在本文中，简单总结一下对为什么要看源码、如何看源码这两个问题的思考。

# 看源码的意义

看源码只是一种方法、手段，而不是目的。我也曾经给自己制定过“阅读xxx源码”的目标，现在看起来真的很蠢，一点不smart（specific、measurable、attainable、relevant、time-bound）。

只有搞清楚了阅读代码的目标，才能有的放矢，抓住重点，高效达成任务。

看源码的意义总结起来包含但不限于以下几点：

## 第一：解决问题(BUG)

只要是代码，就会有bug，只是说bug的多与少、深与浅罢了。

现在大家都喜欢发布、使用开源项目，不同的开源项目社区成熟度、代码质量又会有较大的差异，遇到bug就不足为奇了。

当然，遇到bug肯定是先在网上搜索是否有类似的问题，一般可以在google、Stack Overflow、项目的issues里面有对应的关键词搜索。如果搜不到，那么就只能看源码解决了。

## 第二：知其所以然

我在*如何学习新技术、团队技术选型时要注意些什么*里面提到过，如果我们需要将一个开源项目用到自己的项目中，那么就必须了解这项项目的优缺点，并深知原理，对部分细节（尤其是项目的优势、feature）进行深入研究。

如果是成熟的开源项目，遇到问题也许能google到很多答案；但如果是一个处于快速发展中的开源项目，多了解其架构、核心原理，也能帮助快速定位问题。

另外，有的项目文档可能不那么丰富，但又不得不使用，那么如何以正确的姿势使用呢？也得参考源码。

## 第三：学习

看源码也是一种不错的学习方式（虽然不一定不是最佳的方式），尤其对于比较优秀的开源项目，能让人大开眼界。

即使是出于学习的目的，也是有很多侧重的，比如

1.学习语言：代码风格、规范、惯用法、高级语法。对于某个语言的新手，找一个熟悉领域的开源项目来深入掌握这门语言，也是一个不错的注意。

2.学习设计：数据接口、框架、整体架构

3.学习理论：算法、协议。比如我之前写过的raft协议，光看论文是很枯燥的，而且算法理论到工程实践还是有一定的差距，这个时候结合开源项目(mongodb)实现往往更事半功倍。

4.其他

## 第四：改造

一般来说，我们刚开始仅仅是使用一个开源项目，但随着使用的深入，会发现一些自己需要的功能并没有很好的支持，向项目组提的issues也可能得不到快速的响应。

这个时候就要自己开分支，改代码，加功能了。当然，比较好的是将自己分支比较好的新feature 给原项目提merge request，反哺开源项目，比如阿里的Blink。

## 第五：借鉴

他山之石可以攻玉，如果有需要重新开始自己造轮子，那么参考一些已有的、优秀的轮子肯定是有好处的。

**副产品：找工作**

这一点，不应该作为我们阅读源码的出发点，但是确实能在实际中对找工作、面试有加成，算是副产品吧。

# 如何看源码

看源码的目的很大程度上影响了看源码的方式、需要阅读的代码的范围。

比如说，如果是为了修一个线上bug，那么阅读代码的范围就紧紧围绕bug本身；而如果是为了了解某个分布式算法，那就需要按大量的、可能运行在不同节点（进程）上的代码，了解其交互原理、工作流程。

下面说一些通用的方法。

## 先看文档，整体把握

一般来说，文档是对代码的高度凝练，一个高质量的开源一般会包含tutorial、specification、API reference等documents，通过选择性的略读、精读这些文档，就能大致了解项目的整体架构、设计原则。

正确的路线是通过文档去认识这个项目，然乎通过阅读代码去验证文档、深入细节，而不是通过直接啃源码来了解这个项目，以偏概全。

## 理解代码组织，文件名，类名

当需要看代码的时候，不要找到一个文件就开始，先看看代码组织，粗略看看文件名、类名，基本就能猜测到每一部分。

比如redis的源码就组织得很好，基本上看文件名就可以快速定位每一个command的实现位置。

## 关注一个问题，从问题追踪代码

看源码的目标决定了此时此刻的关注点，不管是解决遇到的bug还是学习某个算法，都让我们聚焦到一个具体的问题，从这个具体的问题去追踪代码，忽略掉当前无需关注的细枝末节，步步深入，直达目标。

当然在解决一个问题的时候，有可能会引发新的问题，尤其是学习的时候，此时只需记录新问题（放到收集篮，不要立即发散），待之前追踪的问题解决之后，再来看新发现的问题。

## 解决一个issue

如果自己没有问题，那么就帮忙解决别人的问题，通常来说，开源项目都有许多待解决的issue，从中选择一个入手即可。

## 调试

只要可以，一定先让代码编译通过、跑起来，这样不管是加log、打印调用栈还是断点调试都方便很多。

尤其是对于像python这种动态类型代码，不跑起来很难知道到底在干啥。

## 加注释，做笔记

如果某份源代码的阅读并不是一锤子买卖，日后还可能回顾、重新阅读，那么就一定要做好代码注释和笔记。

笔记主要是框架图、类图、流程图，目标是建立索引，方便日后快速回忆。而注释就是阅读代码时的细节，重新阅读的时候看注释（特别是函数的注释）能节省很多时间。

以上，仅仅是一些个人总结，还请多多指教！

<https://www.cnblogs.com/xybaby/p/10794700.html>