

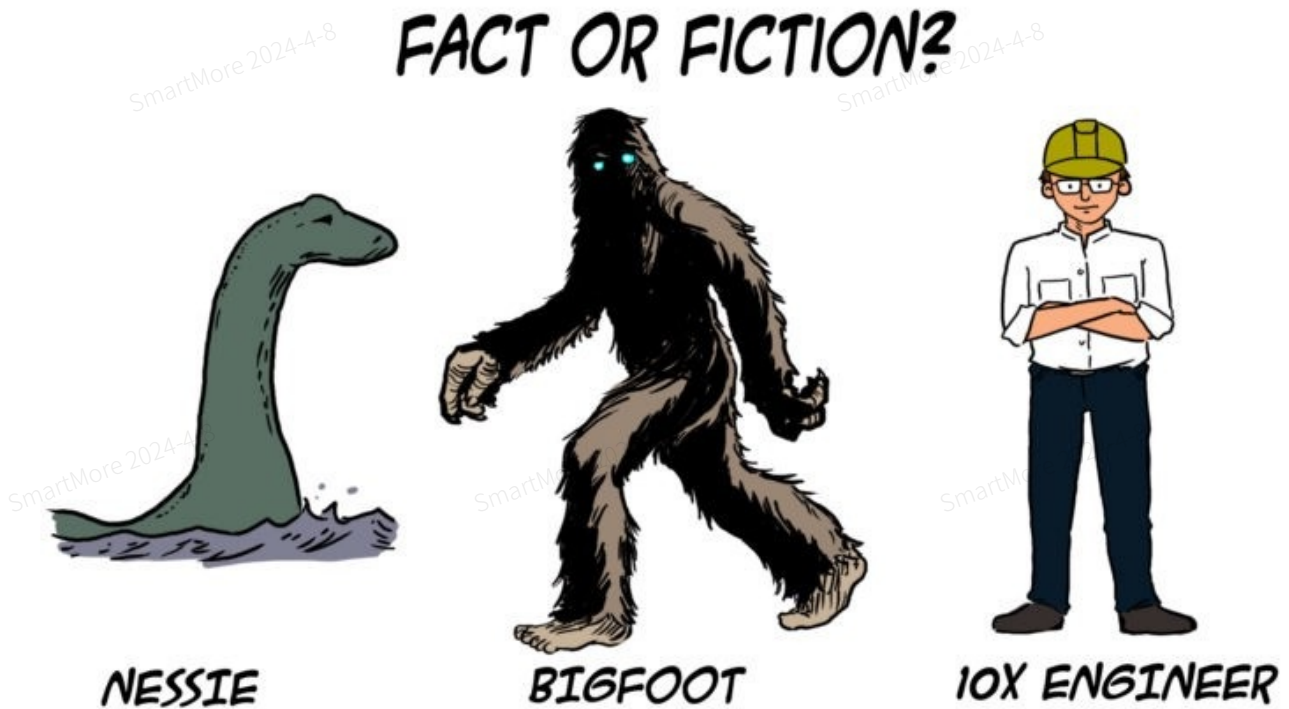
目录

目录	1
文章示例——优秀软件工程师的自我修养	2
初级工程师常犯的3个错误。你在犯这些错误吗？	2
1. 对工具的研究不深入	3
任务实例：建立一个网站	4
2. 不寻求帮助	5
3. 不能创造商业价值	6
最后的思考	7
富文本节点类型	9
标题	10
一级标题	10
二级标题	10
文本样式	11
列表	12
无序列表	12
有序列表	12
任务列表	12
代码块	13
引用	14
提示	15
数学公式	16
表格	17
图片	18
视频	19
文件	20
HTTP API	21

