学习计划

1. 评卷反馈

- 学员的前端知识结构掌握良好，对于常见知识点有一定的了解，但是存在回答不够全面的问题，例如 src 和 href 的区别，script 标签区别没有回答出执行顺序等。

这方面建议学员多花时间去总结完善自己的知识体系，这一块可以多从别人总结的文章或面经入手，梳理自己的知识结构图谱。

- 框架的使用良好，但是原理层面的理解有欠缺。例如双向绑定、nextTick 的原理部分回答的不是很好。建议后续可以跟着视频和补充资料加强源码的学习，多梳理总结，多多融合自己实践的理解，并利用源码的特性进行实践封装。
- 工程化方面同样需要适当的实践，对于 webpack、babel 等常见工具能有一定的实践与理论的掌握
- 计算机网络和算法相关的知识点也需要适当掌握

学习目标优先级

以下为课程所涉及到的前端知识，根据学员自身实际情况作出优先级目标。

其中高优先级为必须花时间和精力掌握；中优先级可结合自己的实际情况，掌握至少 60% 的内容；低优先级按自身时间安排看是否有经历去掌握。

优先级	模块	相关课程
高	JS 基础	《Promise 规范及应用》、《JS 模块化详解》、《浏览器系列》、《ES6 系列课程》、《面向对象编程/原型及原型链》、

		《this 指针/闭包/作用域》
高	框架介绍与应用	Vue：《Vue.js 基础》、《Vue.js 高级用法》、《、《Vue.js 项目实战》等系列课程 React：《React 基础》、《React 高级用法》、《React 组件库设计》、《React 实战-仿知乎》等系列课程
高	框架源码	Vue：《Vue.js 核心模块源码讲解》、《Vue-Cli 详解》、《Vue.js 前端路由及异步组件》等系列课程 React：《React 核心源码解析》
高	工程化	《前端工程化详解》、《ESNext 规范及编译工具简介》《模块化开发与 Webpack》、《前端 AST》、《自动化构建、测试、部署》
中	设计模式	《设计模式解析与实战》
中	TypeScript	《TS 应用》
高	计算机网络	《网络 HTTP》

冲击大厂必备	算法	《算法问题详解》
考虑是否可以融合进自己的项目当中	解决方案与应用	《大厂常见问题与解决方案》、《JS 开发常见优化总结》
中（根据自己实际掌握情况选择，作为加分项，不强求）	前端可视化	《可视化》
	React-Native	《React-Native 入门与原理介绍》、《React-Native 实战》
	Flutter	《flutter 与 dart 开发入门》、《flutter 实战》
	Node.js	《Node.js 基础教学》、《Node.js 原理详解》、《Node.js 实战》等系列课程

2. 前置准备工作

2.1 知识框架整理

在复习之前，可以将学习模块划分为大致以下类型，建议把大致的时间作为一下拆分，例如每个模块需要花多长时间。**这块根据自己的学习情况合理划分，对于薄弱项建议多花一些时间**。结合学员的情况，以下给出参考的时间占比分配：

- 基础知识（HTML、CSS、JS）30%
- 框架应用与原理 40%
- 工程化方向（webpack、babel 等） 10%
- 其余零碎知识点（计算机网络、算法）10%
- 项目梳理（优化等亮点）10%

学员需要再针对每个模块再进行细分，可以参考这篇文章，文章中将每个模块涉及到的知识点

以图谱的方式罗列出来，清楚每个模块大致包含的知识点。学员可以参考这种方式来组织自己的架构。

列出架构后，按学习的模块像给项目排期一样**预计自己会花多少时间**，比如异步相关的知识 2 天，Vue 源码 3 天这样。

2.2 复习思路

复习过程中尽量做到彻底吃透一个知识点，知识点的相关资料可以参考「学习资料」章节的补充材料链接。每个知识点最好能够整理出自己的思路，例如概念、作用、优缺点、实际应用等等。

