如何证明你真的读过这本书?

積ん読

日语里有个很好玩的词，“**積ん読**”（tsundoku）：

> 指那些买回来堆在那里还没读过的（甚至后来干脆不看了的）书……

细想想，每个人都有很多很多“積ん読”。小时候我们拿回家的教科书中就有相当一部分，其实就是“積ん読”，虽然那时候掏钱买书的是父母，不仔细看、或者干脆不看的时候，也知道自己在偷懒…… 再后来就是“主动犯罪”了 —— 比如，很多人买到手里的英语词汇书是根本就没有翻到过第二个列表的，乃至于过去我常常开玩笑说，中国学生都认识一个单词，abandon，不是吗？这个单词是很多很多人“决心重新做人”而后“就这样罢”的铁板钉钉的见证者。

在没有电子书的时代，印刷版书籍多少还有一点“装饰品”功用，可是到了电子书时代，谁知道你的设备里有多少付费书籍呢？攒下那么多，其实并没有炫耀的地方，给谁看呢？据说，Kindle 的后台数据里可以看到清楚的“打开率”，大抵上也是在 ¼ ~ ⅓ 之间，也就是说，差不多有 ⅔ ~ ¾ 的电子书籍被购买下载之后，从来就没有被打开过。

如此看来，付费之后并不阅读，只能欺骗一个对象了：自己。跟心理学家们之前想象的不同，我认为人们通常是不会欺骗自己的，至少很难“故意欺骗自己”。所以，对于“买了之后坚决不读”这个现象，我不认为“给自己虚妄的满足感”是最好的解释。

更朴素一点，更接近真相的解释是：

> 那百分之七八十的人，其实是想着给自己一个希望……

—— 等我有空了一定看。嗯。

说来好笑，其实每个人共同拥有的目标之一是这样的：

> 成为前百分之二十的少数人……

然而，PK 掉百分之七八十的人的方法真的很简单很简单啊：

> 把买来的书都真真切切地认真读过就可以了。

这实在是太简单了罢？！可是…… 我知道你刚刚那个没出息的闪念：

> 那我少买书甚至不买书不就可以了吗？

你自己都知道这是荒谬的，却忍不住为你的小聪明得意 —— 其实吧，幸亏有你们在，否则我们怎么混进前百分之二十呢？

PoW

比特币这个地球上第一个真正被证明为可行的区块链应用中有一个特别重要的概念，叫做“**工作证明**”（Proof of Work）—— 你干活了就是干活了，你没干活就是没干活，你的工作是可被证明的……

借用这个思路，我设计了个方法，让你有办法证明自己就是看过这本书，就是读完了这本书 —— 你能向自己也向别人证明自己曾经的工作…… 是不是挺好？

证明的方法是使用 github.com 这个网站以及版本控制工具 **git**。

具体步骤

请按照以下步骤操作：

> 1. 注册 github.com 帐号 —— 无论如何你都必须有 github 账户；
> 2. 使用浏览器访问 https://github.com/selfteaching/the-craft-of-selfteaching；
> 3. 在页面右上部找到“Fork”按钮，将该仓库 Fork 到你自己的账户中；
> 4. 使用 `git clone` 命令或者使用 Desktop for Github 将 the craft of selfteaching 这个你 Fork 过来的仓库克隆到本地；
> 5. 按照 Jupyterlab 的安装与配置 的说明在本地搭建好 Jupyterlab —— 如果在 Jupyterlab 中浏览本书的话，其中的所有代码都是可以“当场执行”的，并且，你还可以直接改着玩……

> 6. 在阅读过程中，可以不断通过修改文章中的代码作为练习 —— 这样做的结果就是已阅读过的文件会发生变化…… 每读完一章，甚至时时刻刻，你都可以通过 `git commit` 命令向你自己 Fork 过来的仓库提交变化 —— 这就是你的阅读工作证明；
> 7. 仓库里有一个目录，`my-notes`，你可以把你在学习过程中写的笔记放在那里；
> 8. 仓库里还有另外一个目录，`from-readers`；那是用来收集读者反馈的 —— 将来你可以写一篇《我的自学之路》，放在这个目录里，单独创建一个分支，而后提交 `pull request`，接受其他读者投票，若是达到一定的赞同率，那么你的文章就会被收录到主仓库中被更多人看到，激励更多的人像你一样走上自学之路……

当然，为了这么做，你还要多学一样反正你早晚都必须学会的东西，Git —— 请参阅附录《Git 入门》。

时间就是这样，我们没办法糊弄它。而有了 git 这样的工具之后，我们在什么时候做了什么样的工作，是很容易证明的 —— 这对我们来说真是天大的好事。

如何使用 Pull Request 为这本书校对

另外，在你阅读的过程中，发现有错别字啊、代码错误啊，甚至有“更好的表述”等等，都可以通过 `pull request` 来帮我改进 —— 这也是一种“工作证明”。