# 文章示例——优秀软件工程师的自我修养

原链: [How to Be a 10x Software Engineer](#) | by Michael Lin | Feb, 2022 | Medium

## 初级工程师常犯的3个错误。你在犯这些错误吗？



最优秀的工程师一夫当关万夫莫开。就像一支单兵部队一样，相比一个初级工程师团队，他们能够更高效更优质地提供更多价值。

但这怎么可能呢？不是越多越好吗？

在我担任Netflix和亚马逊的工程负责人期间，我曾与刚毕业的实习生一起工作，一直到首席工程师（亚马逊的L7及以上级别），我可以证明最优秀的工程师确实存在。我也可以自信地说，他们不仅仅是：

- 打字速度快 10 倍
- 工作 10 倍的时间
- 写 10 倍以上的代码

事实上，最优秀的工程师的打字速度可能只有一半，工作时间也只有一半，而且花在删除代码上的时间比写代码多。



200. Number of Islands

Medium 1937 77 Favorite Share

11110  
11010  
11000  
00000

grid[i][j] = '0';  
callBFS(grid, i+1, j); //up  
callBFS(grid, i-1, j); //down  
callBFS(grid, i, j-1); //left  
callBFS(grid, i, j+1); //right

10倍的工程师往往在LeetCode方面也更出色，但这并不是他们与众不同的地方。

最好的工程师和初级工程师之间的区别可以归结为心态的问题。他们使用正确的工具，问正确的问题，并知道如何确定优先次序。这些技能与编码关系不大，即使是非技术人员也能培养。

令人惊讶的是，将最好的工程师与普通工程师区分开来的是非技术性技能。

在这篇文章中，我讨论了初级工程师常犯的3个错误，以及高级工程师如何以不同的方式解决同样的问题--导致巨大的不同结果。

1. 对工具的研究不深入

亚伯拉罕-林肯曾经说过："如果我有8个小时来砍一棵树，我会花7个小时来磨我的斧头。

一个初级工程师可能会花8个小时用一把钝斧头砍树，而高级工程师会花1小时挑选合适的电锯，再花5分钟砍树。



诚实的亚伯会成为一个伟大的工程师

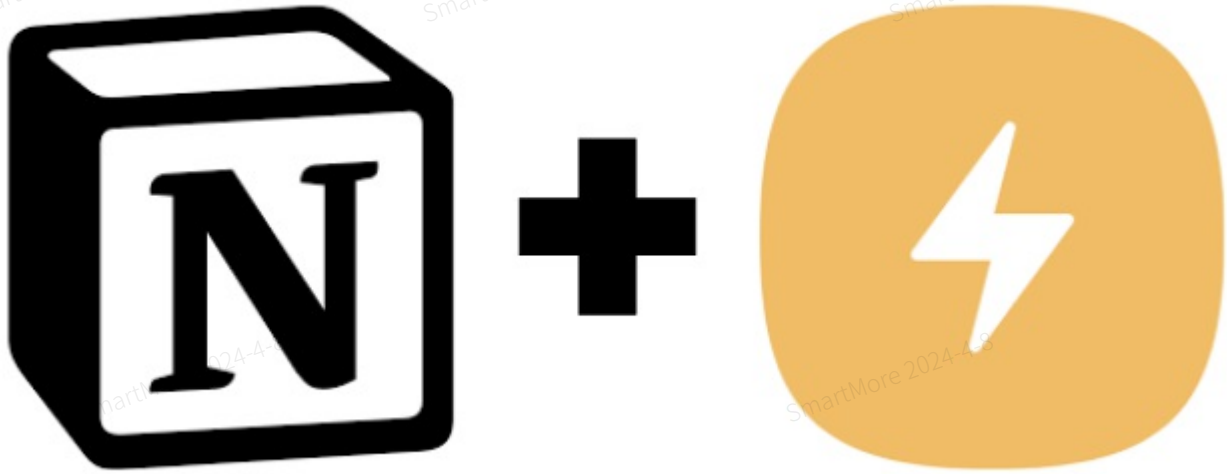
我看到初级工程师常犯的一个错误是，他们只顾埋头写代码，坚持使用他们知道的工具，并试图将其适用于各种情况。

