**P8**

2021-10-5 14:54:17

【问】

那就先大概的讲一下你的这个专业的背景吧，以及做过的一些开发的经历，大概是多长时间的？

【答】

我本科专业是XX工程，然后正好我们学校的XX工程也是非常实践的工程，就是一进学校就是做项目的那种，所以我们从，如果硬算我第1个项目的话，可能能够追溯到大一12月份，那次是一个组队的一次python比赛，python就是那种在一天或者两天之内写一个项目，那个比赛应该是我严格意义上来说的第1个项目，然后那个项目基本是我一个人写的，因为当时组内的其它同学可能也不太会写程序，基本是我写的，然后那场比赛严格来说算个人项目，然后团队项目的话是在第2学期，就是大一下学期，有一次个小游戏比赛。那个是我们4个人一起，应该是我们4个人一起写了一个游戏，那个可以算是我的一个团队项目。然后之后的话就是大二那年，就是做的项目就比较多了，我们课程里面会有那种规模相当大的一个，我们大二上的一个专业课，它的大作业就是一个项目，然后那个项目规模还是比较大的，就是我们也是4个人做完，整个项目代码可能代码有4万行，文档可能有10万字，这样的规模。然后那一年我还参加了一些组队比赛，嗯，那些项目可能也是在一两万行左右的代码，然后之后就是一直做项目，因为我们专业就是做项目的一个专业，然后到了大三暑假的时候我去了微软苏州实习，然后就在里面实习了三个月，也是负责一个内部监控系统的开发。到现在我研究生期间，我实验室也是一个非常工程的实验室，所以也是基本上是在做项目，但是研究生期间的这些项目基本都算是我个人项目，因为我找不到同学，同学也做工程，大致就是这样。

【问】

这样说你的开发经历，其实从大一算的话有四五年的时间了。那算是经验比较丰富的。然后就是想问一下你，你可以举一些比较具体的例子，可以回想一下你最近或者是以前学过的某一个新的工具，或者新的软件，新的技术，然后大概的讲一下你学习这个软件的整个一个过程，就是你从听说这个工具或者这个框架开始，然后去学会它，然后运用它来做一些开发的工作，或者是完成了项目，整个过程你是怎么做的。

【答】

行啊，这个其实我感觉，一般开发者的话学习路径一般是学新框架吧，可以这么说啊，学新框架的流程基本是固定的，就是提出了这个框架，然后去看它的官方网站或者GitHub，GitHub上面就会共享它的文档，然后文档里面一般就会，一般首先是一个getting started的部分，那部分就是给没有用过这个东西的开发者来学习，然后一般讲getting started的部分就是能够简单的了解这个框架是做什么的，然后看getting started的时候应该可以开始写一点东西了，我在写这个东西的过程中，可能还会去看一些更详细的，一些更细节的部分，就是它文档里面也会包括一些除了getting started，其它的具体某一块的文档的部分，就是当getting started不够的时候就会去看比较深入的那些部分，主要来说就是先看getting started，然后就边写边看其它部分。

【问】

相当于就是边学边用，相当于你是看了这个getting started的这个概览，然后了解一下大致的功能，做法等等，然后就开始用了，然后用用到的过程中遇到问题，然后再去看那个文档的深入详细的那部分对吗？

【答】

对对，然后如果文档不够的话再去网上搜一些，就搜出什么issue或者stack overflow上面的一些问答。

【问】

那你在做的过程中遇到问题，就是你刚才说的是遇到问题你会返回那个文档来重新看，那还有其它的解决的办法吗？

【答】

遇到问题的时候，一般来说就是看文档或者是在网上去搜这个问题，嗯，如果没法解决的话，我有时候会去看一下源码，但是一般是不会去看的，因为源码，因为每个人写的代码风格其实不太一样，你看源码一般也看不懂啊，而且有时候有的东西很复杂，你那个源码你也没办法看。

