【问】

嗯，好的，那我就现在就是按照我们这个访谈提纲来问一下吧，然后就希望你对每一个问题呢，有可能需要举一些例子，举一些比较具体的例子，然后希望你能够讲的稍微详细一些。那第1个问题就是想请您先简单的介绍一下您的专业背景，还有做过的开发的经历，可以简单的说明一下您做过多长时间的开发的工作。

【答】

我应该是2014年毕业的吧，嗯，其实2014年之前也有过实习，但是完完整整地从事某个项目的话，应该还是从14年开始，然后到现在的话应该有7年左右的工作经历。在之前的话，我第1家公司是一个做应用商场的一个公司，然后当时的项目主要都是公司内部的项目。里面的文档相关的工作其实是比较一言难尽啊。然后到后来这家公司的话，主要做的是开源相关的，然后在这方面的话我们对文档的关注相比之前的公司的话要重视很多。

【问】

相当于大概有7年的做开发的经历。那下面一个问题就是，我们这个研究是想了解一下开发者本身的一些特征，因为有可能他们的一些特征会影响到使用文档的习惯，所以我就想问一下您，您觉得作为一个程序员，你觉得程序员这个群体和其他专业其他职业的人相比有哪些不一样的地方？

【答】

嗯，其他专业，有没有什么具体的对比的对象？

【问】

没有具体的对比的对象，就是和其他职业的人相比嘛，您可以大概的性格特点啊，学习的一些特点，学习的方式，学习的习惯，甚至工作的方式，工作习惯，生活习惯都可以说一下。

【答】

嗯，其实我觉得性格特点方面的话，就普遍外界都有一个印象，说是程序员通常比较内敛，然后不太爱说话，然后天天对着机器什么的。但从我接触到的程序员的群体来看的话，这种人确实占很大一部分，但是也有很多人是有点社交牛逼的。所以很难给他一个准确的一个标签。

【问】

那您觉得在生活中可能性格比较内敛的人多一些，但也不是绝对的，对吧？那如果是在网上，在网上比如说程序员都会有一些自己喜欢的爱好者、开发者的社区，有一些兴趣社区，那在这些社区应该是比较活跃的吧。

【答】

是的，但也有不活跃的。但是通常活跃的这些，特别是开源社区里面活跃的这些贡献者，我感觉都有一个特点是，他们都比较爱表达。

【问】

爱表达，意思是喜欢用文字来表达自己的观点吗？

【答】

不局限于文字，但文字是他们主要表达的一个通道，他们也会通过视频或通过语音之类的来表达。

【问】

明白了，那您觉得从学习的习惯上来说有什么特点吗？程序员他的学习上或者是他的工作跟其他人有什么不一样的？

【答】

嗯，可能不同的领域会不太一样，比如说像数据库领域的话，因为很多技术其实都是直接在论文上有，或者是储存相关，还有算法相关的这些领域可能大部分人他的学习都是通过阅读相关行业里面每年新的论文或过去的一些经典的论文来了解最新的技术和发展的方向。但是如果像是一些，比如像前端那种，或者是偏后端的这种，做业务类型的这种领域的话，可能他学习的方向会更偏重于某一种语言或者是某一种框架，就是具体的一些实现方法，对，他可能会关注这些会多一点，因为这些才能够帮助他能够快速的去实现业务的需求。

【问】

那您觉得程序员这个职业它是有一个快速学习的需求是吗？就是需要不断的学习新的知识。

【答】

对，是的。因为嗯其实软件行业，或者说计算机行业，它的历史并没有像其他一些传统的那些什么机械行业那么长。但是它的迭代速度是非常快的，嗯，像我们现在所使用的一些经典的技术，可能也就是最近10年或最近20年才提出来的。演变的速度特别快，嗯，特别是像前端领域这种，就几乎是没过几年就会有一种新的技术出来，就是一种新的流行趋势吧，如果你不学习的话，就很可能就被淘汰。

【问】

那你学习的这个过程大概是什么样的呢？我们想了解一下您最近或者是您之前有学过某项新的技术新的工具或者新的软件吗？就一开始对你来说是完全陌生的，然后您把它学会了，最终学会了这个东西之后，用它来做一些具体的工作，开发的工作，然后希望您描述一下这个过程是什么样的，就是学习的过程是什么样的。