如果一个普通的工程师只知道如何使用锤子，他们也会用锤子来挖洞。

他们几乎没有花时间去研究其他的替代方法--或者是否有可能完成写0代码的工作！  
使用正确的工具，就是在几周的劳动和10分钟内完成任务之间的区别。这就是10倍的差异。

## 任务实例：建立一个网站

最近，我有幸与一位初级工程师打赌，谁能更快地建立一个个人网站。这位新毕业的工程师花了两周时间，写了1000多行代码。而他在两周后甚至都没有完成！我花了1天时间，写了0行代码，甚至连汗都没出。你可以在[这里看到我的主页](#)的结果。



Notion page + super.so for deployments = [michaellin.io](https://michaellin.io)

当我请这位初级工程师解释她是如何处理这个问题的时候，就清楚了为什么花了这么长时间才完成。

"好吧，我在学校学会了如何制作React应用程序，所以我只是用它从头开始创建了一个网站。但要把图片和CSS弄好是很难的。而且我也不知道如何部署它。所以我想我必须在AWS上写一个自定义的部署脚本，但被控制台弄得有点糊涂。微型实例和大型实例之间又有什么区别呢？"

请注意，这位工程师的方法错过了几个关键点。首先，她从未讨论过：

- 需求--没有提到SEO、评论或拥有预制模板是否重要。
- 替代工具--他们只知道React+AWS，并坚持使用它。

想象一下，要从头开始重建评论功能。或者确保SEO正常工作。这些功能中的每一个都是一个团队的工作，以正确实现。难怪她从来没有完成！

## 2. 不寻求帮助

这是一个非常简单的问题，很容易解决，但却造成了大量的时间浪费，我不得不提一下。一些初级工程师有这样的误解：高级工程师就像一个孤独的天才。如果他们一直盯着这个问题，最终会得到解决。



10倍的工程师不是 "孤独的天才"，他们也需要寻求帮助！

但这是一种相当天真的思维方式。很多时候，区别在于**他们缺少背景--他们自己不可能推导出的信息。**

因此，他们不是直接寻求帮助，而是在代码库中苦苦思索，一遍又一遍地看同几行代码，而向队友提出一个5分钟的问题就能立即解决这个问题！

一个知道如何求助的经验不足的工程师，总是能打败一个从不求助的天才工程师。

有时，为了继续下去，显然需要额外的背景。例如，我们常常不清楚：

- 为什么代码库的结构是这样的
- 需要从其他团队调用哪个API
- 部署是如何进行的

这些都是上下文情况的例子，在这些情况下，你最好寻求帮助，而不是进一步挖掘代码库的内容。不要害怕寻求帮助！

### 3. 不能创造商业价值

10x工程师首先是投资者。

他们明白他们的工作是一种投资--而他们的投资回报必须大大超过所花费的时间成本。他们了解机会成本：花时间去建立一个功能意味着不花时间建立另一个功能。

工程师必须权衡机会成本--"在所有你可以建立的功能中，这个功能是对你时间的最佳利用吗？"



他们明白，代码是达到目的的一种手段--商业目的。如果他们能在没有代码的情况下实现他们的目标，那就更好了。编写的工作更少，需要维护的代码也更少--这是一个双赢的局面。



