结构化的总结前端面经

面试官心理

知己知彼，百战不殆。在正式阅读本文之前，我们先了解下本文的目的。与其他的拓展知识与增加技术深度不同。本文简单的目的就是为了让你通过面试，相信这也是大多数人阅读面经的目的。下面将对面试官的心理进行分析，从面试官的角度来确定我们的面经要怎么做。

面试环节

从面试轮次上来说，分为：

- 技术面，部门内部面试
- 技术面，部门leader面
- 技术面，交叉面试（兄弟部门）
- 技术面，部门大领导面试（高p）
- hr面

部门内部面试更多的是从技术的广度和深度挖掘，这部分也就是所说的八股文最多的一环。

leader面分为两种，技术leader和业务leader。技术leader一般会对技术的深度进行挖掘，挑2-3个技术难点，展开叙述。业务leader相对简单，会着重询问以往项目经验，项目难点等。

交叉面试一般由兄弟部门来面试，避免面试时单一部门的判断不准。还有一种情况是hc在多个部门都有，大家也是在挑人。

部门大领导一般侧重广度，对你以往做过的项目比较感兴趣。hr面更侧重对个人的素质，价值观等方面的考察。

一般来讲，基本的面试分为几个步骤

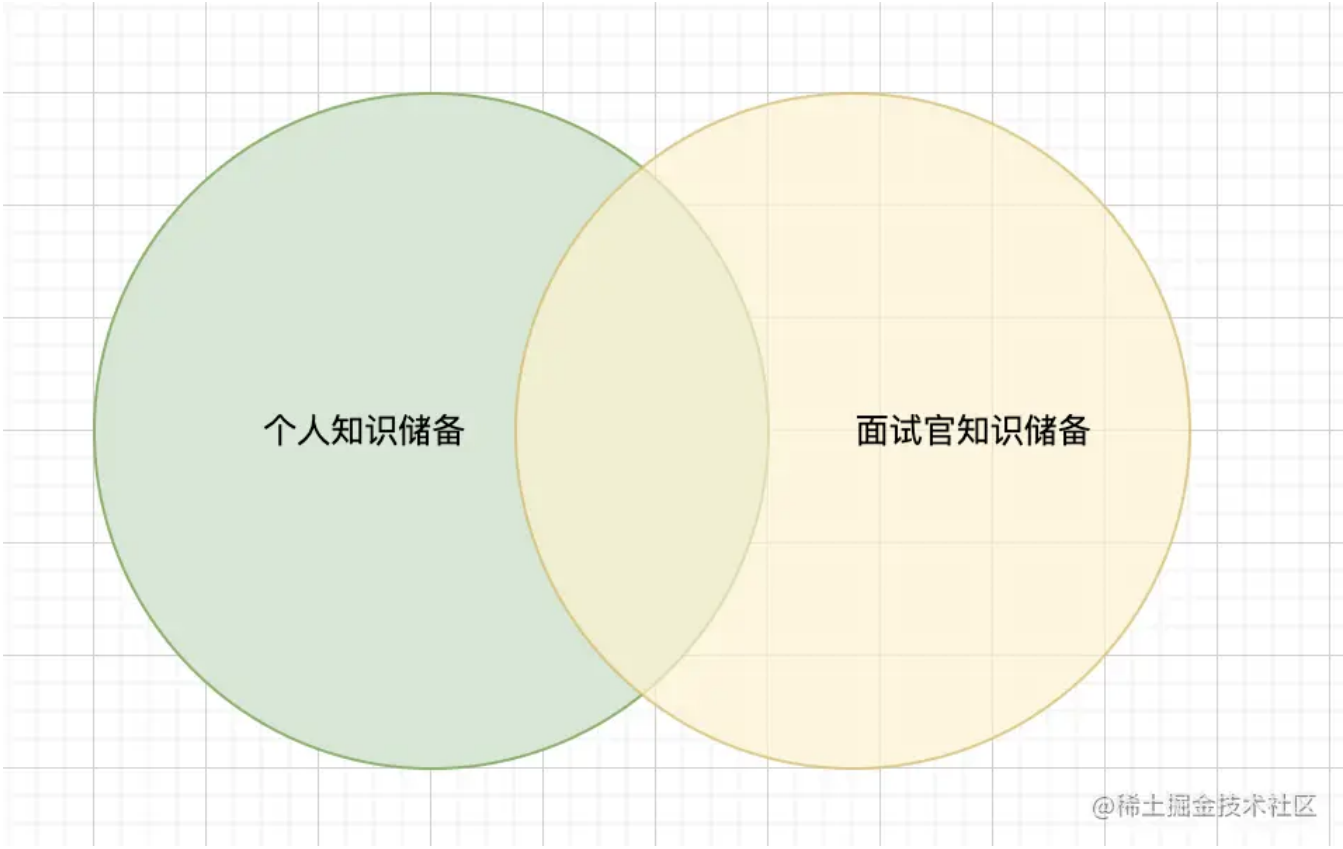
- 自我介绍
- 技术知识
- 项目介绍

- 发展期望

本文着重讲述技术知识这个部分

知识的自我储备与他人储备

我们思考一个问题，面试官的问题是否有迹可寻。答案是肯定的，任何一个人的知识都不是凭空得来的，所以技术问题肯定围绕这个人在项目中实际应用，或者在即将做的业务相关，再或者部门的整体要求（例如算法，技术栈等）。



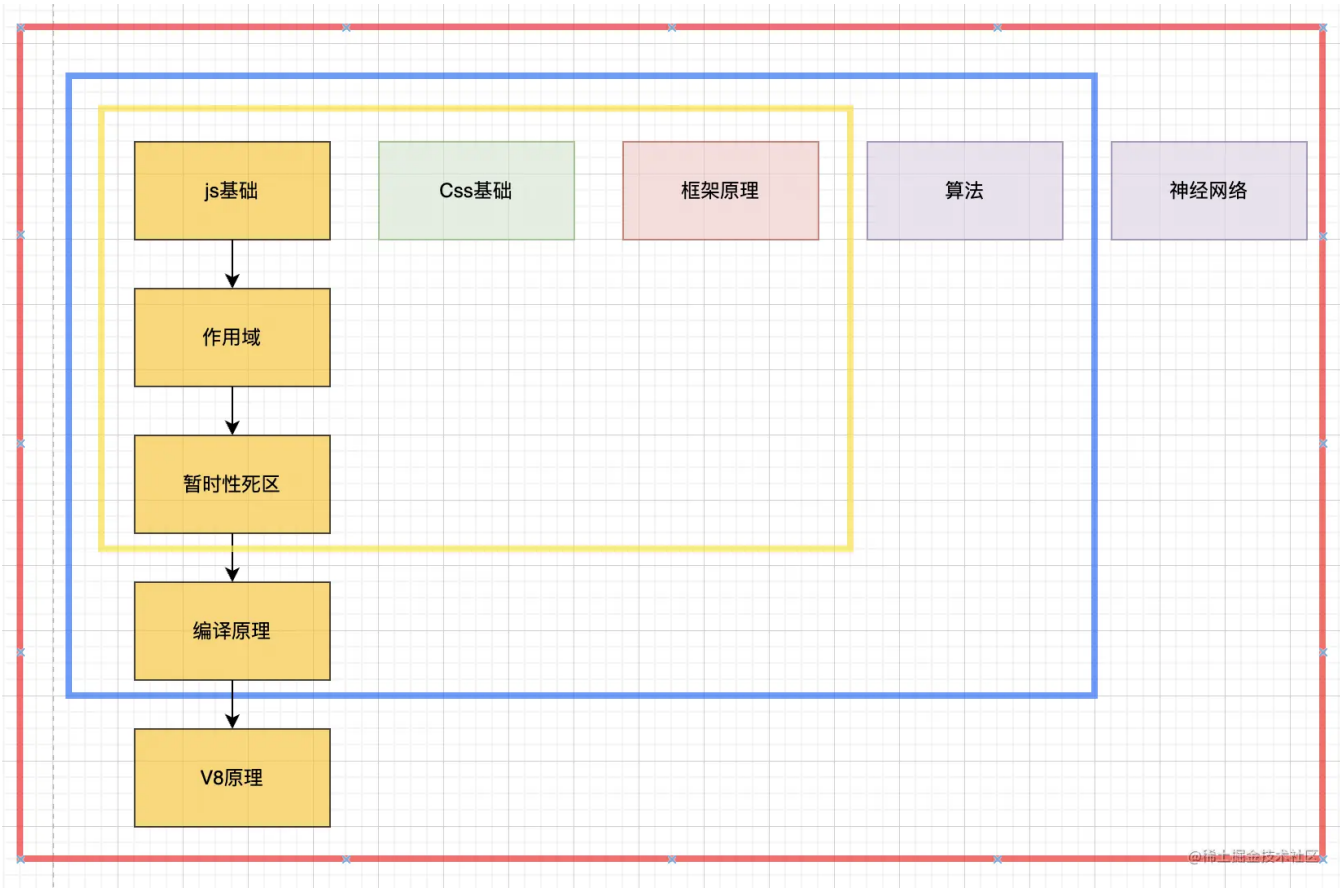
个人的知识储备跟面试官的知识储备是不一致的，或者说任何两个人的知识储备都不可能完全一致。