【问】

那就是优先还是自己去解决，不是一上来就问其它人问老师和同学，优先自己解决。

【答】

对。

【问】

刚刚你也讲了你的开发经历，然后你作为一名，也算是程序员嘛，你觉得在中国程序员和其它专业的人，其它职业的人相比有什么特征？

【答】

特征，具体可以说是什么样的？

【问】

很笼统的，比如说你们自己对自己的一个性格啊，职业、工作的习惯，生活的习惯、或者是学习的习惯都有哪些比较典型的？

【答】

我觉得可能一大特征是遇到问题先靠自己吧，就是在计算机世界里面遇到的问题一般都是你自己的问题，不会是别人的问题，所以遇到问题首先会看自己有没有错，然后如果发现自己无法解决的，就先自己去看一看，实在不行了再去一些问答网站上询问，然后可能其它跟人打交道比较多的那些职业，可能很多东西不是一个人能解决的，所以那些职业可能就是遇到问题第一反应可能是拉人拉个群一起聊一聊什么。

【问】

那从学习的习惯上来看呢？

【答】

学习的习惯上，我觉得虽然可能会知道一直学习这个概念吧，可能会强调这个概念，就是知道自己什么都不懂，知道自己应该去不停的追，不停的看整个业界的发展情况，不停的去看，就是新出现的什么东西，然后尝试去用一下。

【问】

就是说这个职业它要求是要不断的学习的，因为新的东西出来就很快，你要不断的学习才能跟得上。另外就是你们可能就偏向于自学对吧？

【答】

对，偏向于自学。因为新东西一般不会有太多资源能够让你去很简单的学习的，你可能会需要去啃一些刚出来的一些不能完善的一些资料。

【问】

噢，那就想问一下你，你觉得在你的开发工作中，技术文档是占据什么样的地位？它的重要程度如何？

【答】

从几个方面来看吧，首先是如果你在学习一个东西的时候，那肯定技术文档是最重要的，因为即使是开源项目也不会有人去看代码，即使这个项目是开源的，大家在使用这个东西的时候是不会去看它代码的。

【问】

是因为代码很难看懂嘛？

【答】

嗯，对，而且开始看代码的效率一般比较低的啊，一般就是在学习过程中肯定基本都是看文档的。然后如果是合作方面的话，就是你自己写的项目，然后想把这个项目推出去给其它人用的话，文档也是个比较重要的东西，因为你在写文档的过程中比你嘴巴说是多了一个思考的过程，你嘴巴说可能就是像现在这样卡顿，但是你写文档你会必须想每个计划，就是怎么把这个东西描述的简练又清晰这样。主要这两方面吧，程序员的工作，要么就是用别人的，要么就是自己写。

【问】

一个就是对于自身学习来说，看文档是效率比较高的一种方式，然后对于内部团队的沟通来说文档能够加强你的沟通的效率。

【答】

对对。

【问】

嗯，好的，哦，那你平时都会用哪些类型的文档，你可以就是想想自己在平时的学习还有工作的过程中，都有求助哪些资料？这些都可以算做文档。

【答】

其实刚才您说的那些文档其实都有看的，比如说，就是它们自己写的那种，就是用来入门的那种框架的文档吧，那种是特别需要看的，然后有时候，嗯因为你在用一个框架，这个框架可能会提供很多的API，每个API是做什么的，这个一般会有一个专门的API reference，那块一般不会一开始就看，但是有时候遇到的时候会看一下。

【问】

用到是指你要去看具体的参数或者某个函数？

【答】

对对，或者是看这个包提供了哪些功能，会在那里面看。然后用户文档就是面向最终用户的，这个在开发过程中其实也会看，比如说你在开发过程中还有一堆开发工具，那么对于这些开发工具，我们开发人员就是它的用户，所以就看这个用户文档，然后那些刚才说的论坛这些地方，其实你在搜一些问题的时候就会进这些论坛，因为这些论坛，因为它们也是某个具体的问题，可能有些具体的问题跟我遇到的问题比较相近，那么在解决这种问题的时候，看论坛的效率是要比看文档的效率更高的，因为文档它只是开发者觉得你会遇到什么问题，而不是用户遇到了什么问题，但是论坛的问题都是用户遇到的问题，很可能跟你遇到的问题比较像，那么论坛是一个比较好的，主要是这些。

