**P9**

2021-10-6 14:35:31

【问】

好，那我们现在开始吧。第一个就是想让你介绍一下你的专业背景和你做过的开发的经历，比如说你可以说一下你从什么时候开始做开发，大概做了多久的开发。

【答】

我先介绍一下我的专业背景。我是XX大学XX学院XX方向的专业，最早的项目经历应该是来自课程项目。第一个项目是Java课程的课程项目，要做一个在线的判题系统，这是最开始的开发经历。首先，我们要实现核心的功能，然后再给这个项目添加前端，嗯，然后再添加一些用户的管理，比如说注册登录登出这些附加的功能，这是最开始做的项目。后来参加一些比赛会做一些数据分析的项目，数据分析的工具也分也用了很多种，比如说用Python，比如说用r，这些数据分析类型的项目，它相比于课程项目来说其他难度更简单一点，主要是对于给定的，更像是数据挖掘，对于主办方给定的一些数据，你在其中发现一些规律，并用这些发现的规律来进行对未知数据的一个预测，嗯，对再之后的项目经理就是科研项目，还有的毕业设计的实习的项目。我参加的一个科研项目主要是为开源软件贡献代码，做的事情其实不是一个系统的项目，只是在已有的项目的基础上进行一些漏洞的修复，添加一些新的功能，更像是维护一个项目。现在是在实习，其实越接近工业界，你做的项目越很难是一个完整的系统的项目。给你的任务就是维护一些已有的项目。这就是我现在在做的事情。

【问】

相当于你的开发经历有两年以上的时间，对吧？

【答】

可以这么说。

【问】

然后有在学校做过的开发，也有在公司做过的项目。

【答】

对。

【问】

嗯，好，那你可以回忆一下，你最近或者是之前吧，之前有学过哪些新的工具或者一个新的软件，新的框架，然后可以大致讲一下这个学习过程，从你开始听说这个工具、这个框架开始，到学会它，完成一些真正的开发工作，这个过程是什么样子的，你可以详细地展开讲一下。

【答】

我可以讲一下我最近的工作用到的一个框架，叫做ABP，是C sharp语言的一个开发网站的框架，这个学习过程，因为我对整个C sharp语言都不是太熟悉，所以第一步是先从官网上学习语言的基本语法，然后也看了几本书，就简单地入门地介绍这个语言的特性的几本书，然后对这个语言有简单的掌握。怎么判断有简单的掌握呢，就是你能看懂代码，然后也能写一些代码。当我具备这个能力之后就开始看ABP这个框架，主要还是看官网上给的官方文档，根据它的那个实战的案例来一步步地进行操作。我为了让自己更熟练，我会把示例代码都手敲一遍，加强记忆。然后具体时间工作，我还没有把它运用到具体的工作项目种，还没有来得及，但是我觉得我已经能够看懂这个框架的代码，能够理解它的一些结构和使用方法。

【问】

那你可以再具体地讲一下这个C sharp语言它的官网上的，应该是它的介绍文档吧，它的文档是什么样子的？都包含哪些内容。

【答】

首先开始最简单的都是一个hello world的一个基本的应用程序，教你怎么把信息打印到控制台上这样的操作，然后接下来应该会讲这种语言的数据结构，包含字符串、数字，然后接下来会讲这个语言的一些逻辑结构，比如如何进行判断，如何进行循环，最后会讲一些比较高级的应用，比如说这个语言怎么做一些更高级的功能，比如说网络通信，比如说多线程这样子的东西。

【问】

那它是会提供很多的示例来教你怎么做是吗？

【答】

会提供实例，而且那个网站文档做得还挺好，它提供那种交互的实例，就是说，它自己网站提供你一个开发环境，你在那个网页上输代码，点击运行，就能看到运行的结果，我觉得很友好。

【问】

下面就是想问一下你对技术问答的认识，你觉得技术问答再开发过程中扮演什么样的角色，它重要吗？

【答】

我觉得每一个开发者在开发过程中肯定要接触各种各样没有接触过的东西，不会说接触的东西都是自己学过的，在课本上见到过的。你面对一个新的工具，新的语言，其实最有效的办法就是看它们的开发者自己写的开发文档。因为我觉得它们的开发者才会更加了解这些语言的特性，然后才权利，也不是说有权利。他们写的文档才是最权威、最详细，然后事无巨细的。根据他们的开发文档才能最快地掌握这个新的语言、新的工具。