所以面试官的大部分问题会围绕面试者的知识储备和业务需要的技术来问。即，一个全栈的面试官在面试前端候选人时不会问后端的问题。

所以两者知识储备越契合的时候，面试通过的几率越高。

技术的广度与深度

上一段说了知识储备的问题，那是不是一定就要求了面试者要完全了解面试官的知识储备呢？答案是否定的，这涉及到了一个面试的广度与深度的问题。



如上图，每一个面试者其实会对应不同的级别有能力模型的要求。

比如一个p5，我们只要求对各种基础熟悉，能使用框架，能解决开发时遇到的问题。

而一个p6，我们要求是能够独立开发，对工程化，编译原理熟悉，能够指导p5的工作。

而一个p7则要求对底层了解，框架精通，技术视野广阔等。还可能加上成果产出。

所以在面试的时候，面试的问题不是无端的讲出来的，往往跟随面试官自己的知识脉络，层层递进。

可能开始的时候，问了一个作用域的问题，看到你回答的很好，所以想提高点，继续询问编译原理。在编译原理回答的很好的时候，继续去问v8底层。这种层层递进的询问，能够有效的了解你的知识的深度有多深，你的天花板在哪里。

而对某一个领域深度挖掘后，确定了你在这个领域的天花板之后，会换另外一个领域继续挖掘。在满足基础的要求后，会延伸到其他面试官所熟悉的领域，用来测试你的知识广度。

面试期待

一般来说面试官自己心里是有一个能力模型作为评估候选人的，但这个模型并不是稳固的，而是变化的。

	研发人员胜任力模型					
维度\等级	权重	实习工程师	初级工程师	中级工程师	高级工程师	架构师
工作年限 极少数情况可酌情处理。	必备	0-1年	1年以上	3年以上	5年以上	5年以上
综合评估项 评估师、本人都可以有一定的额外加分权利。 评估师也可在评估其它项之前，先对该员工进行一次感性的打分。	10					
技术能力 对开发技术产品和解决技术问题所需要掌握的开发能力、设计能力、架构能力	40	*能独立完成指定任务，处理局部问题。	*了解技术实现细节，能够对技术方案提出自己的建议。 *了解线上部署环境，能够分析和排查线上故障 *局部代码优化，基本的SQL调优 *对于移动开发，熟练掌握Andriod、IOS开发相关知识，及移动终端技术 *对于前端开发，熟练掌握CSS，js等语言，了解jquery、fdev4框架，遵守前端开发规范	*熟悉公司当前所使用到的所有主要技术。 *能够独立地基于公司的平台、产品进行项目级的开发、调试、部署。 *能独立解决问题，能够负责重要业务模块的需求分析及设计实现，且对线上部署环境比较熟悉，能够独立分析和快速排查线上故障。 *熟悉 CLR/JRE 原理，.NET/JAVA 高级特性和类库，.NET/JAVA 网络与服务器编程，多线程编程。 *具备辅导他人的能力和技能，有良好的分享习惯。 *使用并使用过 5 个以上的设计模式。 *熟悉 UML 类图、活动图、序列图。 *红线：从不编写重复的代码。 *对于移动开发 熟练掌握Andriod、IOS开发相关知识 移动终端技术 至少在某一方比较精通 *对于前端开发，独立完成复杂页面前端设计和开发，能够制定前端开发规范。	*精通公司当前所使用到的所有主要技术（如DDD、TS 等）。 *某一领域比较精通，具有一定权威（比如服务化、分布式系统、大型网站架构、大数据量存储、业务分析和规划等）。 *熟悉OOA、OOD，精通并使用过10个以上的设计模式。 *熟悉 UML 各种图形的设计。 *较强的封装能力，并对公司贡献了一定的可复用组件。 *具备高质量编码能力：重用性，低耦合，可扩展性，高性能，可维护性，安全性高。 *绝对符合编码规范的编码能力。 *对常用的语法和框架、类库，了解其实现的原理。 *开放的技术心态与思维，不局限于单独的语言或平台。并至少有两个以上编程语言的应用案例。 *对于移动开发 全面掌握Andriod开发或IOS开发、部署相关知识 移动终端技术 在多个方面比较精通 *对于前端开发，能带领设计并实现前端开发框架，带领设计并实现前端开发工具	*理解并熟练掌握一种以上的架构设计方法论。 *能独立带领产品或业务开发团队向前发展，独立承担架构规划，建设以及调优。 *有大容量、高并发分布式系统架构设计经验；或有企业级产品平台构建经验。 *精通 UML 的设计。 *精通 OOA、OOD。 *精通并使用过全部 GOF 设计模式。 *熟悉常见的架构模式（企业应用、互联网、移动端）。 *对公司底层技术有一定的贡献。 *对产品、项目提出意义重大的架构的建议。 *掌握框架设计方法；开发过有一定复杂性的框架或类库。 *能够负责复杂度高，平台级产品或跨团队的产品的架构，系统设计和实现。 *全局技术架构设计规划，复杂问题排查总结。 *开放的思考，考虑其它上下游业务线的技术架构影响与协同。至少应用过三门以上的编程语言。 *服从公司整体架构安排。 *精通但不限于：IOC，AOP；分层的应用框架设计思想；SOA，事件驱动；分布式系统原理：CAP，最终一致性，幂等操作等；大型网络应用结构：消息中间件，缓存，负载均衡，集群技术，数据同步；高可用，可容灾分布式系统设计能力；大容量数据存储和检索系统设计能力：数据库分区，NoSQL，搜索引擎等 *若已经通过国家《系统架构师》认证，则可认为理论方面较为全面。
工作态度&目标导向&文化匹配度 根据业务目标，能通过行动，克服各种障碍达到工作结果	10	*完成指定分解目标 *为达目标，积极、主动。	*团队协作，以团队目标为主要目标。 *能在参与业务中发挥自己能力，为业务结果发挥自己的作用。	*能主动考虑结果导向，能够在拿结果中发挥主要作用。 *理解执行结果达成的关键环节。 *能较好的处理风险。 *对不能完成的任务，不会轻易说不。 *文化匹配度：高。	*自我驱动！ *结果导向，能够打破常规，坚持为了拿结果做正确的事情。 *推动结果达成，资源与方案推动力。 *能较好的预知、预防风险。 *文化匹配度：高。	*能够对复杂事情进行规划，并能通过推进规划的实施，达到预期的结果。 *规划预测与推动，有自己主导的项目产品或技术。

上图是一张技术人员的能力模型，一些是硬指标。必须达到。而有一些是加分项。

技术的深度与广度，在胜任的前提下，其他的都是加分项。所以准备面经的第一步就是先了解自己职位对应的能力模型，然后有对针对性的补足自己的能力。然后整理深度与广度两个部分，其中深度优先级高于广度。精而不广和广而不精的能力取舍，在基层开发人员身上，精度更重要。

一般来说面试官在询问问题时，对于基础问题已经回答很好的时候，尝试挖掘技术深度的时候，这些深挖的内容都是加分项，这部分的评分会更高。有一部分面试官喜欢从你的简历上挖掘问题，所以简历上不会的不要写。

而在某个领域得不到期望的回答，换另外一个领域时，对于个人的评价相对而言会逐步降低。当他尝试问了超过3个不同领域，而无法得到满意答复的时候，基本就意味着你的面试可以终止了。

互联网面经的缺点

前面我们大概了解了面试官的心理，这部分我们来聊聊互联网面经的缺点，简而言之三个问题

- 面试题过于零碎
- 面试细节不详

- 深度广度并存，无🏁

常见的面经基本是以下格式：

xxx公司 xxx面

1.http的缓存协议

2.手写防抖节流

3.xxxxx

从这只言片语中，我们很难理解为啥考官要这么出题，为啥突然从A问题跳到了B问题，他面试的岗位是否一定需要xx知识。

而实际的面试过程中，可能是面试官，先在领域A问出了1问题，然后从1问题延伸出了2问题。在2问题觉得可以结束A领域的时候，从B领域问了3问题。然后最后作为扩展，问了4问题。