【问】

论坛就是在国内像博客园这样的吗？

【答】

博客园，对，一般用的多的框架博客园里面会有一些，虽然看的很少，但是确实很实用的一些东西，然后，我其实一般是看英文资料，因为感觉中文资料质量确实不太行。

【问】

那英文的话就是哪个论坛？

【答】

主要是stack overflow，有的GitHub issues下面也会有一些问题。其它的redis也会有一些问题。

【问】

那你平时找到这些资料的途径主要是靠搜索引擎来搜？

【答】

对，就是靠搜索引擎。

【问】

那除了这个之外还有其它的方法途径吗？

【答】

我个人，因为最近我做的项目其实一般都是个人项目，所以一般都是靠搜索引擎来搜索，但是之前比如说跟其它人合作的话，遇到一些其它人负责的模块，还可以去问一下其它人。然后我之前喜欢加一些技术讨论QQ群，现在不喜欢加了，之前有时候有的问题也喜欢在那里面去问。

【问】

这个QQ群是由谁来维护的？  
【答】

也是一些技术人员。就是有的项目，尤其是国内的项目，它会在那个里面贴一个什么钉钉群QQ群的一个二维码，你扫进去，那么这个群里面的大家都是在讨论这个项目，有时候有的问题就在这里解答，然后国外好像也有这种群吧。我忘记那个平台叫什么了。国外也有这种开发者维护的群，然后里面全是关于这个群的一个讨论组，这个里面会有挺多的讨论组。

【问】

那这种开发者的群不算是官方提供的一个地方吧？

【答】

如果是从官方文档，如果是从文档进来的那个一般是官方的。一般里面也会有一些那个库自己的开发者在里面解答，比如说那个v2ex，是叫这个名字吧，国内的一个论坛，那个论坛里面，它那个就是开放者自己的一个讨论社区了。有的时候这里面会有一些讨论区，这种群就是开发者自己的群，就是爱好者加进去的一个群。

【问】

那你找到这些文档之后，找到这些资料之后，你会怎么快速的找到，快速的定位自己需要的信息呢，因为文档有可能是篇幅比较长的。

【答】

一般定位信息也是靠搜索引擎来搜，因为这种文档一般它能够被搜索引擎检索，所以我遇到问题就是直接在搜索引擎里面搜，搜索引擎能带我进去。然后实在不行了，可能就是文档，现在一般是用框架式文档，有一些文档框架，然后它部署成一个静态网页，然后这种一般会提供一个搜索的入口，一般有时候也可以这样搜进去。

【问】

我的意思是打开了这个文档的页面之后，你的一个阅读习惯是怎么样的？是从头到尾读还是说就是通过一些其它的手段来快速的定位一些关键词。

【答】

文档其实不会用来读，就是按我之前说的那种流程，一开始先看getting started，getting started的部分肯定是一步一步来吧，都这样，但是到更详细的部分不会去读，就是我用到什么，我需要什么东西，我先去搜索引擎上面去搜，它会直接把我带到文档的某个页面区，我不会没事去读它那些详细的部分，因为没有用，知道吗？

【问】

嗯，那下一个问题是，你刚才也提到你读过一些，你主要是读一些英文的文档嘛，然后就相当于你读过一些国外的文档，然后国内的你应该也读过了，那就是从你读过的文档的经验来看，你觉得它们的质量怎么样，国内的国外的你都可以举一些例子，比较好的例子，比较典型的写的很好的文档的例子和写的很差的文档的例子，然后你可以分别说说它们好在哪里，差在那里。