【问】

所以就觉得它对于开发者来说还挺重要的，是必须要参考的一个东西。

【答】

对。我也见到过一些写的很差的开发文档，这其实会很阻碍这个工具在别人那里的一个受欢迎度，如果大家都觉得这个工具很好，大家为这个工具做宣传，然后新的开发者过来了发现这个文档写得很差，你根本不知道怎么用，其实也会阻碍这个工具得流行。

【问】

下一个问题是你都用过哪些类型的开发者文档，除了这种官方的文档，还有什么？

【答】

资料的话，除了官方提供的还有一些，有一些书籍会提供一些在线的章节供大家阅读，我觉得能出版成书籍也能证明它的质量，但是它跟官网的开发文档的区别是它往往是滞后的，它往往是几年前出版的，所以不包含最新的一些特性，对，所以这是我觉得除了官方开发文档之外的第2个可以参考的一个教材。

【问】

那你在遇到问题的时候会看文档吗？在遇到问题的时候会怎么办？

【答】

首先是看文档，但是文档里一般很少包含一些常见问题的解决方法，然后就会去一些网站，比如说stack overflow，比如说GitHub上面，去谷歌搜索，最好是用英文搜索，然后看看别人是怎么解决这些问题的。

【问】

就是在遇到问题的时候也会看这种技术博客和程序员社区里面的问答是吧？

【答】

对对。

【问】

那你是怎么找到这些你需要的技术文档的？主要是通过搜索吗？

【答】

对，主要是通过搜索，因为你在使用一个语言、一个工具之前，你肯定要下载，就是找到官网进行下载。其实你能很明确地分辨哪些是官网，哪些是那种衍生的网站。我就会从官网的下载页面跳转到它的首页，因为基本上每一个工具都会有它的文档、documentation的一个链接。

【问】

除了直接到官网下载或是通过搜索引擎搜，还有其他途径能够找到你需要的资料吗？

【答】

比如说有一些软件有一个内置的help，可以查到它的一些文档。

【问】

也遇到过这种类似的？

【答】

哦，对对对，你直接在工具里面他就有包含它的那个文档。

【问】

嗯，那你找到这些文档之后你怎么去阅读？有可能这个文档它内容非常多，非常长，你怎么快速找到自己需要的信息呢？

【答】

因为一般大部分文档都有一个getting started，就是让人能够快速地上手这个工具的一个快速的教程，一般就看那个快速教程上手，然后继续去看，如果它这个文档非常复杂的话，你就看完这个快速教程之后上手做开发，然后遇到不会的地方再来看一看这个详细的文档。

【问】

就是说看完这个getting started之后就基本上能够上手去用它了，只不过是磕磕巴巴的，然后需要做深入的工作的时候才会回来看文档？

【答】

嗯对的。

【问】

然后你应该也读过一些国内或者国外的文档吧？那你可以举一些例子吗？可以举一个比较典型的例子，比较好的，然后说一下他好在哪里，然后再举一个你觉得写得比较差的，然后说一下它不好在哪里。

【答】

比较好的文档，比如说python语言的官方开发文档我就觉得写得挺好的。它这个官方文档除了介绍这个语言的各种用法之外，它还包含了一个叫best practice的东西，就是开发指南。比如说你这个变量命名应该怎么命名，这个代码风格应该怎么做，就是更详细的这种东西，能够更详细地来指导你代码的开发。

【问】

它是提供了很多例子是吗？

【答】

对，详细的指导的例子。

【问】

那除了内容本身，这个文档的结构编排得怎么样？

【答】

文档结构编排也是由浅入深的，而且它还有这个，比如说你查一个函数的用法，你点击就能进入这个函数的源代码里，所以我觉得就很方便。

【问】

那这个文档是放在官网上的documentation的中心是吧？它有一些交互上的让你比较喜欢的设计吗？

【答】

它并没有提供一个交互式的开发的工具，但是它会，我有印象有些文档会在底下提供一个让读者做的思考和讨论，还有一个小的任务。比如说它今天教你怎么用，它底下就会举一反三，让你尝试完成一个小的任务。

