*摄像机从舞台左边摇摄。画面中显示着一个打开了空白页面的编辑器。一个人弯着腰坐在桌子前面，头朝着桌子。*

上面的场景对每个以写作为生的人来说都很熟悉，空白的页面混杂着许多情感。它充满了新开始的兴奋和新鲜感。然而也充满了不知从何起笔的绝望。

让我来结束这个场景的放映。

这是一篇帮助你给第一个项目写文档的指南。万事开头难，我希望这份指南能把你引导到正确的道路上。最后，你应该有一个可以公开发布的项目。

请轻松地阅读完这篇文章，或者简单地把它当作参考。

为什么要写文档？

你将会在 6 个月后使用你的代码

我发现一开始从利己的角度来解释这个问题会比较有吸引力。写文档最好的理由就是你将会在 6 个月后使用你的代码。你 6 个月前写的代码跟别人写的代码对你来说通常没有什么区别。你将会带着一种熟悉的感觉读你的代码。然后一种不良的预兆悄悄逼近，你发现写代码的人毫无经验，毫无智慧。

当你读完几个月前很简单易懂或者取巧的代码之后，你就会开始同情你的用户。只要我写下为什么我要这么做，生活就会变得如此简单。文档能让你记录代码为什么这样写的原因。同样地，代码注释解释了为什么，而不是怎么做，文档也是这样。

你想要别人使用你的代码

你已经写了一段代码，并且发布了它。你这样做是因为你认为别人可能觉得它有用。但是，人们需要先知道为什么你的代码对他们可能有用，才会决定使用它。文档可以告诉人们这个项目对他们有用。

* 如果人们不知道你项目存在的意义，他们不会使用它。
* 如果人们不知道怎么安装你的程序，他们不会使用它。
* 如果人们不知道怎么使用你的代码，他们不会使用它。

只有少数人会深入研究你的代码并且使用它。几乎没人会使用你的代码，除非它有好的文档。如果你真的热爱你的项目，给它写文档，让其他人使用它。

**你需要别人的帮助**

开源项目是具有魔力的，对吗？你发布了代码，然后 code gnomes 出现并且让你的代码更好。

当你发布代码的时候会有一种奇妙的感觉产生。它通过各种方式出现，但总是能让你兴奋不已。有人在使用我的代码？这是一种混合了恐惧和兴奋的感觉。

我创造了价值！

如果它出错了怎么办？！

我是一个开源项目开发者了！

天哪，别人在使用我的代码。。。

写好的文档能够减轻这种恐惧。很多恐惧来自于给世界创造东西。我最喜欢的关于这个问题的引文如下所示：

恐惧来自于你做着重要的事情。

如果你在做不让你恐惧的事情，那么它对你或者世界都没有益处。

恭喜你感到恐惧！它意味着你在做重要的事情。

实际上，不完全是这样。

开源项目确实令人感到惊奇，但它同样必须符合现实世界的规则。你必须有付出，才有收获。

在你为项目付出大量工作之后，才能获得贡献。

在你的项目有用户之后，才能获得贡献。

在你的项目有文档之后，才能获得贡献。

文档也为你项目的第一次贡献提供平台。很多人从来没有为开源项目贡献过，让他们修改代码比修改文档可怕得多。如果你没有文档，你将失去这类文档贡献者。

文档让你的代码更好

有一个古老的事实：把东西写下来帮助你更好地思考。头脑里产生一个听起来不错的想法很容易，但是把想法写到纸上就需要思想上的升华。

写文档能提升代码的设计。在纸上讨论你的API和设计决定可以让你用一种更正式的方式思考它们。它还有一个不错的副作用：让别人按照你原来的意图贡献代码。

你想成为一个更好的写作者。

写文档跟大多数人体验过的写作方式不同。技术写作是一种非与生俱来的艺术。写文档将会是你成为更好的技术写作者这条路的起点，作为程序员，这是一个有用的技能。

写作会随着时间的流逝变得更容易。如果你好几个月没写东西，再次动笔就会变得更加困难。边做项目边写文档将会让你保持一个合理写作节奏。

用什么工具写作

简单的开始是取得实际成果的最好方式。我将会提供一条平坦的路给你走，在你有了基本的想法之后，你可以扩大范围。这些工具很强大并且容易使用。这将移除写作的障碍。

这篇文章中的例子在 Markdown 和 reStructuredText 中都是有效的。reStructuredText 有一点难用，但是更强大。我推荐你两个都看看，然后决定哪个你想要用。

