像专业人士那样聊天

在深入代码之前，大部分面试官都喜欢聊聊你的背景。他们想知道：

- 关于编程的元认知。你有考虑过如何才能更好地编程吗？（译者注：元认知（Metacognition）或译为后设认知，这个名词由 Swartz 及 Perkins 发明及定义，即“认知的认知”或“知识的知识”。简言之，就是对自己的认知过程（包括：记忆、感知、计算、联想等各项）的思考。）
- 自主精神/领导能力。你看到你的工作完成了吗？就算你不必这样做，你会主动修正不对的事情吗？
- 沟通。他们会和你聊聊技术上的问题是有用的，还是痛苦的？

你至少应该准备好下面中的一项：

- 你解决过技术问题中一个有趣的例子
- 你克服人际冲突的一个例子
- 领导能力或自主精神的例子
- 你在过去项目中应该做什么的故事
- 你最喜欢语言的一些细节，以及关于你喜欢和不喜欢说的语言上的一些事情
- 关于公司产品或业务的问题
- 关于公司工程战略的问题（测试，Scrum等）

好好准备相关材料。你要展示你做过最自豪的事情，你很希望了解他们正在做什么，而且你对语言和工作流程有一些自己的想法。

沟通

一旦涉及到编程的问题，沟通是关键。在过程中需要帮助但可以清楚与人沟通的候选人，要比那些可以轻而易举解决问题的候选人更加难得。

理解它是哪一类问题。问题有两类：

1. 编程。面试官希望看到你解决问题，并且可以写出整洁和有效的代码。
2. 聊聊而已。面试官只是想让你说点什么。这些问题通常要么是高层次的系统设计（“你如何设计 Twitter 的备份？”），要么是琐碎的事情（“JavaScript 中 hoisting 是什么？”）。有时候琐事是为了引出一个“真正的”问题，比如，“我们如何能快排一个整数数组？好的，现在假设我们不再使用整数而是……”

如果面试官在进入“真正的”问题之前，只是想很快得到一个类似闲聊的回答，而你却开始编写代码了，她会感到失望。只需要问一句，“我们要为它编写代码吗？”

让人觉得你们是一个团队的。面试官想知道和你一起解决问题会是什么样子的，所以要让面试官觉得你是乐于合作的。用“我们”来代替“我”，比如，“如果我们采用一个广度优先搜索，我们会很快得到一个答案。”如果要你选择在纸上还是在白板上编写代码，最好选择白板。这样你就可以坐在面试官的旁边，面对着问题（而不是和她隔着一个桌子）。

说出你的想法。我是认真的。“让我们试着这样做——但我还不确定它是否会起作用。”，如果你被卡住了，就说出你的想法。说说什么可能会有用。说说你认为什么东西可以工作，以及为什么它不工作。这也适用于琐碎的闲聊问题。当被要求解释 JavaScript 闭包时，“它跟范围有关，而且是在函数中实现”，这样的回答就可以让你拿到 90% 的分数。

说你不知道。如果你碰到一个产品（例如，特定语言的边边角角，一个涉及分析运行时的难题），不要不懂装懂。相反应该说“我不确定，但我猜测是这样，因为……”，这个『因为』包括通过展示其它选项的荒谬性来排除它们，或者从其它语言或问题中找到类似的例子。

放慢步调。不要自信地脱口而出一个答案。如果它是正确的，你仍然需要解释它，如果它是错误的，就会显得你很鲁莽。你并没有因为速度快而赢得什么，相反更可能会因为打断她或急于得出结论而惹恼面试官。

摆脱困境

你有时会被卡住。放松。这并不意味着你已经失败了。记住与找到正确答案的能力相比，面试官通常更加关注从不同角度探索问题的能力。就算希望渺茫，也要继续探索不放弃。

用户。别浪费时间光在脑袋里面想 —— 在黑板上思考。画几个不同的测试输入。用手画出如何获得想要的输出。然后思考将你的方法转换成代码。

解决一个更简单的问题。不知道如何找到集合中第四大的项目？先思考如何找到最大项，再看看你能否改变一下方法。

先写一个简单和低效的方法，再想办法去优化它。使用暴力的方法。尽一切努力得到答案。

更加大声说出你的想法。说出你知道的。说说你认为什么可能会工作和为什么它行不通。你可能会意识到它确实有用，或者一个改进的版本会游有用。或者你可能会得到一个提示。

等待提示。不要一脸期待地盯着面试官，只需要停下来“想一下” —— 你的面试官可能已经决定给你一个提示，只是在等待一个时机，避免影响到你。

考虑空间和运行时间的限制。如果你不确定是否能优化解决方案，大声说出你的想法。例如：

- “我至少要查看所有的项，所以我不能再优化了。”
- “暴力的办法就是测试所有的可能性”
- “这个答案将包含 n^2 个项，所以我至少要花这么多时间。”

把你的想法写业界

你很容易让自己陷入混乱。你先集中精神把想法写下来，最后才去关心细节。

调用辅助函数并进行下去。如果你不能很快想到如何实现一部分的算法，不论大小都跳过它。呼叫一个命名合理的辅助函数，声明“这将会完成某事”，并进行下去。如果辅助功能不重要，你完全可以不去实现它。

不要担心语法。只要略过它就好了。如果必须的话请转换成英语。只需要说你会回头看它的。

给自己留下足够的空间。你后面可能要在两行之间添加代码或注释。从白板的顶部开始，在每一行之间留下一个空白行。

为最后的检查做好标记。别担心你的循环应该是 “<” 还是 “<=” 。你可以做一个标记提醒自己最后去检查。只要把整体算法写下来。

使用描述性的变量名称。这样会花一些时间，但会防止你忘记代码正在做的事情。使用 names_to_phone_nums_map 代替 nums。在命名中暗示类型。返回布尔值的函数应该以“is_ ” 开头。拥有一个列表的变量应该以“s”结尾。选择你能理解的标准并坚持下去。

当你完成后，记得做好善后工作。

你输入一个例子，手动过一遍你的方法，并大声说出你的想法。当程序运行时，你写下变量保存的值 —— 在脑袋里做这些不会给你带来任何加分。这样可以帮你查找问题，并且可以消除面试官对于你正在做什么的疑惑。

查找一开始的错误。你应该在循环中使用“<=”而不是“<”？

测试边界情况。包括空集、单个元素集合或负数等等。加分项：提一下单元测试！

不要觉得麻烦。有些面试官并不在意这些善后步骤。如果你不是很确定这一点，可以这样说：“我通常会用一些边界条件来测试代码 —— 接下来我们要这么做吗？”

安卓

最后，处理实际问题是无法替代的。

用纸和笔来编写代码。诚实地面对自己。刚开始可能会觉得很棘手。这没什么。如果你现在能克服这种棘手的问题，等到真正面试的时候，就不会显得笨手笨脚了。

我们遇到的实际问题反映了面试的流程，即当你陷入困境时会得到提示，当你的算法可以进一步优化时也会得到鼓励。