在于多，强烈建议在保证深度的情况下再保证广度。

## 小结

希望你阅读本篇文章后可以达到以下几点：

* 从知识清单中找到自己的知识盲点与欠缺
* 具有知识体系化的思想，开始建立自己的知识体系
* 阅读文章时将知识归类到知识体系中，并不断完善自己的知识体系
* 从文章中获取到了有用的资源

文中如有错误，欢迎在评论区指正，如果这篇文章帮助到了你，欢迎点赞和关注。

如果你有什么好的知识、资源推荐，欢迎在评论区留言。

想阅读更多优质文章、下载文章中思维导图源文件、可关注我的[github博客](https://github.com/ConardLi/ConardLi.github.io" \t "https://segmentfault.com/a/_blank)，后续的文章我也会按照知识清单来写，你的star✨、点赞和关注是我持续创作的动力！

推荐关注我的微信公众号【code秘密花园】，每天推送高质量文章，我们一起交流成长。