【答】

嗯，还是先举国外的吧，我觉得写的文档比较好的就是微软系列的东西，因为微软自己有一个MSDN，叫docs.microsoft.com，对应该叫这个，这个是微软自己维护的一个文档网站，然后这个里面就会有微软所有的产品的文档，比如说它的那个网页框架，那个东西的文档非常详细，它既有那种get started的部分，就是你可以一段一段的，可以从0开始学习一个框架，它还有那种给稍微热门一点的用户，就是我具体想要什么功能，比如说我具体想实现数据库的链接，那么它就有一个部分教你怎么做这一个功能，就是很具体的一个功能，它有这么一块，这是给终极用户，然后它也有那种API reference那样那种类型的文档。而且微软的那套文档的话，它的那个覆盖面是非常广的，基本上是你遇到什么问题就能在上面搜到。然后再举一个例子吧，这个例子就是，比如说Spring，就是Java的一个IOC库嘛，以及它的那个springboot做web，那个框架其实也是一个用户非常广的框架，但是其实我觉得没有什么人读它的文档，因为我确实，它的文档我觉得它就差一点实用性，就我可能看过几眼它的文档，但是我印象不深了，就是因为在springboot上面遇到的问题，一般你会去搜索引擎上面搜，不会直接跳到文档里面去，会跳到一些论坛或者stack overflow，或者国内的文章，一些知乎文章上面去。因为它的文档没有覆盖到开发者想要的东西，就是这样，所以它的实用性不大。然后微软的就是，它会针对开发者想要的东西，提供一个这样的文档来给你看，这样子。国内的话，因为这些框架主要是国外的，国内写的比较多的框架，比如说XX的开放者文档，那个我看过几页，怎么说呢，XX的开发者文档，我觉得算是还可以了，就是我想要的，我想要看的那些功能部分它还是会写的，嗯，我觉得XX的开放者文档还是可以的。国内的其它的文档我觉得可能就不好说了，因为确实没怎么用国内的东西，然后我觉得比较不好的吧，是一个叫swc的东西，我可能偏web前端一点。swc是一个用rust写的前端的一个库，它可以用来转译js，这个库它的文档我真的，就是连它的getting started部分我都跑不下来的，它的文档实在是太简略了，什么都没有。

【问】

意思是太简略了，跑不下来是因为它没有给例子吗？

【答】

是给例子，但是除了很简单的例子之外就没有进阶一点的部分，就是我把getting started跑下来之后，我想进行一些复杂的一些配置什么的，它都没有写，所以是这样。然后跟它对应的，另外有一个叫esbuild的，也是跟它差不多的，可以用来打包js转译这种东西，是国外的一个框架。那个东西它的文档就很详细，它就有很详细的getting started部分，然后到每一部分，比如说我要加TS支持，我要加入react支持，这种部分，它对react是怎么支持的，对js是怎么支持的，它都有详细的一篇文档来说。那么这两个框架其实它们的领域非常相近，但是它们的文档就是一个天上一个地下。

【问】

你能大概总结一下这些点吗，你刚才说的点其实是比较笼统的，我看主要是在内容上面，大概总结一下文档，它的内容好在哪里，不好在哪里。刚才你提到的内容上它会比较详尽，会提供一些例子是吧？

【答】

嗯，对某一个功能类型的例子，这样说吧，一般来说软件框架都会有getting started部分，这个部分一般来说，就算我自己写框架我也会写这个部分，但是它的文档好与坏就是在getting started之后，有没有足够的文档让你更深入的了解和使用这个框架。因为你用一个框架，你不可能只用它的基本功能啊，你会有它的高级功能。那么好的文档，比如说微软的文档，它会对你可能用到的每一个功能，它都会写一篇文档来教你怎么把这个功能集成到一个项目里面。那么不好的文档可能就会差在这一部分，就是我想用更高级的功能这部分。

【问】

然后想问一下你平时读到过的这些文档里面，如果发现它有一些问题啊，你能找到给文档的原作者反馈的途径吗？

【答】

这个如果是开源项目的话，现在的开源项目的文档一般是独立维护的一个项目，然后那个项目一般会独立成一个GitHub repo，如果想对这个repo反馈的话，就往这个repo里面发issue就行了，我自己还改过一些。