在时间紧张的情况，可以用刷题的方式来快速记忆。相关的题目在「学习资料」章节也有给出链接。

2.3 成果检验

当完成一个模块的复习之后，可以尝试去刷该模块的面试题来检验自己的梳理学习结果。要求能够做到能够**看到概念后完整阐述出这个知识点的所有内容**。例如什么是闭包，可以从闭包的概念，它的作用，闭包可能会引起什么样的问题，以及它在实际中的应用场景是什么这些角度来阐述。也可以由此引申到浏览器垃圾回收机制的概念，这样也有助于将知识点串联在一起记忆。

2.4 冲刺刷题

复习的最后阶段，建议留一点时间给自己去刷面试题，如果那些所谓的大厂面试题你能回答出60%以上（冲刺大厂的话至少 70%、80%以上），再加上你的项目有亮点，那整体来说是比较稳的。

综合面试题

https://juejin.im/post/5e4c0b856fb9a07ccb7e8eca

https://juejin.im/post/6844903776512393224

https://segmentfault.com/a/1190000022164672

https://juejin.im/post/5e8b261ae51d4546c0382ab4

https://juejin.im/post/5d690c726fb9a06b155dd40d

面试分享：两年工作经验成功面试阿里 P6 总结

中高级前端大厂面试秘籍，为你保驾护航金三银四，直通大厂

整理的过程中需要明确自己的目标，如果是想要冲击大厂，那么势必要对于知识点有足够深入的理解。例如框架源码的阅读，对于核心概念的理解等等。这里不能只是会背参考答案，最好是能够自己亲自去阅读完整的源码，融合自己的理解。

在最后的这个阶段，**如果碰到没有掌握的知识点，这时候就不要特意花时间去深入学习**，用刷题快速记忆即可。

3. 学习资料

3.1 基础

3.1.1 HTML + CSS

课程通常一般是从 JS 知识点讲起，不会包含这两块的知识点。这两块在面试时虽然被问的比例不大，但是是前端最根本需要掌握的知识点。务必花时间掌握。可以参考包含但不限于以下的方面：

- 多多练习页面的书写，多看看技术论坛大佬们开源的项目，模仿观察其页面 & 组件的 HTML 与 CSS 的写法，包含 HTML 标签的运用，页面结构的分配，CSS class 的命名等
- 掌握 CSS 处理器，例如 SASS、LESS 等，学会规范书写 CSS
- 掌握 HTML 与 CSS 常见的问题，例如常用的布局、CSS 布局、居中问题、BFC、重绘重排等等，可以从别人发的面经中提炼总结

HTML 推荐：

高频前端面试题汇总之 HTML 篇

html5 语义化标签

CSS 推荐

核心知识梳理

常见布局

flex 布局

grid 布局

使用技巧

BFC

动画

CSS 面试题相关

https://juejin.cn/post/6844904077806190605

https://juejin.im/post/5e53cefbf265da57570475a7

https://juejin.cn/post/6844903567707357197

3.1.2 JS

JS 模块的知识是前端最重要的基本功，需要牢牢掌握。

课程几乎能够覆盖大部分的 **JS 知识，会涉及到 Promise、JS 模块化、浏览器、原型、this 指针**等等知识。

但是整体来说 JS 的知识点较为零散，需要自己总结 JS 的知识图谱，包括**但不限于**以下：

- this 指针
  - this 的类型
  - 如何改变 this
  - call、apply、bind 的区别，手写代码
- 原型与继承
  - 模拟 new 的实现
  - 即成
- 执行上下文、作用域（链）、原型（链）
- 异步
  - promise 规范
  - async、await
  - event loop
- 类型
- 模块化
- 正则
- 事件机制
- 常用工具函数
  - 防抖

- 截流
- 浏览器渲染机制
- 渲染机制
- 回流、重排
- ES6/7/8
- 垃圾回收
- 设计模式

同时可以把知识关联在一起记忆。例如：

- 提到原型 -> 原型链 -> 多种继承方式 -> ES6 class 继承
- 异步 -> 宏任务/微任务 -> promise -> async await -> Node event loop
- 闭包 -> 内存泄漏 -> 浏览器垃圾清除机制