【问】

用它教你的这个知识点去尝试完成一个小的任务，对吧？

【答】

对。

【问】

你看的这个python的文档是中文的还是英文的？

【答】

英文的。

【问】

那你觉得读起来在语言是有障碍吗？

【答】

我觉得它用的词都是比较简单易懂的，我觉得没有太大得障碍，你只要掌握一些英语的常用词就行了，没有一些很生僻的词汇。

【问】

读起来能读懂，那阅读的速度怎么样？

【答】

阅读的速度应该是正常水平。

【问】

就正常读英文的那种水平是吗？

【答】

比正常读英文更快一点，因为它用词实在是很简单。

【问】

所以这也是你喜欢它的一个点。

【答】

对对。

【问】

那你能举一个写得比较差的文档的例子吗？国内国外的都可以。

【答】

写得比较差的，我想不到具体的例子，因为你觉得文档写得很差你可能就不会去用它，它的名字可能就消失在你的脑海里了。

【问】

那你可以总结一下你见过的比较差的文档的一些特点，一些共同点吧。差在哪，那些点。可以把几个差的文档的不足之处都综合起来说。

【答】

我觉得比较差的文档一般维护的时间都比较少，一般要么就是好几年前，就是五六七八年前的文档，它从来不修改，就一致放在那里，无论你工具怎么迭代，它那个文档就是没有动过。还有就是一个最新开发出的工具，它的文档可能来不及写，然后它就放上不太全的文档。我觉得它们的维护时间不够长是比较差的文档的一个特点。第二点就是语句不通顺，有一些国外的文档会提供中文的版本，我也很喜欢看中文的版本，因为咱们是中国人，所以看中文肯定更方便，但是它提供的那些中文版本甚至不如你直接阅读中文的原文，它可能是机翻或者是找翻译水平不高的人进行的翻译，所以中文看起来非常的难懂。我觉得这个翻译有还不如没有。它可能会错误地提示你，这是第二个特点。我暂时想到的就这么多。

【问】

这几个特点已经是成为阻碍你把它继续看下去的一些差的点了，是吗？

【答】

对，我觉得尤其过分的是那种过时的文档，你跟着它文档的教程一步一步地走下来，你发现它出来的结果你自己怎么也出不来，因为它的文档是旧的，并没有及时更新，所以就很具有误导性，你就会觉得很沮丧。自己明明一步一步地按着它的来了，还是出不来结果，还要自己一步一步地修改，找错误，修改错误。

【问】

下面一个是关于文档的这个反馈评价。你有没有见过这种文档，在底部或者在某个地方给你提供一个反馈的途径的文档，就比如说它会问你这个对你有帮助吗？或者你喜不喜欢，或者是你有什么意见，觉得哪里有有错了可以给文档的作者提供反馈的这样一个途径，有没有见过这种？

【答】

有见过，有两种方式，第1种是在文档下面，每页文档下面设一个评论区，你可以发自己的问题，然后作者就会进行答疑。第2种是开源的文档，它直接在文档每个页面都有一个GitHub上的链接，你点进去你可以在他的GitHub仓库里进行提问，然后作者看到的话也会进行答疑。