【答】

我刚毕业的时候啊，或者我还没毕业，当时还在实习，我当时是学习Linux，当时的学习是通过一本畅销书学习的，那么畅销书叫什么，《鸟哥的Linux私房菜》。那其实严格来说不是教材。好像跟教材也差不多，因为它的编录其实也是挺完整的，嗯，他更多是鸟哥这个人他自己的一些技术的总结。还有一些常见的一些坑。但是我工作以后的话，基本像这种阅读整本书的这种经历就比较少，通常是直接看论文，看它的原理，然后还有它的这个最终的实验的数据，然后看思维实验是不是可行啊，然后自己再尝试动手来复现看看。

【问】

就是说你前期是会看一些比如说教材或者是类似于教材的东西，或者是论文来来了解它，初步的了解它，那就相当于是一个入门的过程对吧？

【答】

对。

【问】

那入门了之后呢，您是要做什么？

【答】

当然是实现了。

【问】

那实现的过程中应该也还是有遇到问题吧，就还是会再返回去再看。

【答】

对，是的。

【问】

那您可以大概的把这个学习的过程分一个阶段吗？就分前期、中期、后期这样的阶段，然后可以大概的说一下各个阶段的特点。

【答】

你是说个人经历的这个前期后期的阶段呢，还是说单纯学一门技术的阶段？

【问】

单纯学一门技术的阶段。

【答】

好。或者我可以分享一下我学rust的这个过程。我最开始学习rust的话，其实也是通过他们官方的文档，就是最开始的那一本book，他们有个第一版book。我是看那个，当时那个book还不完整，我是看那个书一步一步的了解它的各种各样的概念。了解它的基本语法。嗯，了解这些以后呢，然后开始自己尝试用这个语言去写一些项目，但我用的最多的，当时用的最多的其实是用来用来刷OJ。就是看用rust做这些OJ的表现是什么样的。如果我能够把基本的一些题目都能够用这门语言来做完的话，那么我对它的基本语法应该是非常了解。然后之后的话，就是入职这个公司以后，然后开始用rust去写一个真正的项目，并且开始跟别人一起协作，通过跟别人交流一起看，对这门语言应该怎么使用，和它的一些会遇到什么样的坑，就是通过分享，然后就更进一步的去学习这个。然后到后期的话，其实跟项目本身有很大的关系，就跟项目的演进有很大的关系。就你在做项目时会遇到很多什么性能瓶颈，或者是一些异常行为，这时候你会需要去分析它，然后通过这种分析的话，你也会了解到该怎么样才能够更好的使用它。

【问】

您刚才也提到了前期通过看他的文档来入门是吧，然后之后就是尝试用这个语言，就了解了这个语言，然后尝试用这个语言来做一些题对吧，之后呢就到了公司之后用它来上手做一些项目，那这个在做这个项目的过程中应该是会有遇到一些问题，那你遇到问题的时候你会怎么办？

【答】

谷歌。就搜索。大多数情况下啊，网上都其实都有人已经遇到过这样的问题，嗯，当然还有少数问题可能是从来没有人遇到过，只有你遇到了，这时候的话我一般是会去那个项目的维护者那里去提issue。

【问】

项目的维护者，他是相当于是有一个类似于GitHub那样的可以提issue？

【答】

就是GitHub。我就习惯去官方项目那里提issue。嗯，我也会去加入那种什么，就是他们的线上的一些group，比如说像IRC或者是其他的一些什么forum之类的东西，然后去里面去提问题。

【问】

去提问题，那会有人回复吗？

【答】

会的。那回复您的是这个项目的负责人还是说有其他的开发者？

【答】

Contributor。

【问】

那除了就是说自己去搜索，然后提问题，还有其他的可以解决问题的方式吗？

【答】

就自己分析问题。

【问】

自己分析问题，是基于什么？

【答】

是基于比如说他如果某个行为不太对的话，啊，就直接分析它生成的这个汇编有没有问题？是不是符合预期？然后，如果这个都搞不定的话，嗯，当然这是比较简单的片段可以这么搞。如果是整体的一个非常异常的行为，比如说像什么突然间call down嘛，然后又没有任何信息，这种大型二进制的话，这时候会通过一些debug工具去分析它。