【问】

那除了开源项目，那种非开源的项目有反馈的途径吗？

【答】

这个我想想。这个我可能没咋用。

【问】

就比如说国外的一些产品，一些框架，它的官方文档里面会有反馈的途径吗？

【答】

对的，一般这种文档它自己也是开源的，也是按照开源的那套反馈和贡献的流程走的，就是在GitHub发issue。

【问】

那只要是有它有反馈的途径，然后你发现它有问题，或者你有什么疑问，你都会愿意去提一些反馈一些评论是吗？

【答】

如果发现我文档看不懂的时候，首先都是先去查，然后再去问其它人。然后实在有问题才去发issue。

【问】

那如果是就觉得哪里写的不好或者哪里有错误的地方你会给它提意见吗？

【答】

这个我没有，但是有时候，我还是提过一点，但是一般不会提的，那些东西是我觉得举手之劳就可以去改一改的，就觉得没有太大的必要非得花这个时间去提意见。因为提意见，你得组织你自己的语言嘛，我其实还是比较懒的。

【为】

那你有这种动力吗？就希望你通过提意见，然后让这个文档变得更好，然后也帮助其它人规避你遇到的类似的问题，有这种动力？

【答】

这种记录下我遇到什么问题，然后我解决的，我会写博客，我是更喜欢这种途径。因为写文档这个事情呢，你得把你的东西提交到别人的项目里面，别人的仓库里面，那么你就得保证你自己的东西和别人写的东西是契合的，比如说你写的语言至少看着不那么Chinglish。然后因为我们本来就有语言不一样，可能写出的东西别人可能不咋看得懂，这个其实看很多GitHub上一些中文开发者写的英文文档也是有这个问题的。我自己有时候，比如说我可能看一两年前我自己写的英文资料，我可能就觉得啊我写的什么东西，所以就不太敢和愿意去贡献英文的一些文档。

【问】

那你提到语言的问题，你平时看的应该比较多的还是国外的一些文档，你觉得从阅读上来看你觉得语言是一个问题吗？你会觉得有不太能够理解的地方嘛？

【答】

这个反倒我觉得，因为大家都是用英文编程的，看英文文档可能比看中文文档效率还高一点，因为那些概念也不用再重新翻译一遍，但是阅读速度上会降低一些，但是还是更愿意去读英文的文档，因为觉得更权威更靠谱。

【问】

那你自己有写过技术文档吗？

【答】

应该有写过的。

【问】

可以大概的展开来讲一下你写过的文档的例子吗？是什么项目当中写到的，是什么样的文档，然后里面包含哪些东西？是给谁看的？

【答】

就是这样子，我大二的时候，我们当时有个组队项目，那个项目里面不是说有10万字的文档吗？这种文档很大部分其实是给老师看的，因为当时也有一部分是用来给大家互相交流，嗯，比如说，那个项目里面是个标准的一套瀑布式开发流程，就是首先是需求文档，就是你得搞清楚你这个项目是要做什么，这个东西我觉得非常必要的，因为你自己做的什么都不知道，这个肯定是真的没法做，接下来是设计文档，就是架构设计，就是整个大的架构是怎样的，然后是详细设计文档，就是按瀑布式开发流程来说的话，那个就是一些类图，数据表，这些都属于在这里面，然后后面是测试文档，测试文档那个时候我觉得没有什么用，然后最后是一个用户文档，就是面向用户的，就是把这个功能，你这个软件的主要功能和流程写在用户文档里。比较重要的文档，我觉得是就是刚才说的那些需求，架构，详细设计以及最后的用户文档。当然还有一些其它文档啊，比如说什么测试文档，我觉得那个就是应付老师的。

【问】

这个主要就是在团队开发内部，就是给老师还有给队员看的。

【答】

对，我觉得能起到主要作用的还是需求和设计文档，设计文档分为架构和那个详细设计文档。详细的设计文档就会包括什么接口文档这样的，我在这个项目里是画在详细的设计文档里面。这种是个比较完整，就是比较大的一个项目了。但是后面，比如说我自己写了一些GitHub的小库，那种东西一般也不会分的这么详细，比如有一个库它是做什么的，一般就写一份文档，然后就先介绍你这个东西是做什么的，然后就写一块getting started的部分，就是教别人怎么用，然后写一个API的一些流程。不对，接下来部分应该是详细功能，就是用户想做一点更高级的功能应该怎么做，这个部分，然后最后就是一个详细的API reference。一般就是这种情况。