【问】

是通过发issue来提问是吗？

【答】

对的，你也可以自己，你觉得这个文档写得不够好，哦，我觉得开源的好处是读者觉得这个不够好，它可能会自己去修改，去贡献，给这个文档贡献代码。

【问】

那你又提过评论，有提过意见吗？

【答】

对，我有提过意见，是因为这个文档实在太旧了，就是我刚刚提到过的，我根据文档的步骤并没有得到应有的结果，就会在GitHub上或者是评论区留下我自己的疑问。

【问】

嗯，那你提意见的主要的目的是帮助他改进质量吗？

【答】

嗯，对，也是让我自己能够更好的使用这个工具。

【问】

嗯嗯，好明白了，你自己有写过技术文档吗？

【答】

我自己有写过部分文档，只写我自己做的那部分的使用的文档。

【问】

这种文档是给谁看的呢？

【答】

给开发者看的，还有给使用的人看的。

【问】

是从同一个项目组的人吗？

【答】

其实主要是给使用的人看的，对也要给同一个项目组的人看，要不然你做的这个东西他们不知道怎么用。

【问】

嗯，你能大概说一下这个你写过的文档它都包含哪些内容吗？

【答】

我写的一般都是一个工具的一小部分，首先我会写这个工具的名字，然后放这个logo，然后第1部分我会写我们开发用的一些技术，比如说用了Python，比如说用前端用了flash，后端用了啥，然后第2部分会写这个环境，就是我们的这个开发环境应该怎么安装。就是我们这个工具应该在什么样的环境运行，比如说python的版本号是3还是2，比如说各个工具的版本号，比如说系统，我们只能在Linux系统下使用，不能再Windows系统下使用。第三部分我才会简单介绍这个工具的一些使用方法，包括它的一些参数啊，你怎么调就可以达到什么样的目的。一般其实有这三步，其他的开发者应该就可以完成一个，嗯，应该就能把这个工具用起来了。然后第4步，如果我当时写的很详细的话，我就会放一些更加高级的用法，如果写的时间比较短，比较急促的话，我就会放一些使用的截图，然后最后会放一些版权信息，比如说这个工具版权属于谁，这个工具使用哪个协议进行开源，你如果要商用这个工具应该要先跟我们联系怎么怎么样。

【问】

那你在写这个文档的时候有遇到什么难点吗？还是说写的过程就比较顺利？

【答】

嗯，如果是我个人的项目写文档其实就很自在，我就可以按照什么风格写都可以，若是给别人的项目写文档的话，你就要先参考别人项目其他文档的风格，要不然你写的文档独树一帜就会显得很突兀，你要参照他其他项目的风格，然后你还要参照，因为你是给别人的仓库提供代码，提供文档，你还要参考他人的意见，有的严格的那种公司或者严格的一些开发者，他就会提一些意见，然后督促你完成这个版本的迭代，比如我之前有写过文档，然后来审核文档的人就会觉得我这个文档写的不够面向新手，就显得步骤都太高级，应该写的更加详细一些。嗯，对，我就会加很多详细的步骤，每个步骤都还要解释一下，为什么要这样子做，然后放那种使用截图的地方，我会把你是如何运行的这个东西，做一个动画的一个动图，这样子可以更好的给新手解释。

【问】

那总结下来一个难点就是风格要一致，风格要统一，然后另外一个就是要考虑到这个读者，如果读者是新手的话，还要照顾到新手可能不是很熟悉，要尽可能的写的详细一点对。

【答】

对。

【问】

嗯那下面一个问题是如果有这么一个理想化的状态，就是你能够用到一份质量非常好的文档，那你期待这份文档是什么样子的？你希望或者说对你来说一份好的文档，他应该具备哪些优点？你可以大概的总结一下。

【答】

可能我觉得这个文档不能写的太生硬，除了要有怎么教你用这个工具之外，最好还要放一些自己在开发过程中的思考，自己的一些哲学，这个工具为什么好，它好在什么地方，大家为什么要用它，这种东西的加入会觉得让这个文档更加的有人情味，也不是有人情味，就更加的有个性，就会觉得很亲切，大家学习的不只是这个工具怎么使用，还能学习到你在开发过程中的经验，开发过程中，对，就这样子。

【问】

你是说希望比如说官方文档里面也会加入这些东西吗？

【答】

嗯，对可以加入，就是比如说你这个文档里一个功能，为什么要加这种功能，这种功能的好处在于哪里，也可以加上一些这样的内容。

【问】

就是说不光告诉你怎么做，然后也要告诉你这么做的原理是吗？

【答】

对，这样子别人也能参考你开发这个工具，这个特性的这个方法，也有助于开发者成长。

【问】

那除此之外呢还有吗？

【答】

我觉得语句还是要通顺的，你不能你如果有多个语言的话，你不能就用简单的机翻，你最好招募专业的翻译者进行翻译，就最好是有双语版，然后双语版翻译的质量也要保证。因为我能理解多语版他肯定是想推广到更多的读者嘛，但是你如果翻译的磕磕巴巴其实更加劝退。