形成自己的知识图谱，方便观察自己薄弱的一块是什么，以及便于联想记忆。

这一部分出了课程笔记总结外，也可以通过论坛文章辅助。

课程补充推荐资料：

- 书籍
  - JS 高级程序语言设计 （红宝书）
  - 你不知道的 JS（电子书）
  - JS 语言精粹 （书）
- 文章
  - Javascript 深入系列
  - JavaScript 深入之从原型到原型链
  - 最后一次搞懂 Event Loop
  - 从浏览器多进程到 JS 单进程渲染
  - 前端模块化
- 高频 JS 题型（分为上下两个系列）

3.1.3 框架

需要做到能够熟练掌握至少一个常用框架（Vue/React）。

Vue

在熟练掌握框架基本使用的基础上，掌握高阶方法例如 nextTick、slot 等等。这些高阶方法在处理一些较为复杂的场景和排查问题的时候很有用，平时需要多累计，多看看文档，同样也可以多看一些好的项目来学习写法。

官网的 api 你基本上要全部过一遍。并且你要利用一些高级的 api 去实现巧妙的封装。平时需要多累计，多看看文档，同样也可以多看一些好的项目来学习写法。

还需要掌握组件设计原则，可借鉴 Element-UI（例如通过 extend 实现一个 messageBox），合理设计业务组件与基础组件。

推荐：

Vue-admin（常见的组件封装、路由设计）

confirm 组件的实现

实现一个动态加载的表单

长列表渲染优化

React

首先也是需要把官方文档中的基本概念掌握，需要熟练掌握官网中所提到的技巧，就算没有用过，也需要知道在什么场景下使用。

React 已经进入了 Hook 为主的阶段，相关的社区库也逐步完善。

举几个例子：

常用的有哪些？都有什么作用？

如何使用 hook 在依赖改变的时候重新发送请求？

写过自定义 hook 吗？解决了哪些问题。

讲讲 React Hooks 的闭包陷阱，你是怎么解决的？

可以带着上面的问题，阅读以下关于 Hook 方面的文章自己思考一下：

useEffect 完整指南

096.精读《useEffect 完全指南》.md

函数式组件与类组件有何不同？

同样，React 最好可以搭配和 TypeScript 一同使用，这里给出一些比较好的实践参考文章

react-typescript-cheatsheet

React + Typescript 工程化治理实践

最后是 React 的项目实践，可以作为入门或加深理解的辅助。

React 项目实践参考

3.2 进阶

3.2.1 TS

熟练掌握 JS 的情况下，可以开始接触学习 TypeScript，明白其背后的使用意义。推荐以下文章：

TypeScript Handbook 入门教程

工具泛型在日常开发中都非常的常用，必须熟练掌握。

TS 一些工具泛型的使用及其实现

这五篇文章里借助非常多的案例，为我们讲解了 ts 的一些高级用法，请务必反复在 ide 里尝试，理解，不懂的概念及时回到文档中补习。

巧用 TypeScript 系列 一共五篇

TS 进阶非常重要的一点，条件类型，很多泛型推导都需要借助它的力量。

conditional-types-in-typescript

3.2.2 框架源码 & 优化

框架源码的阅读便于开发时能够更高效地发现问题，排查问题。还可以掌握高阶 API 的使用原理，顺便可以学习框架的设计架构，以及 JS 高阶用法，好处多多。

推荐可以看论坛的总结文章，需要对关键模块的代码有印象，不要死记硬背，需要适当懂其中的实现原理。

Vue

Vue-技术揭秘

Vue-技术内幕（详细）

以上两份文章是针对 Vue 源码的详细解析，尤其是第二篇。这里如果时间充裕的情况下，推荐去阅读以下 Vue 源码的核心部分讲解。主要包含以下几个方面：

- 响应式原理，双向绑定 （http://caibaojian.com/vue-design/art/7vue-reactive.html）
- 生命周期有哪些，每个阶段做了什么事情
- 组件间通信
- $nextTick 的原理
- key 的作用，为什么不推荐使用 index
- Object.defineProperty 的优劣
- Vue3（proxy）与 Vue2 的对比

