我是一个优秀的工程师么？

Jaybai 2017.7.5

研发职场工作~~套路~~思考

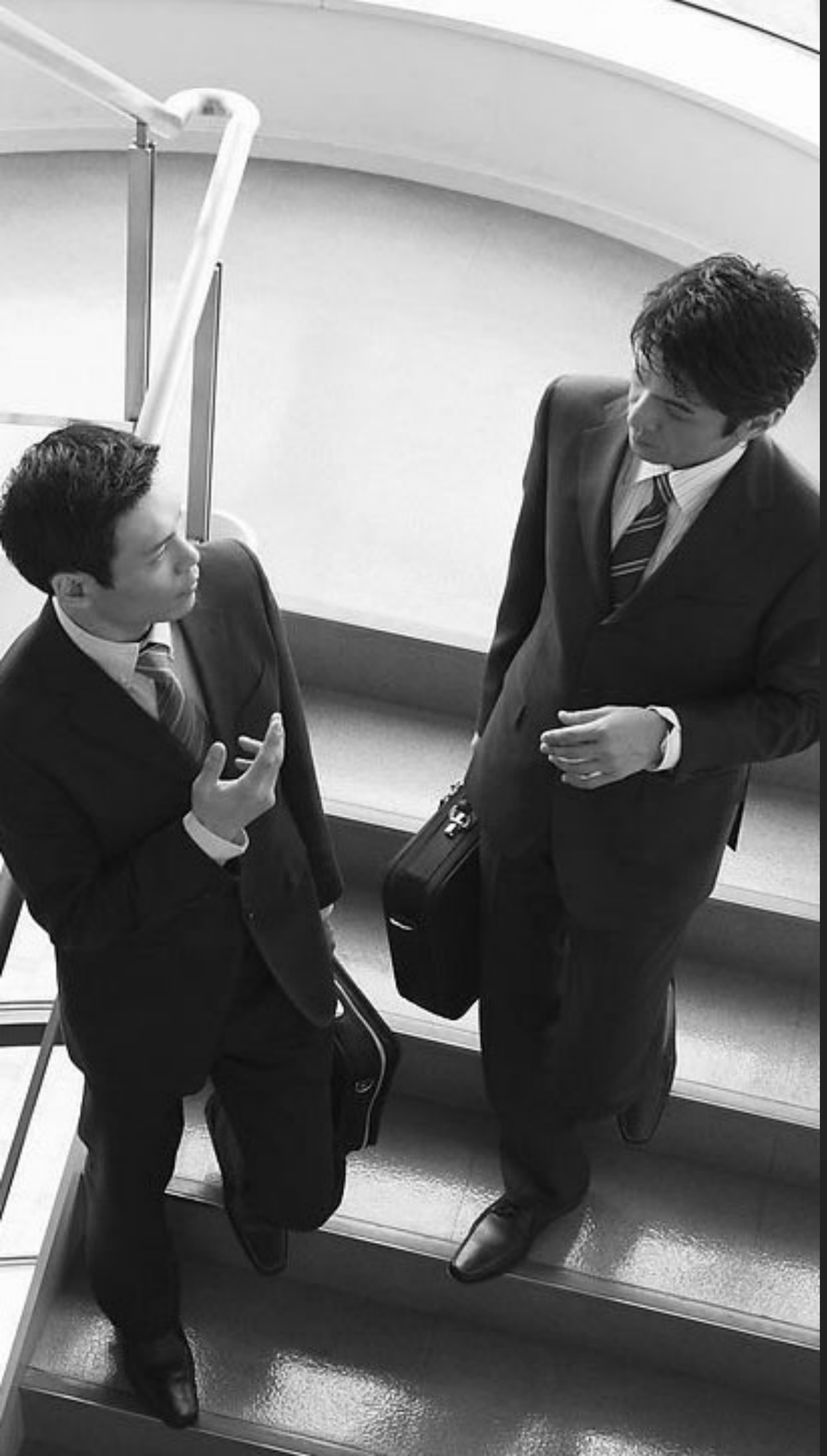
我的标签

- 灵活的胖子
- 自认为脑子好使的胖子
- 理性和感性的矛盾综合胖子
- 乐观的悲观主义胖子
- 好为人师的胖子

分享啥?

关键词：研发、职场、工作、思考

- 沟通
- 编码
- 处理问题
- 效率
- 学习
- 思考



沟通

没有什么是一顿烧烤解决的不了
如果有那就两顿

关键词：目标、态度、同理心、有效

- 确认沟通的目标
- 伸手不打笑脸人
- 兑现你的承诺
- 大家都在想什么
- 沟通有效么
- 你可能还需要一些小技巧

同理心

- 需求方在想什么?

  需求沟通、需求评审、需求变更

- 领导在想什么?

  工作内容、如何反馈

- 老板在想什么?

  公司方向、工作方向

- 同事在想什么?

  我该告诉我的上游我要什么？我该给我的下游知道什么?