【问】

就是借助工具，自动化的这种debug的工具去分析？

【答】

对对对，是的。

【问】

那您其实可以大概的把这些问题分下类，就不同类型的问题是有不同的解决方案是吧？那您可以就举一两个例子说一下不同类型的问题通过什么途径来解决吗？

【答】

嗯，比如说它有一些语法上的限制，就是他就不能那么写，但是其实这么写是没问题的，通常这种情况下是编译器的bug。对这种情况下你就只能去提issue，告诉他有这个缺陷。他不一定能解决，即使解决你可能也暂时用不上新的这个版本。这种情况下，你只能通过另外一种写法绕过它。然后这是一类问题。然后另外一类问题是它运行时的问题，就是它运行时的时候，比如说哪个缺陷导致它挂了，那么这时候就需要通过各种各样的debug工具，或者是日志去看是发生什么问题。然后还有一个问题是性能相关的问题，就它跑的慢，你不知道他为什么慢，这时候就去要去profile它，或者是写一些非常简单的例子去看这种写法是不是最优的。

【问】

就相当于是去试错是吧，自己去试错？

【答】

嗯，我觉得性能的问题就不算试错的，他更加类似于分析，它就是分析。

【问】

那也会有遇到一些比较系统性的问题，就难以描述的，一些宏观的问题。

【答】

宏观的问题，那可能就是软件设计本身的问题，这种情况下的话通常是跟外部工具没什么关系，就是你自己这个项目本身存在哪些设计缺陷才会有这种问题。

【问】

那下面一个问题是您觉得文档在您的开发工作当中扮演什么样的角色呢？

【答】

嗯，非常重要的角色，事实上我每天都在打开文档，但是我打开的文档通常是比如说像是标准库的一些文档，更侧重于API本身，API文档多一点。原理性文档的话，只有在比如说重大版本的演进的时候，就我用的那些，比如说我用rust，它发生了重大版本的改进，我们就说他会添加一些什么新的features啊，或者是有些重大的变更啊，这时候呢会需要去了解一下它背后的逻辑，就是他为什么做这样的变更，变更会带来什么好处啊，这是第2个原因吧。然后业余的时候其实还会接触到的第3类文档，我不知道他算不算文档。就是比如说像一些博客啊什么的，就是技术分享型的文档。然后还有一个就是第4类，就是专业领域的，就是论文相关的问题。这种的话通常是，嗯，比如说是重大的这种课题活动，比如说某个重大会议开了以后下面会发一些论文，这种会集中看一看。或者是当我要做某些领域的事情的时候，你遇到事情的时候，需要先了解下这个领域背景的时候也会想看。

【问】

就是说您用这个文档，其实主要是还是来了解还有熟悉这个技术对吧？

【答】

嗯，是的。

【问】

嗯，那有没有就是通过文档来解决您遇到的一些问题的情况？就相当于它是一个工具书，你有问题的时候去查文档，有没有这种情况？

【答】

有，但是这种文档我用的比较少，其实我觉得初期的话，初学者可能会非常依赖于这个，比如说像你，我如果用这一类文档的话，通常是我在用什么特别的编辑工具或是特别的编译器的时候，我会需要用这些文档，但是一旦你掌握了这个技术，熟悉了这个编辑器或熟悉这个系统以后基本就不会用到这些。

【问】

嗯，好的，那您刚才其实也列举了一些类型的文档是吧？那你还有可以补充的吗？就是能想到的用过的哪些类型的文档，可以给它们分个类。

【答】

嗯，就刚刚分的四类。其实刚刚的四类，加上你刚说的，那应该有5类。一类是API文档。还有一类是原理性的文档，对类似于教材一类的东西。还有一类是论文，那种专业性比较强的。还有一类是纯分享的。还有你刚提到的这类就是类似于cheatsheet。

【问】

那您平时是通过什么方式去找到需要的文档，是怎么找的？

【答】

像API文档这种他都有固定地址，所以就不用特别去找，它的官方网站都会有。原理性的文档的话，他其实也有官方的这个网站，但是主要是通过社交媒体知道的，因为他要发个什么大版本或者做什么重要变更，他其实会提出来。

【问】

社交媒体主要是指哪些啊？

【答】

Twitter。然后还有的话是，像博客这种的话，嗯一方面是别人的分享，另一方面的话是邮件列表，就会订阅一些特别的技术相关的一些邮件列表，然后它里面会有编辑去做一些，就每周做一个汇总之类的，然后通过那个来看。还有一些比如像是Google group，他本身也有人会在里面分享，然后他会做一个每周的汇总，然后也可以通过这些途径来获得这些信息。