(1) 使用浏览器访问 https://github.com/selfteaching/the-craft-of-selfteaching

(2) 点击右上角的“Fork 按钮”，将该仓库 Fork 到你的 Github 账户中

(3) 创建一个新分支，可以取名为 `from-<your_username>`，比如，`by-git.basic.tutorial`；之后点击 Create Branch 建立新分支。

(4) 在新分支下进行修改某个文件，而后提交 —— 提交前不要嫌麻烦，一定要在 Comment 中写清楚修改说明：

以上示例图片中是修改了 README.md 文件 —— 事实上，你应该提交的是的确有必要的校对。

另外，**请注意**：在创建分支之前，要将你的 Fork 更新到最新版。具体操作方法见下一节《如何在 Github 网站上将自己的 Fork 与原仓库同步》。

(5) 在页面顶部选择 Pull request 标签：

而后点击 `Compare & pull request` 按钮 —— 如果看不到这个按钮，那就点击下面刚刚修改文件的链接，如上图中的“Update README.md”（这是你刚刚提交修改时所填写的标题）。

确认无误之后，点击 `Create pull request` 按钮。

(6) 随后，Github 用户 @xiaolai —— 就是我，即，the-craft-of-selfteaching 这个仓库的所有者，会被通知有人提交了 Pull request，我会看到：

在我确认这个 Pull request 修改是正确的、可接受的之后，我就会按 `Merge pull request` 按钮 —— 如此这般，一个修正就由你我共同完成了。

注意

提交 Pull request 的时候，最佳策略如下：

> - 提交 Pull request 之前，必须先将你的 Fork 的 master 与原仓库同步到最新；
> - 从 master 创建 **新的 branch** 进行增补、修改等操作；
> - 尽量每次只提交一个小修改；
> - 提交时尽量简短且清楚地说明修改原因；
> - 耐心等待回复。

当自己的 Fork 过来的仓库已经被你在本地“玩残”了的时候，它千万不能被当作用来提交 Pull request 的版本。自己本地怎么玩都无所谓，但需要向别人提交 Pull request 的时候，必须重新弄一个当前最新版本到本地，而后再在其基础上修改。

如何在 Github 网站上将自己的 Fork 与原仓库同步

(1) 在你的 Fork 页面中如下图所示，点击 `Compare` 链接：

(2) 将 `base repository` 更改成当前自己的 Fork，在图示中即为 `gitbasictutorial/the-craft-of-selfteaching`：

(3) 这时候，页面会显示 `There isn't anything to compare.`，因为你在比较“自己”和“自己”。点击 `compare across forks` 链接：

(4) 将 `head repository` 更改成 Upstream Repository（即，上游仓库），在图示中即为 `selfteaching/the-craft-of-selfteaching`：

(5) 稍等片刻，你会看到比较结果；而后你可以创建一个 Pull request —— 这是一个由你自己向你自己的 Fork 仓库提交的 Pull request：

(6) 而后你在 `Pull requests` 标签页里会看到你刚刚提交的 Pull request：

(7) 同意并合并之后的结果是，你的 Fork 与上游仓库同步完成了：

当然，有时会出现一些你无法解决的问题，那么，还有一个最后的方法：

> 将你的 Fork 删除，而后重新到 https://github.com/selfteaching/the-craft-of-selfteaching 页面按一次 `Fork` 按钮……

如何使用 github 记录自己的学习过程

你可以在本地建立一个分支（branch），例如，取名为 `study`：

```
git branch study
git checkout study
```

如此这般之后，你在本地工作目录中所做的任何修改，都可以提交到 `study` 这个分支之中。

你每次在 Jupyterlab 中浏览 `ipynb` 文件，按 `^ + Enter` 执行 code cell 中的代码的时候，该文件都会发生一些变化；你也可以随意修改文件中的任何地方，比如，添加一个 code cell，将某段代码从头至尾“敲”一遍；也可以修改某个 code cell 中的代码看看执行结果有什么不同；还可以添加或者修改任何 markdown cell —— 就当自己做笔记了……

总而言之，当你阅读完某一章节并如上所说那样做了一些改动之后，那个 `ipynb` 文件就发生了一些变化。于是，你就可以执行以下命令：

```
git add .
git commit -am 'my study result'
git push
```

如此这般，在 `study` 这个分支中就记录着你的学习轨迹。

当然，如果在这过程中，你发现本书自身有需要校对的地方，那么，你需要切换到 `master` 分支，执行以下命令：

```
git checkout master
git pull
```

而后再修改，进而按照上一节的方法提交 Pull request。

未来，在 https://github.com/selfteaching 下我会专门设置一个 repo，用来自动扫描 github 上本书的学习记录 —— 这种记录在过往的书籍当中是不可能存在的，然而，现在却可以了。在我看来，将来这种记录的作用甚至有可能比“学历”还要重要。