核心概念了解完之后，可以继续通过刷题来扩充知识面以及加深理解：

Vue 源码详解之 nextTick：MutationObserver 只是浮云，microtask 才是核心！

探索 Vue 高阶组件 | HcySunYang

Vue3 究竟好在哪里？（和 React Hook 的详细对比）

网上都说操作真实 DOM 慢，但测试结果却比 React 更快，为什么？

常见 Vue 面试题

Vue 面试题

React

在性能优化方面，需要掌握基本的组件重渲染，以及 React 如何对函数式组件进行优化，再到高阶组件的相关知识，可以参考以下文章

optimize-react-re-renders

react 中为何推荐设置 key

如何对 React 函数式组件进行优化

【React 深入】从 Mixin 到 HOC 再到 Hook

React 16 加载性能优化指南（比较细节，不必死磕）

React 源码学习笔记

React 常见知识点总结

React 常见面试题

3.2.3 工程化

掌握 webpack、babel 的用法，包括常用配置、优化手段等等。

大厂还需掌握其实现原理，例如 webpack 热更新、自己写 loader、plugin 等。否则掌握其常用优化配置手段即可。

推荐：

Webpack

从 0 到 1 构建 Webpack（没有实践过的建议亲自配置一遍）

玩转 webpack，使你的打包速度提升 90%

SplitChunksPlugin 私房菜

常见 Webpack 面试题

Webpack 打包产物分析(一)（结合前端模块化理解）

Webpack 打包产物分析（二）

Babel

读完这篇你还不懂 Babel 我给你寄口罩

.babelrc 文件配置详解

AST

AST 入门

AST 实践：try-catch

3.2.4 计算机网络

这部分知识需作为知识面的一个扩充，前端在调用接口、资源请求的时候都不免会接触到网络请求。需适当了解这部分知识点。

常见考察的知识点包括但不限于以下：

1. HTTP 版本之间的区别
2. 缓存
3. HTTPS
4. 常见状态码
5. DNS
6. TCP/UDP

除了以上知识点之外，通常会结合实际的场景题目来考察，例如比较典型的「从输入 URL 到回车网页渲染完成的整个过程发生了什么」这类问题，还可能结合网络请求优化、代码部署等问题。

推荐：

「查缺补漏」巩固你的 HTTP 知识体系

TCP 协议灵魂之问，巩固你的网路底层基础

前端代码部署方案

细说浏览器输入 URL 后发生了什么

预测最近面试会考 Cookie 的 SameSite 属性

谈谈 HTTPS

深入理解浏览器的缓存机制

大公司里怎样开发和部署前端代码

3.2.5 算法

先去找相关的入门文章，把极大算法分类掌握，做一些典型的例题。对算法有了大概的概念后，去 LeetCode 平台刷题，难度以 medium 为主，hard 题掌握例题即可。刷题频率无需过高，保持手感即可，切记盲目疯狂刷。因为很多题目解法是相同的，万变不离其宗，需要自己多总结，多记录。

建议按分类做题，但是要自己**总结每个题型的解题模板和套路**。比如滑动窗口、递归、动态规

划等等这些都是有一些基本固定的套路的。按目录刷题的时候，做完一定要去看题解，这里不局限于 js 的题解，可以看看热门的几个解答。里面通常会给出解题的思路，这里就需要你总结出来，然后可以去找同类的题型去套用看看。一个类型基本上做两三道就有感觉了。重点在于总结。刷题不要过分追求数量，找到解题方式之后适当做一些变种题型即可。在把每种题型都过一遍之后，再去进行综合随机刷题即可

进大厂必刷。

推荐：

写给前端的算法进阶指南，我是如何两个月零基础刷 200 题

3.2.6 手写系列

该类型的题目不要过于焦虑，精力有限的情况下可以用比较简短的方式写出来，但是相关的细节逻辑务必要牢记。

如何模拟实现一个 new 的效果？
如何模拟实现一个 bind 的效果？
如何实现一个 call/apply 函数？
三元-手写代码系列

手写 Promise 20 行

剖析 Promise 内部结构，一步一步实现一个完整的、能通过所有 Test case 的 Promise 类

4. 简历准备

简历中的重点主要集中在项目经历部分。如果是要去好一些的公司，这一块的准备需要突出自己的亮点。面试中不可避免会被问到“项目中有遇到什么问题（或难点）”。一般的书写思路是 star 法则，即问题的背景+你的职责是什么+你做了什么样的事+产生了什么样的结果。这里主要围绕你做的工作以及一个**可量化**的结果。这里的可量化指的是最好有具体数据支撑，例如提升了 xx 秒，提高了 xx%，而不是空泛的一句“极大提升了页面的效率”。