其中只有1，3是硬性要求，而2，4是加分项。

还有个问题是面经背了，但是面试官根本没问。或者面试官从1问题，延伸出了2问题。1问题对答如流，2问题吭哧瘪肚。

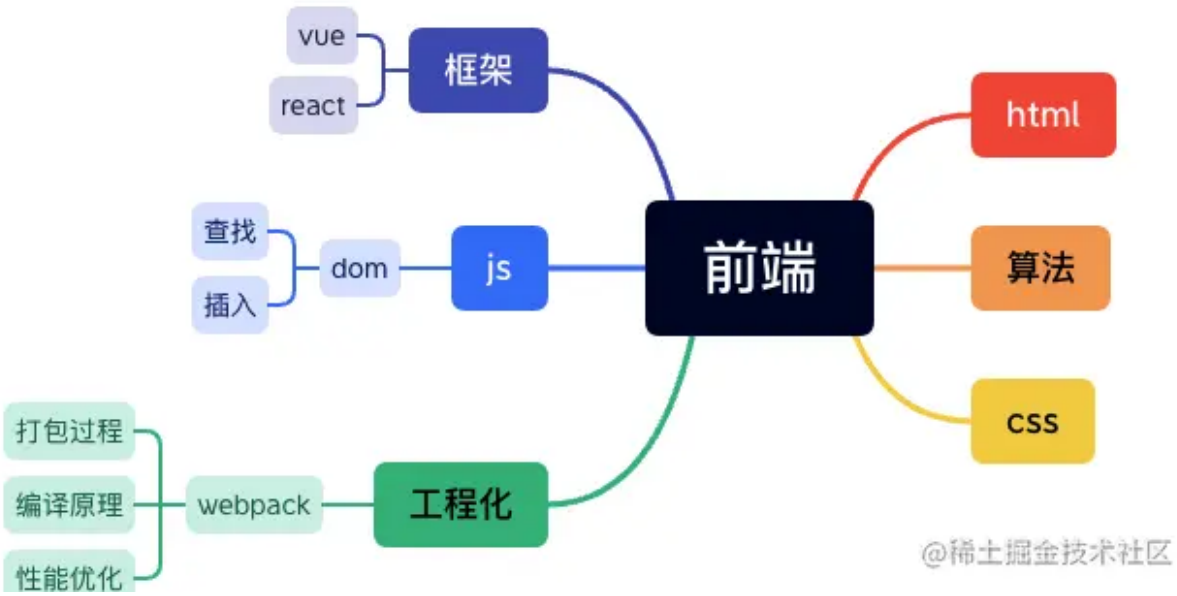
我们在茫茫的面经海洋里遨游，哪个是珍珠哪个是沙子呢？所以我们需要用 结构化 来重新定义我们的知识点，建立我们的一个珍珠甄别体系。

关于结构化的详细定义不在这里赘述，简而言之就是从中心点发力，按照结构树的形式，逐步推导，整理。 在这个过程中，各个知识点不是零散的，分散的，而是有结构，有逻辑，有组织的。

如何整理自己的面经

我们需要通过脑图，形成自己的一个知识体系表，在后续新知识的加入时，对节点进行增删改查。定期检视节点上的内容是否掌握，从而将自己的知识体系形成。

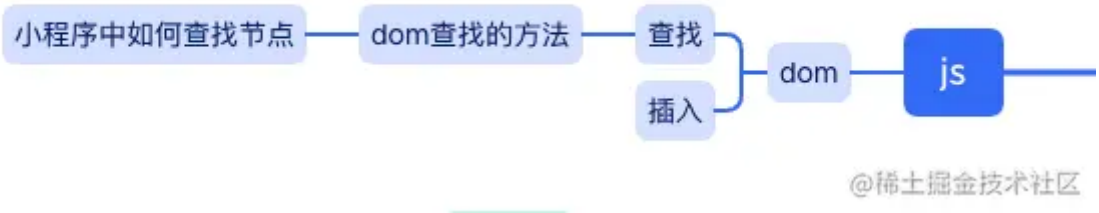
面经节点



我们可以先整理这样一份简单的节点路径，然后把你知道的知识放置上去。

然后我们可以先假设问题，例如dom查找有哪几种方法，

这就 形成了第一个面经的问题。然后我们延伸这个问题。那么在小程序中是如何查找的?