【问】

您提到的这些途径，其实都是自己就平时可能是比较感兴趣的领域，然后自己就关注的比较多，然后他们就会相当于有消息提醒这样，有更新的话就是会提醒您，然后就能够看到这个文档。那有没有自己主动去搜索？

【答】

有，就是论文这些。很少有人会分享论文。当然我们公司有论文阅读的活动，这也是一个途径。

【问】

那有没有通过搜索引擎来搜索一些具体的文档的情况？

【答】

搜索引擎搜索的话，那是在stack overflow上面搜，算吗？

【问】

Stack overflow上面的问答也可以算。

【答】

啊，这种有。这种比如说接触到一个陌生的系统，或是某个陌生的问题的时候，会通过Google来搜。

【问】

那想问一下您具体是怎么搜的呢？您可以想几个例子吧，举几个具体的例子，比如说您怎么构建这个表达式？

【答】

我其实很少构建表达式。我直接就输入关键词。

【问】

输入关键词，可以举一些例子吗？

【答】

嗯，比如说，当我遇到某个内核报错的时候，我不确定这个内核报错是怎么回事呢，我就直接把他的报错信息粘贴到Google上，一搜就基本能够搜出来。

【问】

除了这个报错信息，那关键词，是不是直接输入一两个关键词，中间空格这样的？

【答】

是的。

【问】

嗯，那有没有用到一些比较高级的搜索的方法？

【答】

非技术我可能会用到，技术的话我们好像很少用到这种高级。

【问】

还是习惯用关键词是吧？

【答】

对。

【问】

那您觉得关键词搜索的效果好吗？能搜到想要的信息吗？

【答】

基本能搜到。基本第1页就是，可能是我关键词抓的比较准啊。

【问】

那搜出来的第1页的信息是比如像stack over flow上面的一些具体的问答的页面是吧？

【答】

对，这里面其实我有个特别的原则，我会挑一些最近的问答来打开，我不会挑什么10年09年这种，通常我就不看了。

【问】

就是看最新的，嗯，那除了这些问答，会有这种情况吗？就是通过搜索关键词能够直接跳转到这个技术它的文档，官方文档的页面，有这种情况吗？

【答】

有，其实我搜API文档通常就是通过这种方式搜。但是一般来说的话，其实搜索引擎它本身就会学习你的行为，知道你更偏好于哪些内容，所以你搜的越多，它通常就能够给你推荐的越准，所以更容易能够找到你想要的内容。但是是很多时候，如果你打开太多了，其实浏览器里面也有，所以这时候你不需要去通过搜索引擎搜，你直接输入这个关键字，你浏览历史里面就出来，直接打开就可以了。现在还有一些特别的工具，比如说像Mac OS里面它能够，它有某种插件可以直接在你的系统里面显示这些记录吧，直接就可以打开，但我没有用到这些功能。

【问】

那就是您搜索的效果还是比较好的，是吧？那有没有遇到你搜不到的情况，怎么找也找不着的？

【答】

遇到，这种情况的话，要么就是有人遇到同样的问题，但是没有人给出一个答案，要么就是从来没有人遇到过这样的问题，那这种情况下就只能靠自己去解决。靠自己去debug，分析。

【问】

那您比如说找到一篇文档已经打开它，您是怎么快速的定位自己需要的那块信息呢？怎么阅读，就想问一下您阅读的习惯。

【答】

嗯，我觉得这种应该要分类。比如说像是论文这种，我会先读他的简介，确认一下他是不是我真的关心的内容，还是说他是个标题党。然后看了这个简介以后，我再看一下他做的实验是否是设计合理的，然后这些都没问题以后我才去看他的原理，看它的原理上是不是说得通的，这是我的论文的阅读习惯。嗯，像是一些博客之类的话，博客的话主要看它是哪些方面的，如果是我以前很了解的方面，我只是想看一些新内容的话，我会很大概的浏览，就是估摸着每一段是什么意思，每一段什么意思，然后没有新内容我就不看了。如果是我完全没有接触过的话，那么我会从头到尾一字一字的去看。如果是API文档的话，我就直接去搜了。就是到那个页面以后，我就是搜我要的那个函数。