需求评审

- 评审前的沟通

  避免评审会变成了需求沟通交流会

- 抓住主线

  评审之初要确认评审的主题，到底要达到什么效果。过程中的发散的问题，需要主持人及时收住，解决不了的记录下

- 客观评估

  站在自己的专业的角度给出建议和利弊但不要试图强硬说服，不要有情绪

- 留下后路

  时间上的缓冲期、需求的可能变更点、技术方案的扩展性

- 总结结论

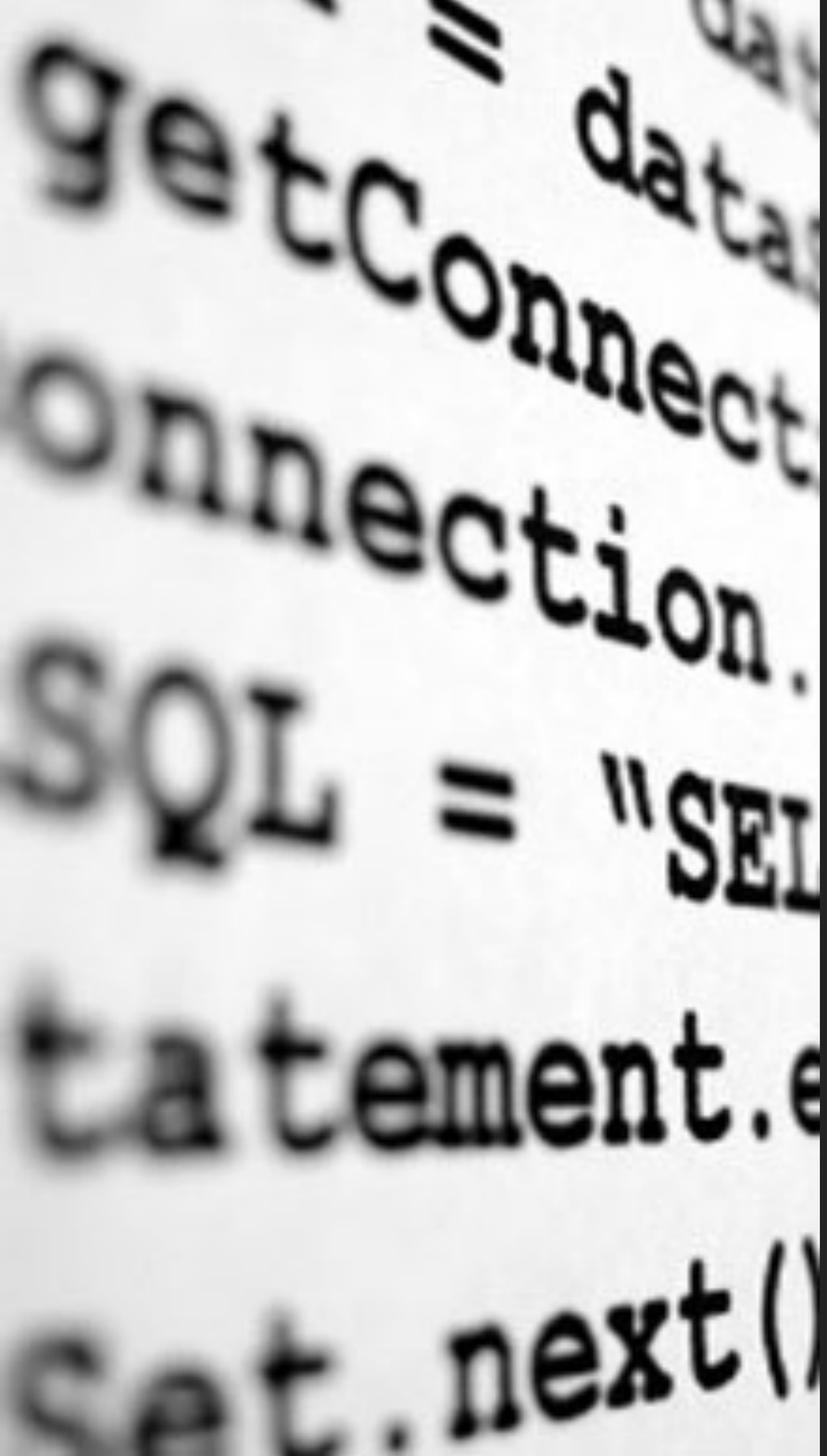
  一切没有结论的需求评审都是耍流氓

有效沟通

- 如何提问
  - 自己想过
  - 封闭式问题
  - 用对方听得懂的语言表达
- 确认效果
  - 说清楚了
  - 听明白了
  - 有结果了

小技巧

- 搞好私人关系
- 人都是需要赞美的
- 吃亏就是占便宜
- 默契需要时间来培养
- 双向确认
- 当面 > 电话 > 及时文字 > 邮件



编码

不会做设计的测试不是好码农

代码设计

- 复用，你真的懂么？

  复用不是简单的把代码块挪出来公用，而是对接口编程、分离可变和不可变、减少方法长度、减少方法数、减少对细节的理解

- 面向对象 vs 面向过程

  功能划分还是步骤划分

- 设计原则

  单一职责、里氏替换、依赖倒置、接口隔离、迪米特法则、开闭原则

- 代码的坏味道

  重复（代码、功能）、过长（方法、代码、参数）、过大（类、文件）、过度（设计、信息链、依赖）、发散（变化、修改）、冗余（代码、类、方法）、临时（表、类、方法、变量）、魔法（数字、字符串...）

编码小技巧

- 防御式编程

  非法输入、断言、错误处理、隔离、默认值...

