代码评审review

代码审查清单

常规项

- 代码能够工作么？它有没有实现预期的功能，逻辑是否正确等。
- 所有的代码是否简单易懂？
- 代码符合你所遵循的编程规范么？这通常包括大括号的位置，变量名和函数名，行的长度，缩进，格式和注释。
- 是否存在多余的或是重复的代码？
- 代码是否尽可能的模块化了？
- 是否有可以被替换的全局变量？
- 是否有被注释掉的代码？
- 循环是否设置了长度和正确的终止条件？
- 是否有可以被库函数替代的代码？
- 是否有可以删除的日志或调试代码？

安全

- 所有的数据输入是否都进行了检查（检测正确的类型，长度，格式和范围）并且进行了编码？
- 在哪里使用了第三方工具，返回的错误是否被捕获？
- 输出的值是否进行了检查并且编码？
- 无效的参数值是否能够处理？

文档

- 是否有注释，并且描述了代码的意图？
- 所有的函数都有注释吗？
- 对非常规行为和边界情况处理是否有描述？
- 第三方库的使用和函数是否有文档？
- 数据结构和计量单位是否进行了解释？
- 是否有未完成的代码？如果是的话，是不是应该移除，或者用合适的标记进行标记比如‘TODO’？

测试

- 代码是否可以测试？比如，不要添加太多的或是隐藏的依赖关系，不能够初始化对象，测试框架可以使用方法等。
- 是否存在测试，它们是否可以被理解？比如，至少达到你满意的代码覆盖(code coverage)。
- 单元测试是否真正的测试了代码是否可以完成预期的功能？
- 是否检查了数组的‘越界‘错误？
- 是否有可以被已经存在的API所替代的测试代码？

你同样需要把特定语言中有可能引起错误的问题添加到清单中。

这个清单故意没有详尽的列出所有可能会发生的错误。你不希望你的清单是这样的，太长了以至于从来没人会去用它。仅仅包含常见的问题会比较好。

优化你的清单

把使用清单作为你的起点，针对特定的使用案例，你需要对其进行优化。一个比较棒的方式就是让你的团队记录下那些在代码审查过程中临时发现的问题，有了这些数据，你就能够确定你的团队常犯的错误，然后你就可以量身定制一个审查清单。确保你删除了那些没有出现过的错误。（你也可以保留那些出现概率很小，但是非常关键的项目，比如安全相关的问题）。

得到认可并且保持更新

基本规则是，清单上的任何条目都必须明确，而且，如果可能的话，对于一些条目你可以对其进行二元判定。这样可以防止判断的不一致。和你的团队分享这份清单并且让他们认同你清单的内容是个好主意。同样的，要定期检查你的清单，以确保各条目仍然是有意义的。

有了一个好的清单，可以提高你在代码审查过程中发现的缺陷个数。这可以帮助你提高代码标准，避免质量参差不齐的代码审查。