【问】

就是通过他那个关键词搜索？

【答】

浏览器的搜索功能，CTRL+F。

【问】

那种原理性的文档，比较长的原理性的文档？

【答】

原理性的文档就是只能从头到尾慢慢读。

【问】

从头到尾，就是看他简介，然后具体的功能这些都都要读是吧？

【答】

对，是的。

【问】

那有没有这种情况，就是你带着一个问题，目的性很强的问题，然后来找您需要的信息，然后需要快速定位。就根本没有时间就从头到尾读的，有没有这种情况啊？

【答】

这个好像没有，我们的工作性质好像不包含这种十万火急的。

【问】

就还是有时间大概的看一下吧？

【答】

是的，但是你说的那种带有目的性的阅读，它其实是有，这种情况下也是泛读，就是每一段都泛读一下，看看有没有，比如说我要的关键字有没有在里面，都没有的话我就跳过了。其实很多时候很多原理都是触类旁通的，就是你懂得这个以后对应的那个其实也是懂的，那么这时候也不需要去细看，每一章点开看一下大概是个什么东西，就知道是什么意思。就挑从来没见过的东西看就行。

【问】

那您应该也看过比较多的文档，您可以举一些，国内国外的都可以举一些您看过的比较好的文章的例子吗？然后说一下他们好在哪里？

【答】

嗯，国内的话啊，我举PingCAP的例子会不会？

【问】

没关系哦。

【答】

嗯，我觉得我们对文档的组织应该是国内，应该是做得比较细致的一家公司，是吧？对，我好像还没见过有比我们做得更细致的一个国内的技术性的公司。然后国外的话我其实非常推荐rust系的那本书，嗯，rust语言它本身有一个入门版的书，还有一个非常深入原理的书，这两本书都写得非常好。

【问】

大概的说一下，总结一下他们好在哪里，大概可以从几个维度上来说，就是内容、组织结构、还有一些其他的设计。

【答】

它就是由深到浅。入门那本书，他是直接从他的原理，还有一些关键概念，然后一步一步去写。进阶的那本书的话，它主要特点是它会有一个例子，就告诉你说我要做一件事，一件什么样的事情，那么我做这个事情的时候是怎么样利用语言，是怎么样考虑的，然后去一步一步的把这东西做完。

【问】

就说他会给您一个比较形象的例子，帮助您去理解这个语言是吧？

【答】

对对，然后他也会有一些非常，嗯，我不知道是作者表达技巧还是怎么样，反正就是你会觉得他讲的东西都比较容易理解。

【问】

好像国外很多这种讲程序语言的教材，都是会喜欢用一个例子，很浅显的例子来帮助理解。那您能就也可以说一下PingCAP的这个文档它好在哪里吗？

【答】

我们的文档应该是主要分成三类的是吧？一个是那个reference，还有一个是Configuration，还有一个是how to是吗？我个人也写过里面的一些文本，它的主要特点是说，就是会针对不同的分类会做一些不同的调整。我之前写的，我好像主要是写的how to吧，啊不对，我也写过configuration。

【问】

根据不同的分类做一些调整，具体是指什么呢？

【答】

嗯，比如说how to的话会有比较详细的一些步骤，会解释啊这个东西应该是怎么做的，比如说如何安装部署。它里面还有一个比较好的点，但这个点很多公司也做到了，就是它里面有一些可以复制粘贴的一些命令。

【问】

就是例子是吗？

【答】

应该不算是例子吧，就是可以copy的代码。我很多时候其实，比如说要做我们公司产品的一些实验的时候，也是直接就copy过去粘贴就可以了。

【问】

就马上能用是吧？

【答】

对对，嗯，其实我们基本上就还是那三类，就concept， task ，reference，只是在呈现在那个导航里的时候进行了一些重新组织。

【问】

你提到这个组织，你可以大概的可以说一下这个组织结构是什么样的吗？大概会怎么组织，从头到尾是怎么组织的？

【答】

组织，你是指？

【问】

文档的这个内容的组织。

【答】

你说PingCAP文档的组织吗？

【问】

嗯对。

【答】

这方面可能李琳会比我更了解，这个可以给你提供一个网址直接看就可以了。我阅读我们公司的文档，通常是通过Google。就是比如说我要做什么，一些安装前的一些检查，通常会遇到一些什么命令的话，我直接就搜索，就搜出来了，或者是我要一些什么，比如说我要安装某个产品，他需要一些什么样的配置，它配置模板是什么样的，我也是直接通过Google搜出来。