如果在这方面没有过多的亮点，在书写简历的时候避免堆砌过多同质化的内容，例如两个项目之间使用的技术栈与做的工作基本类似。这种情况尽可能地多展示自己会的技术栈，即展示自

己的广度。

总体来说还是希望学员在工作中能够找到可以提效或优化的地方作为自己的亮点。这一块值得单独话时间去整理，最终呈现在简历上。

4.1 亮点

简历上是否有亮点是面试中很重要的一环，而这也是大多数人遇到的问题。大部分的亮点大致可以总结为三个方面。

4.4.1 效率

1. 打包构建速度提升
2. 高度封装组件减少重复开发
3. 开发 loader 与 plugin
4. 开发脚手架 cli
5. ....

4.4.2 优化

1. 打包产物体积优化
2. 资源加载优化（例如图片资源、滚动懒加载等等）

4.4.3 解决方案设计

这个方面涉及到的比较广，举个例子，B 端项目中常见会用到权限控制。这里的鉴权、权限颗粒度设计、角色与权限的设计等等，在复杂系统下需要考虑的点有很多。那么如何能够设计出一个好的权限系统既能符合自己项目需求，又能够抽象出来作为一个独立平台对接其他项目，这就是一个解决方案的设计。可以观察自己在项目中在这方面是否有发挥的空间。

4.2 查缺补漏

- 时间较为紧凑的情况下，不要盲目去盯着不会的新知识点，而是应该专注在巩固自己掌握的部分

- 必须会的知识点还是需要的，比如想进大厂的话需要刷算法题目，时间紧的话就是刷每个题型的典型例题，时间宽裕的情况可以有计划地对每个题型进入深入的研究，掌握题型的延伸变种等等。
- 在整个复习的过程中，要总结出自己的“杀手锏”，即自己最有信心的领域。例如可以是对于源码的掌握度非常高，抑或是对于 JS 语言的理解很深入，项目打包优化有很多实践等等。以此在面试过程中可以给面试官留下印象
- 面试官不会因为你某一题不会就把你 pass，换句话说，你可以对某一块的知识点不那么熟悉，但是你不能对任何一块都不熟悉，这才是你被 pass 的原因

4.3 岗位分析

看职位的时候，比较重要的是看职责描述。这里面一般来说会描述你之后会负责的产品和项目。通常来说，描述中会直接体现出工作负责的类型，例如小程序、数据可视化、低代码等等。因此在做面试准备的时候需要有相关的侧重点。比如你有小程序开发的相关经验和岗位职责温和，那么投递的时候简历中最好可以将其作为一个突出点来描述。

5. 一点建议

- 平时学习最好能养成写文档的习惯，把知识点总结记录下来，再提取出来变成题目
- 平时多看看例如掘金、思否这样的论坛，一来可以多看看其他人的面试经验、题目，二来可以看看新的技术。可以养成固定的周期去刷刷别人的面试题，来检查自己学习的成果。
- 类似“只会 xx，不会 xx 怎么办”这样的问题，记住一个原则：你可以不会 xx，不会 xx ，但是你不能什么都不会。面试官不会因为你一道题不会就刷掉你。找到自己的“杀手锏”。
- 整体的课程内容很多，且密度很大。不要幻想暗示上课即可顺利进入大厂。例如 Vue 的课程可能只有两三节，这远远不能涵盖所有 Vue 的知识。课余时间得自己额外花数倍的时间去深入。
- 多看的同时多动手写，多想怎么能运用在自己的项目中。

6. 推荐

推荐两篇值得看的专栏以及推荐关注这两篇文章的作者：

写给初中级前端的高级进阶指南（这里会有很好的资源链接与学习路线）

一个合格(优秀)的前端都应该阅读这些文章

技术论坛：

掘金：https://juejin.cn/

思否：https://segmentfault.com/

饿了么专栏：https://www.zhihu.com/column/ElemeFE