【问】

嗯嗯明白，那从结构上来说呢？

【答】

结构上来说我还是觉得由简入深，然后由易入难那种结构。首先就入门的话肯定介绍概念，介绍一些基本的元素，介绍一些数据结构，简单的使用，我觉得一个完整的，10分钟之内能用起来的一个例子，一个demo，其实我觉得是必须的，因为大家可能都很难阅读完，从头到尾阅读完整个文档，大家都想快速使用，赶紧上手，所以你得放一个getting started的一个教程。

【问】

嗯，这个教程就是要么是比较简单的文字版，那如果有视频的话，你希望有吗？

【答】

视频的话当然最好。

【问】

嗯，还有其他要补充的吗？交互设计上你觉得有什么期待？

【答】

如果有能力的话，能做到交互式设计也挺好，这里的交互式设计指的是什么？

【问】

就比如说那个你刚才提到的代码可以直接运行，就给你一个开发的环境，让你试着运行一些例子。

【答】

这个当然是最好的对，如果再设计一些，你不是直接在那个代码框给定代码，让我点运行，你设计一些题目让我自己填入代码，然后运行，如果通过的话就会得到一些反馈，那样会更好啊。

【问】

你觉得有例子或者有题目的文档学习效果会更好，是吗？

【答】

对，让我自己可以思考。

【问】

嗯嗯，还有要补充的吗？

【答】

文档如果能有那种开发者社群的联系方式可能也更好，因为除了加一些比如说邮箱地址，比如说邮件列表，比如说XX群啊QQ群啊，比如说各种通讯APP，或者有个单独的可以提问题的地方，对留言板可能也挺好的，让大家可以反馈自己的意见。还有就是持续维护，持续更新，版本要跟上，然后对于大家反馈意见，你要做出一些改变。

【问】

还有吗？

【答】

没有了。

【问】

那最后一个问题可能跟文档本身没有太多的关系，我们这个研究要面向的是中国的开发者，所以我们想了解一下中国的开发者对自己的一个认知吧，就是你觉得在中国啊，程序员、开发者这个群体跟其他专业其他职业的人相比有什么特点？会有哪些不同的地方？可以从工作，从学习，从生活的方式来看，思考的方式等等，都可以大致讲一下。

【答】

工作，我感觉其实工作中的程序员都挺热爱自己的工作，也挺热爱编程这个事业的，就是大家也不是拿来当一个养家糊口的一个工作，大部分人都是真心喜爱敲代码，真心喜爱编程，因为除了工作之外，感觉大家业余其实也很有开发精神，会捣鼓一些新奇的事物，也会开发自己的小工具，编程对他们来说可能不是一种只是为了赚钱的一个事情，而是一个你可以终身做的一个兴趣爱好。

【问】

如果从学习上来说呢，你们是不是需要经常学新的东西？

【答】

嗯，在工作中确实是这样，每一个新的项目过来的话，碰到的技术栈，碰到的问题都是全新的，你要在快速的时间内学习新的东西。

【问】

嗯，那学习的方式呢？是怎么样的？有什么特点？

【答】

学习的方式，技术文档就是最普遍的一个方式，大部分人都是遇到问题，其实都是自己想办法去解决，很少是比如说在XX上找朋友去到处问，大部分人都是在网上自己搜索，自己检索，自己解决这些问题。

【问】

是因为这样更高效是吗？还是说你们本身就习惯自学，自己靠自己？

【答】

因为计算机这个领域非常大，你遇到的问题可能别人从来没有碰到过，就是你找同行，他/她遇到和你一样的问题概率比较小，除非是很普遍的，你在网络上搜索可能那个范围更加大，遇到和你相同问题的人的概率就会大一点。

【问】

还有其他的吗？

【答】

没有了。

【问】

那我们今天的访谈就结束了，谢谢。