【问】

所以平时其实也不太关注他这个文档是怎么组织呢，是吧？

【答】

对对。但这方面主要是我对这个东西比较熟悉，所以我并不需要去从头看到尾。

【问】

那您可以举一些就是您用过的比较失败的文档的例子，就是用起来不顺手的吗？

【答】

嗯，我觉得这个非常带个人的偏见其实。

【问】

没关系，其实也可以说一下，就是说一下他们不好在哪里，就说一下自己用这个文档的时候遇到的问题，可以说一下。

【答】

嗯，我觉得。我觉得我用的感到非常复杂的一个文档是docker相关的。它里面的各种各样的术语太多了。如果你之前从来没接触过这个东西的话，你要阅读的话有点困难。然后另外一方面的话是他的文档，就比如说我执行某个命令的时候我产生了疑惑，就是为什么这个命令是这么做的，它为什么是长这个样子，那个参数是什么意思。然后当我搜的时候，其实是能够搜到他的官方文档的。但是他们官方文档并没有对这个东西做详细的阐述。就是说对于这个参数的含义并没有给出解释。应该是内容上的缺失。还有就是专业术语太多。

【问】

您看的这个文档，这个docker系列的文档是英文的？

【答】

是的。

【问】

那您觉得读起来是这个专业术语不太好理解，因为都是英文的专业术语，不太好理解？

【答】

其实我感觉像英文的专业术语反而会更好理解，我是从计算机技术这个角度来说的话，因为国内的计算机技术其实起步更晚，很多技术都是，术语都是从外面传进来的，所以一旦你去阅读国内的资料的话，如果他进行了翻译，那你能够搜出来，命中的范围就更少了。如果你之前接触的就是它本来的那个英文的那个名字的话，那么你能搜索，可选的话也就更多。

【问】

就是说这个计算机领域的专业知识其实是从英文传过来的，所以读英文其实更直接对吧？那您觉得您看这些英文文档有障碍吗？您觉得读起来困难吗？

【答】

好像还没遇到困难。

【问】

没遇到困难。觉得还是比较容易读懂的。

【答】

是的，但是这点的话，其实跟你用的操作系统有关系。当然这只是个usability的问题。比如说你用mac os的话，你如果遇到不懂的单词，因为它系统有个内置词典，所以你一下子就能够知道他什么意思。但如果是windows的话应该也可以通过有道这类的东西，插件应该可以实现这种功能。

【问】

也就是说其实也没有太多难的词，然后即使有也可以通过一些技术手段来翻译理解是吧？

【答】

是的。

【问】

那您觉得他这个英文文档本身，本身从语言来看，您觉得它写的是比较简单易懂还是说？

【答】

分类。如果像是API文档这种通常都写的，大多数人都能看懂，没什么问题，但如果是那种博客这种就很难说了，有些作者可能他语言习惯，比较习惯用那些词，然后刚好这个词我就从来没见过，那种表达方式也没有见过，或者谚语，通常这个博客里面很容易就会出现谚语，这种情况下你也没见过。那这种情况，可能阅读起来会有点障碍。

【问】

但总体上您觉得读英文还是没有太多障碍，还是比较顺利的？

【答】

是的。

【问】

下面就是想问一下您对于这个文档体验的一个期待了，嗯您觉得什么样的文档才是好的技术文档？

【答】

我觉得能看懂的文档那是最好的技术文档。并且你看懂以后不会有新的疑惑的那种，即使有新的疑惑也会能够立刻得到解答的，那我觉得这是很好的。我最怕就是遇到那种，就他假设你已经懂得某种东西，但其实这种东西很少人有人能懂的时候就会觉得很困难，因为我不知道你说这个东西是真的还是假的，还是说有问题的，那我还得再另外去求证，再另外去看这个东西讲的是什么。

【问】

他假设你已经懂了这个东西，但其实没有懂，是因为他没有进行一些更详细的原理的解释对吗？

【答】

对，或者没有给出一些reference，没有告诉我说这个东西是在哪里可以看到它的说明。

【问】

那除了容易懂，你觉得还有什么其他比较好的设计的点吗？也是可以再列举几个维度，大概总结一下一篇好的文档应该是有哪些特点。

【答】

如果是那种什么how to的这种文档的话，最好能够让我直接复制粘贴，这样就最好了，因为这种how to其实你平常也不会去看他，但你一旦看它可能是你想要用它的时候。因为计算机领域太多这种要安装要什么这些东西，就是步骤性的东西的话，如果能够复制粘贴的话，而且你打开打同一篇步骤的概率是非常大的，这种情况下有可能复制粘贴的话效率会高。所以针对how to这种文档的话，能够复制粘贴，并且第一次阅读的时候能够清晰的理解每一步的意思，它的含义，那么这个how to应该是非常做的非常好。然后像是原理性的文档的话，嗯，如果他能够有个什么类似于架构图啊，或者是有一些归纳总结这种的话会比较好。