【问】

那这种文档一般就是类似于readme这样的，就放在那个仓库里面。

【答】

对，短的话readme就够了，那长的话就用它那个wiki的功能，或者专门在repo上面放一个DOS目录，然后里面就放了一些文档。

【问】

那你觉得在写这些文档的过程中遇到的困难的地方是什么？

【答】

遇到的困难我觉得，我觉得有一点是比较重要的，就是文档和代码的契合度，因为你东西在更新，那么除了代码更新你的文档也要更新。这个尤其是很多质量不好的文档，它一个质量不好的点就体现在这个地方，就是它的文档落后于代码。这个是比较复杂的，所以现在我自己其实也遇到过这种问题。然后现在在我自己的项目，就是我现在在实验室负责项目，一般采取的措施就是尽可能少写文档，就是能够生成的文档尽可能生成，对自动化生成。比如说那些接口，我觉得是没有必要先用代码写一遍，然后再用自然语言写一遍。我现在是采用工具自动帮我生成接口文档，这样挺好的。

【问】

都有哪些工具？

【答】

RESTful API一般就是有smart doc这种工具，它能够自动根据你代码里面提供的API point，然后它会自动分析这个代码，然后生成一个网页，这个网页就是你这个项目里面所有的接口。这样子就能够一部分解决文档落后于代码的问题，但是因为自动化生成它也有它的局限，就是接口部分现在做的比较好，但是其它的除了API reference，比如说其它一些新功能，比如说你加了一个新功能，你要把这个新功能介绍给其他人，教他们怎么用，那么这一部分还是得自己写。嗯，我看到的一些开源项目它会这么说，它会说，比如它介绍一个功能，但是这个功能有一个非常复杂的规则，它懒得写，它不想写，它不想把规则一句一句列在文档里面，它就说那你自己去看测试目录，它就让你自己去看项目的测试用例。我自己虽然看到这个比较烦，但是我觉得这个方法确实也还是比较好的，因为测试用例其实也是比文档详细的，而且它可以很大的减轻开发者写文档的压力，因为懂的都懂，大家确实不太喜欢写文档。因为其实很多情况下，写文档的时间是比写代码要多的，就写的比较累，就这样。

【问】

那最后一个就是想问一下，如果有一个比较理想化的状态，就是有一个质量很好的文档都能满足你的需求，那你觉得这个文档应该是什么样子？就是说你期待用到什么样的文档？你可以列举你期望中的那种比较好的文档的维度。

【答】

其实一个比较好的文档，我觉得，其实跟我们之前说的那种，就是我觉得好的文档的几点，那几个部分比较类似，就是你首先得有一个getting started部分让我开始上手，然后给我借，然后下一部分是给我介绍每一个常见功能怎么用，然后最好还有一个FAQ，就是开发者觉得开发者觉得可能会遇到的问题，列一点出来，怎么解决这件事，然后最好这一部分开发者还能够不停的加上去。

【问】

就有一个提问的类似于社区的地方。

【答】

不是社区，现在说的还是文档的一部分，就是以Q&A格式组织的一块，就是常见的问题了，然后这种东西开发者最好还是能够维护一下，就是如果开发者接到什么反馈，反馈比较多的问题能够往上面放一下，然后API reference这部分最好是要有的，然后最好是把每一个接口的介绍写详细一点。然后，然后除了这种严格意义的文档之外，最好能够有一个讨论社区吧，不管是什么工具，最好能有个有开发者在的好的社区，然后最好这些开发者也能够在各种论坛上面去活跃一下，就看到有人问这个问题也去答一下。还有就是代码里面的注释，代码里面的注释我觉得也是可以算文档这部分，这部分我觉得也是写得越详细越好。然后就是代码里面的注释和API reference这两个部分一般是1对1的，一般是会有从代码注释生成API reference的工具，这部分越详细越好啊，然后最好还能够提供示例，可以在API reference中提供一些简单的使用方法，比如说这个函数怎么调用，然后最好在文档那里面也可以提供几个示例项目。