再次延伸，普通的是通过id，class等节点查找，那如果我有一个坐标点，我是否能够查找到对应的dom呢？这个问题比较深入，我们还可以标记一个不同的颜色来确定它的重要性。

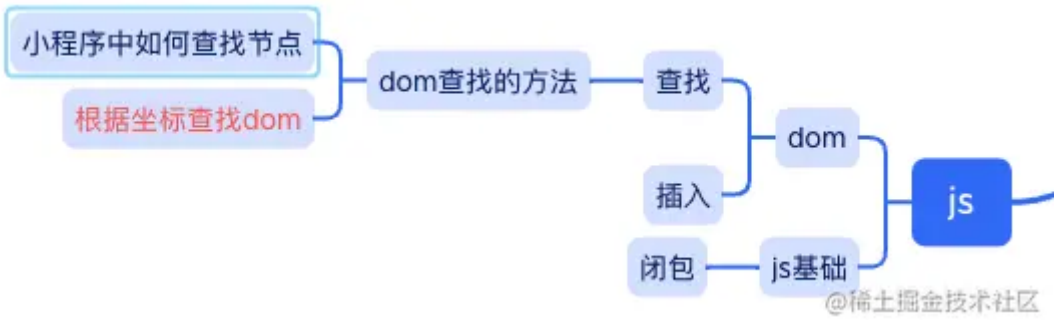
由此，我们就根据自己的预期来假设了很多问题。 这部分问题并不完美，但是我们已经对面试官做了一些预判。

- 面试官:说说dom查找的方法
- 我： 可以通过id，class，attr等查找，例如getElementById,getElementsByClassName巴拉巴拉，也可以通过querySelector这种更加泛化的查找方式，巴拉巴拉. 在小程序中，我们可以通过`selectComponet`查找组件,巴啦啦
- 我： 以上都是通过已知的信息去查找，我们还可以通过给定坐标的方式，通过`elementFromPoint`的方式，巴啦啦

这样就很自然的讲面试官的一个问题，自己延伸出了深度，提高面试官的印象分。当然，如果你不是进攻型的选手，你也可以通过等待面试官的下一步提问。但是因为你已经预判了面试官的预判，所以你可以立于不败之地。

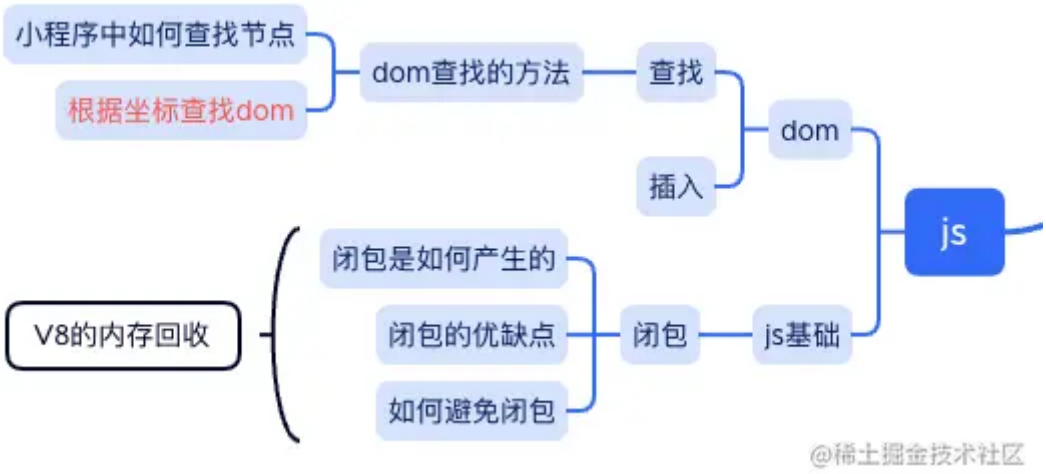
新问题整理

我们在总结了自己的知识之后，还需要不断的吸取互联网上的面经问题，不断的扩充到自己的知识库中。 比如面经中提到了 闭包 ，我们就可以加入到自己的js分类中。



然后我们需要继续思考这个问题延伸，比如闭包是如何产生的？闭包的优缺点？如何避免闭包?

再向后延伸，就是v8了



至此我们深入挖掘了一个知识点，在别的知识点出现的时候，也可以及时的放入到自己的知识体系。

除此之外，我们还需要定时的检视自己的知识体系是不是合理。例如闭包的问题，我们经过一段时间发现，其实它更应该放入内存的分类中。我们这时候就可以重新移动了。同时，我们也可以新开一个分类来专门说v8,将闭包与v8内存联系起来。

现在，对于闭包的问题，我们就可以这样理直气壮的回答了。

闭包的产生，优缺点，如何避免->不同浏览器的处理方式->V8的内存回收。

层层递进的跟面试官清晰的阐述。

最后

祝愿大家在这个寒冷的冬天，都有一份好offer。