【问】

就是说他图文并茂，用图片来辅助您理解。

【答】

其实并不需要特别多的图，关键是你的图得画得很准确，就是说关键的地方，比较难理解的地方，最好用图来表示。比如说你介绍一个系统以后，如果你把它的架构直接画出来的话，那我直接看这个图大概就能够明白是怎么一回事，但是你用纯文字的话，我就有点难懂，很容易忘掉。

【问】

那您说的那个总结性的语言也是说就是把它整个架构用一种很简洁的方式给表达出来，是吧？

【答】

对对，是的。

【问】

那除此之外还有其他需要补充的吗，比较好的设计的点。

【答】

嗯，还有一种就是我刚刚也提到，就是有一个例子，可以从这个例子出发以后就去发散，然后最终能够看到一种成果，这种的话也是了解原理性的东西的一个非常好的一个表达方式。

【问】

就是提供一个比较具体的形象的例子，让你能够快速的，比较容易去理解。

【答】

对，比如说我要解释rust里面那个内存安全是怎么做的？或者说我要介绍这个操作系统里面这个内存是怎么排列的。这样的话如果你硬说硬聊的话其实也能说，但是这个比较难懂，但是如果你说我要写个什么类似于wet之类的东西，那我应该怎么写，为什么这么写，它里面涉及到的内存排列是什么样的。然后到他整个生命周期结束，他又是什么样的，然后最终一步一步，构建出一个完整的wet实现的的时候，你就会既明白了它的原理，然后也知道了某些数据结构为什么是这么设计的，嗯，然后也非常有代入感。

【问】

那下一个问题就是您刚才也提到您写过技术文档，那是相当于是公司的这些文档，您觉得写的时候有什么困难？那难点在哪里？

【答】

我觉得最难的地方其实并不是一开始写的时候，那是后面维护他的时候。因为产品会在不停的迭代，然后每一次迭代的话，可能就会有功能上的一些细微变化，或者是一些不兼容的一些变化，这种情况的话，你得保证你的文档是最新的，就更新的，并且要保证不同的版本之间，不同的版本还得保留着，因为有可能有人用旧的文档，有人用旧的产品，他们就得看旧的文档。

【问】

就是觉得这个文档的更新比较繁琐？

【答】

对是的，而且我们还是一个多语言的系统，多语言的文档系统，不仅得写英文还得写中文。然后还要考虑把文档的事情纳入到整个开发周期里面去，所以整个过程是非常繁琐的。

【问】

那其实这是一个管理上的问题，就是从管理上目前还不知道怎么能够让文档快速的更新，能够跟上软件迭代的步伐？

【答】

嗯，我们目前的做法是每一次迭代的周期结束的时候，冻结代码的时候开始写文档。然后发布的时候必须要保证文档代码，还有各种各样的什么相关的周边的工具都一起发布。我目前的流程是这样的。

【问】

就是边写代码边做开发就已经边写文档了？

【答】

这时候问答其实要分很多部分了，比如说你设计文档，还有你们这个测试文档是吧，这些肯定是，甚至设计文档，你还没开始写代码就有了。然后测试文档的话是，你边做开发的时候，我们现在的测试文档是更提前一点，就是当我们写完设计以后就应该开始写这个测试文档，就是我们预期它应该是长什么样子的，然后我们要覆盖到哪些方面这些东西都是先写出来啊。

【问】

嗯，那最后一个问题就是有关这个文档的反馈的一个问题，就是您看过的文档里面，你会有看到过他在某个地方会提供一个反馈的途径嘛？比如说在文章的底部和文档的边上就有一个反馈的途径。

【答】

见过，他应该是主要两种形式的，一种是给你评论，直接评论他好还是坏。还有一种是给你点赞的方式。你可以点是不是有帮助之类的。

【问】

那有可以提问题的地方吗？

【答】

提问题的话，有些有，有些地方没有。但是我很少会去提问题，对，因为我相信那个问题应该是不会有人去看的。其次的话可能我的问题其他人已经遇到了，所以我更倾向于直接去搜，而不会去提问。嗯，那有过提反馈的情况吗？