【问】

这个示例指的是它提供了具体的实现的代码吗？

【答】

就是怎么用这个东西，你就写一个很小的项目，虽然比较小，但是像这个网站架构它主要实现了这个功能。

【问】

除此之外，你刚才讲的大部分都是关于文档的内容本身的，那么除了内容之外，比如说对文档的这个结构有什么期待？

【答】

文档的结构，就是刚才说的那几个部分。

【问】

你觉得大部分的文档都是按照这个逻辑来写的。

【答】

我觉得这个逻辑还是可以的，就是能够兼顾各种级别的用户，各种级别的开发者。

【问】

那你们阅读的时候，从文档和人的交互上来说，就比如说一些官方文档它的UI设计等等，你觉得有什么可以改进的地方？

【答】

比如说可以加入一些，我觉得如果有那种示例工具，就是你可以很快点一下就能运行，比如说那个kubernetes的文档我觉得它就非常好，它不仅有我刚才说的给入门的人用的，也有每一个部分，每一个模块，每一个功能点的文档，然后它每个文档里面它还会提供那种prerun，就是你点一下那个，然后它会给你开一个虚拟机，这个虚拟机上面就有一个k8s集群，然后它会指导你输什么命令能做什么，然后你可以实践一下，在打开一个控制台，输入一下看看能打出什么，然后我觉得这个体验是非常好的，k8s本身是一个非常复杂的东西，它有了这个东西之后，学习体验会好很多。

【问】

你觉得这样设计的话，是会让这个阅读的过程更直观，更容易理解？

【答】

对，因为我觉得就是你光看东西是没有用的，你得边实践边学习，那么它这种其实就给你提供了一个很好的实践的机会，如果没有这种东西的话，你可能就是得自己先部署一个它的环境，开个项目，然后它写什么，你在自己项目里面写什么。这样对一些开源其实不是什么问题，因为那些开源框架很简单，但是对于K8s这种本来部署就很复杂的东西，你如果说想学习这个东西，还得自己先学习怎么去部署，其实就成本比较高。

【问】

还有其它要补充的吗？

【答】

可能还可以说一下团队内部文档的一些相关的。我刚才说的这些东西都是那种技术框架，技术方面的东西，不太涉及业务方面，但是实际工作中呢，其实很多东西复杂的是业务上的一些概念，比如说概念流程什么的，比如说当时在微软实习的时候，其实当时用的技术框架跟我平时用的差不多，但是我还是花了很长时间学习，在学习什么，是在学习它们内部的业务，比如说它们内部的一些业务概念啊。这个其实也有几年了，有点忘了。这种业务概念在微软内部，我的那个组是用了一个共享的one note笔记本，然后那个里面就会介绍很多它的一些业务概念，然后会给一些文档，一些内容。一般这种公司内部的这种业务相关的文档呢，其实它就是非常偏自然语言的东西了，很多时候我觉得这种东西也没法怎么自动化，而且它和每个公司自己是强相关的，你也不好去统一。我觉得如果想，因为我只有微软那一段时间，我还不知道其它公司怎么样，但是按我的那些同事反馈，可能一般这种公司内部的文档质量可能都不太高，如果想提高我们的质量的话，我觉得这种公司内部的业务文档其实是需要提高的一部分。

【问】

这种业务文档它主要的功能是方便组内的人员之间的工作交接、沟通是吗？

【答】

可以这么说吧，比如说来了个新人，公司里面有很多概念，或者公司自己的框架，然后那这些人他们的唯一学习的资料就是这个文档，然后除了这些文档就只能去问公司内部的人。它就不像这种开源的项目，其实用户面就比较广，资料就比较多。那么对于这种学习需求，对公司内部文档的质量的要求就特别高，但是对这种文档的话，在公司内部的话，其实从另一方面来说，你不懂的可以去问，这个其实也是一个比较好的方式。但是如果那个文档能够更详细的话，其实对于公司，对于团队内部和公司内部自己的各种工作的交接都是会有好处的。

【问】

除了公司内部的之外就没有其它要补充的吗？

【答】

差不多。

【问】

语言上，比如说国内的文档，从语言上你希望有什么改进吗？

【答】

嗯，我觉得其实都还好，因为大家都知道文档应该是更加简练，越简练越好，我觉得其实做得都还可以。

【问】

那问题就问完了，没有其它的。