纯文本版本控制

做为程序员我们生活中纯文本的世界里。我们的文档工具不应例外。我们想要能把纯文本转化成漂亮的 HTML 的工具。我们还有一些最好的跟踪文件变化的工具。为什么我们写文档的时候会放弃使用这些工具？这套工作流是强大的，并且对开发者来说很熟悉的。

基本例子



|  |  |
| --- | --- |
| 1  2  3  4  5 | Resources  ---------    \* Online documentation: http://docs.writethedocs.org/  \* Conference: http://conf.writethedocs.org/ |

上面的文字将渲染成标题，下面带着列表。URLs 将会被自动超链接。这写起来很容易，不但作为纯文本有意义，而且还能很好的渲染成 HTML。

**README**

你第一个应该写的文档是 README。如果你提供了合适的扩展名，代码托管服务会自动把你的 README 渲染成 HTML。它也是大多数用户跟你的项目的第一次互动。所以，一个可靠的 README 对你的项目十分有用。

有些人甚至 [从 README 开始做项目](http://tom.preston-werner.com/2010/08/23/readme-driven-development.html)。

文档写什么

现在我们来讨论重要的事情。确保你的用户能得到他们需要的所有信息，但不要太多。

首先，你需要问自己：文档是为谁而写。一开始，你只需要吸引两类的读者：

* 用户
* 开发者

用户是指那些只是想使用你的代码，而不管关心它怎么工作的人。开发者是指那些想要给你的代码做贡献的人。

你的项目解决了哪些问题

很多人会通过你的文档搞清楚你的项目是什么。有人会提到它，有人会随机地用 Google 搜索一个短语。你应该解释你的项目做了什么和它存在的意义。[Fabric](http://docs.fabfile.org/) 这方面做的很好。

一个代码的小例子

提供一个生动的例子来告知用户你的项目通常会被用来做什么。 [Requests](http://docs.python-requests.org/en/latest/index.html) 是一个很好的例子。

一个代码和问题追踪的链接

人们有时候喜欢浏览源代码。他们可能对归档他们发现的代码 BUG 感兴趣。尽可能地让那些想要为项目做贡献的人做这件事很容易。我认为 [Python Guide](http://docs.python-guide.org/en/latest/index.html) 做得很好，因为它有指向代码部分的链接。

常见问题（FAQ）

很多人会有相同的问题。如果问题一直发生，你应该适当地修改你的文档或者代码来解决问题。但是总是有一些经常被问的关于项目的问题，不能改变的的事情等。把这些记录在文档中，并且使其保持最新。FAQs 通常会过时，但当它们被很好的维护时，它们就是黄金资源。 [Tastypie](http://django-tastypie.readthedocs.org/en/latest/cookbook.html) 做得很好，它把 FAQs 整理成“Cookbook”。

如何获取支持

邮件列表？IRC 频道？文档要记录如何获取帮助和跟项目社区交流。 [Django](https://docs.djangoproject.com/en/1.8/faq/help) 这方面做得很好。

给贡献者的信息

在项目中，人们通常有怎么做事情的标准。你应该记录在文档上，以便于别人写代码时能符合项目的标准。[Open Comparison](http://opencomparison.readthedocs.org/en/latest/contributing.html) 这方面做的很好。

安装说明

一旦人们决定使用你的代码，他们需要知道如何获取它和运行它。但愿你的安装指令只有几行用于基本使用。如果有必要，给出提供更多信息和说明的页面链接。我认为 [Read the Docs](http://read-the-docs.readthedocs.org/en/latest/install.html) 中我们做得很好。

你的项目许可证

BSD?MIT? GPL? 这些证书可能对你来说没什么，但是使用你代码的人会很关心这个。考虑一下你想要用什么证书发布你的项目，务必选择一个互联网上的标准许可证。

下一步做什么

在你遵循上面的指南之后，我们相信你的项目将会成功。进一步的参考资料，查看这篇文章 [如何维护一个开源项目](https://medium.com/p/aaa2a5437d3a)。

模板

给你一个 README 的模板。如果你使用 markdown 的语法，把它命名为 README.md。如果你使用 reStructuredText 的语法，把它命名为 README.rst。



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | $project  ========    $project will solve your problem of where to start with documentation,  by providing a basic explanation of how to do it easily.    Look how easy it is to use:        import project      # Get your stuff done      project.do\_stuff()    Features  --------    - Be awesome  - Make things faster    Installation  ------------    Install $project by running:        install project    Contribute  ----------    - Issue Tracker: github.com/$project/$project/issues  - Source Code: github.com/$project/$project    Support  -------    If you are having issues, please let us know.  We have a mailing list located at: project@google-groups.com    License  -------    The project is licensed under the BSD license. |