【答】

有，嗯，比如说它某些文档他写的就不对，然后如果他可以评论的话，我就我就直接在这里评论，评论说正确的方式是什么样的。

【问】

那你这个评论主要也是为了让他能够不断的修正这个文档，让文档质量变好是吗？

【答】

对，是的。但主要是为了其他人看的时候不要再产生疑惑。

【问】

还有一个就是关于这个文档的交互设计的问题，就是前面我们访谈的开发者当中，他们提到的一些交互上的设计，就比如说他们用的文档里面有提供那种可以直接运行的代码，就点一下运行，然后它就运行，然后就可以看到效果，嗯，那您觉得从文档的这个交互设计上来说，您有什么期待呢？

【答】

我觉得这种可能得有一定想象力，从目前的使用体验来看的话，好像还好。

【问】

就是说这个交互其实对你来说不是最重要的。

【答】

对我更重视的是它里面的信息内容。

【问】

还有的同学在跟我做文档体验的设计嘛，比如说啊，就是我看很多程序员工作的时候喜欢听音乐嘛，如果这个文档可以有一些背景音乐的话，会有助于，程序员会喜欢吗？背景音乐，或者是用background music。

【答】

我觉得是一个非常新奇的想法，嗯。从我观察到的现象来看的话，大多数程序员工作的时候都会戴耳机，然后放他自己的背景音乐，就说你让他放自己的背景音乐，但他可能就不是很想会被其他的音乐去干扰。

【问】

比如这个时候呢，因为我们现在还在做这个研究，音乐对于大脑的加工能力的这个影响，会有一些推荐的歌单，比如说他在旁边有一个，你可以试试我这个看文档专用歌单，就这样的一种一些新的尝试，可能也会愿意试一试吧。

【答】

如果有这么个本文档专用歌单的话，那可能会有人有兴趣点一点，但是我觉得，我理解阅读可能不同于其他的一些信息传递方式，比如说像什么视频啊或者是音频这种。阅读应该是非常追求一个信息的一个密度的，就是我希望是我阅读的时候一目十行就可以得到很多信息，那这种情况下如果再有音乐的话，因为每个人阅读的节奏也不太一样，所以这个音乐是否合适呢？我觉得可能得需要多研究。

【问】

另外比如说啊，不是说这个开发者他们也挺喜欢跟其他成员互动的，假如这个文档有弹幕功能，你比如说你是昨天看过了啊，我觉得非常好，另外一个人过去在用这个文档，在几天之前有人看过啊，有一个弹幕功能，这个会影响程序员工作吗？还是说你觉得这个文档还多了一个社交功能，他可以跟其他人互动，点个赞。

【答】

嗯，我觉得这个取决于实现的效果。嗯，就是其实最开始弹幕出现的时候，也会有人认为它挡住了这个视频，但是最终还是看弹幕本身的信息量，还有你做的效果，可以把它做得更浅色一点会怎么样呢？也可以做到不影响用户嘛，但是我观察到有一些网站，它其实有另外一种功能是可以让一些辅助的一些功能，这种我理解应该是跟弹幕其实有类似。比如说在右边是吧，批注的功能其实还挺好的。嗯，特别是你看到一些晦涩难懂，刚好有一个批注，你就点一下看看，可能会有意外的收获，有这种新的功能上市。而且这个功能好像我在一些博客上也看到了。

【问】

批注也算是一种反馈的途径吗？我遇到问题我批注一下，其实也是给这个文档团队来做一个反馈吗？

【答】

我倒是很少遇到有在批注里面反馈的，对，可能批注里面更多的是表达对这段话的一些意见啊。

【问】

哦，刚才您提到了一个看视频，那您觉得您在入门的时候是更愿意看视频的一些，比如说有一些视频的教程来入门，还是说更愿意直接看文字来入门？学一个新东西的时候。

【答】

我更愿意阅读图文，不是纯粹的文字，纯粹的文字的话确实比较难懂啊。如果有图文的话会比较加快这个效率。但是我不太喜欢看视频，因为视频的话太慢了，就是他说半天的内容，可能我看了一句话就看完了，我觉得视频可能还是偏娱乐多一些，或者是一些科普类的那种，就是不需要追求时间的，就偏娱乐一点的那些，我觉得可能会比较好一点。