- 契约式编程

  面向接口、面向契约的强约束

- 进攻式编程

  强行在断言或者默认处理的情况下中止程序，将问题暴露在开发阶段

- 算法和数学方法

  降低时间复杂度 e.g. 1 + 2 + 3 + … 98 + 99 + 100 = 50 * 101 for循环 –> 常数计算 O(n) –> O(1)，快排 vs 冒泡

- 高效的位运算

  在允许的情况下位运算效率明显高过数学计算（乘除2的倍数、奇偶校验、取最大最小值、判断符号相等、求平均值、取特定位...）

编码小技巧

- 小心使用null

  进攻式编程的时候需要它，真实场景需慎用

- 该死的if else多层嵌套

  有限状态机、do while、switch case、工厂、策略...

- 伪代码最佳实践

  整理思路、方便沟通、注释式伪代码

- 小心那个循环

  耗时操作要小心、不要在循环里计算初始值、循环退出条件

- 自测

  测试驱动开发、边界值、关键流程、结对

编码小技巧

- 理解语言的特性

  善用各种语言自己的特性 e.g. java中的字符串拼接用StringBuilder(高效)/StringBuffer(线程安全)代替String拼接，HashMap小技巧

- 减少浮点运算

  比如金钱精确到分的可以全站统一 * 100

- 内存泄漏&内存溢出

  用完释放/注销、置空、弱引用、预处理、小心长期对象

- 静态变量要小心

  一般语言的静态变量都是不会被回收的，静态变量引用对象不会被释放

- 避免死锁

  减少事务代码、加锁顺序、加锁时限、死锁检查

编码小技巧

理解数据结构的优劣

数组（知道坐标读写快、遍历慢大小固定）

链表（插入删除快，查找慢）

哈希表（知道key存取都快，删除慢遍历慢）

堆（大数据量的存取快，小数量的存取慢）

栈（后进先出、存取非栈顶慢）

队列（先进先出，中间的值存取慢）

二叉树 / 红黑树（查找插入删除快、删除算法复杂）

索引该如何用

读多写少的表、索引列值分散、数字索引比字符串索引快、索引列NOT NULL、索引失效（or、!=、like%、类型转换、函数计算、order by索引列）

问题处理

找到了问题就意味着解决了一半

搞清楚问题 — 重现、经验、沟通、更多信息

↓

确定范围 — 团队的边界、服务的边界、代码的边界

↓

找出问题 — 分析法、二分法、排除法、调试法

↓

解决问题 — 胆大心细、举一反三

↓

跟踪反馈 — 追“根”到底、线上跟踪

效率

找到自己的帕累托最优

效率

- ## 单线程 vs 多线程

  你的脑子只是一个单核CPU、专注的力量、什么时候可以多线程

- ## 大目标 vs 小目标

  把任务拆细直到你不太用动脑子思考为止、想得很复杂做的很简单

  6W2H: which、why、what、where、when、who、how to do、how much

- ## Checklist

  好记性不如烂笔头、同一个坑不能掉下去2次

- ## 工具是你的朋友

  推荐：思维导图、日历、印象笔记、markdown...

学习

“通”了才算学会
你那顶多叫知道

学习

知识体系是树形的

相互关联、触类旁通

如何开始

先用起来、培养好奇心、书籍是你的好伙伴

如何进阶

书籍还是你的好伙伴、实践！还是实践、忘了你的好伙伴（书籍）、三人行必有我师

融会贯通

理解的程度：听/看明白 -> 想明白 -> 干明白 -> 说明白-> 写明白

掌握的层次：不知道自己不知道 -> 知道自己不知道 -> 知道自己知道 -> 不知道自己知道


COMMUNICATION
MESSAGE
CONNECTION
BUSINESS
TECHNOLOGY
INTERNET
PEOPLE
COMPUTER
MOTHERBOARD
CONCEPT
CONNECT
MEDIA
DESIGN
CHAT
IDEA
BROADCAST
MESSAGING
TELEPHONE
PHONE
DISCUSSION
SYSTEM

思考

我所说的，都是错的

每个人都是独立的个体

- 我是谁？我从哪里来？我要到哪里去？

  清晰的自我认知和诉求

- 吾当三日自省吾身

  do{

  实践；

  自省；

  } while(true);

- 我所说的，都是错的

  只有适合自己的才是自己的

你问？我答！

Jaybai

谢谢

THANKS