10倍的工程师是投资者，就像沃伦-巴菲特一样。

我看到很多新的工程师忽略了这些商业目标。例如：

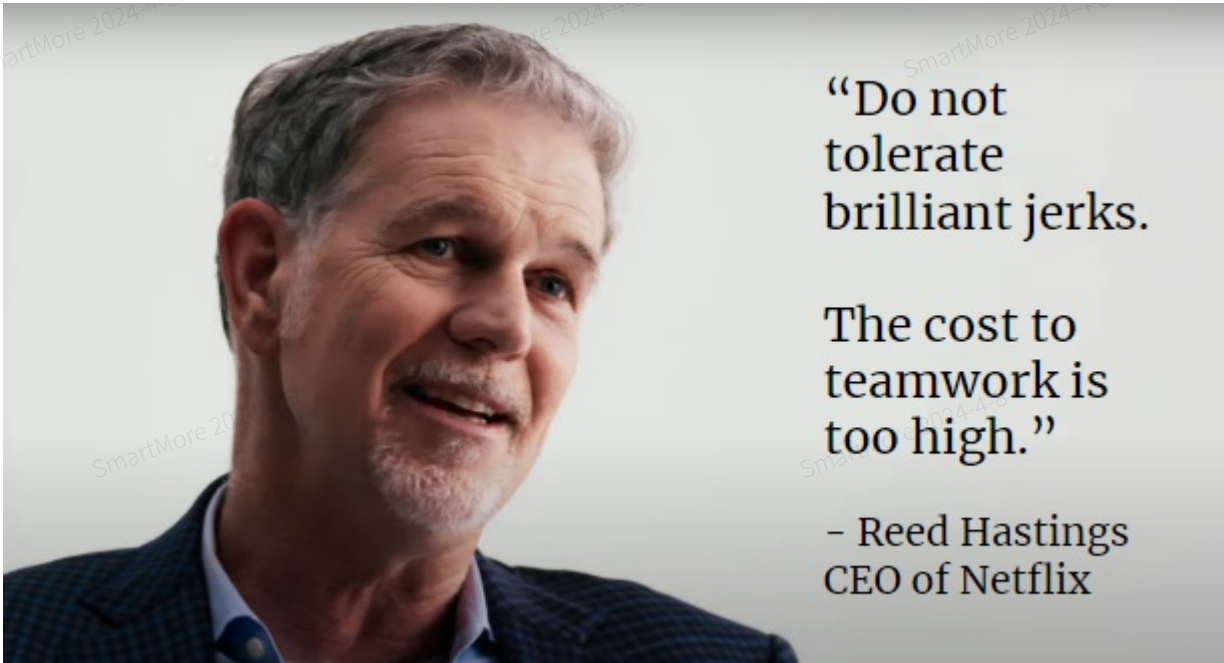
- "有一项新的技术，真的很酷。让我们花5天时间把它整合到网站上"（与产品不一致）。
- "唉，我不喜欢代码的结构方式。让我们花下一个冲刺期来重构"（机会成本--可以用这些时间来建立创造收入的功能）。
- "这个平台历史遗留问题太多了--让我们迁移到一个新的平台"（迁移是否能帮助你大大加快进度，还是只是一个增量改进？）

正是这种计算方法导致了10倍工程师的出现。如果一个初级工程师花了2个小时在一个不增加收入的复杂功能上，而一个高级工程师花了1个小时在一个简单的复制变化上，使收入增加了5倍，我们就得到了10倍的生产力改善。

1/2的时间花在一个能产生5倍收入的功能上=10倍的价值交付。

## 最后的思考

非技术性技能（"软技能"）是最强的工程师和最弱的工程师之间的区别。如果一个工程师避免了上述所有的错误，但他们却很难与之合作，那么他们的10倍技能就会失效。



摘录自Netflix文化牌

你努力工作，成为一名工程师。工程师比不做一个混蛋要难得多。不要因为你的自负而让你的努力工作付诸东流。并且永远记住：

工程师的首要任务是创造价值。



# 富文本节点类型

- 标题
- 文本样式
- 列表
  - 无序列表
  - 有序列表
  - 任务列表
- 代码块
- 引用
- 图片
- 表格
- 提示
- 视频
- 文件
- HTTP API
- 数学公式
- CPP API

#### 四级标题

[illegible]

# 文本样式

**加粗** *斜体* 下划线 ~~删除线~~ `code` click to the url。

左对齐左对齐左对齐

左对齐左对齐左对齐

中中对齐中对齐中对齐

中中对齐中对齐中对齐

右对齐右对齐右对齐

右对齐右对齐右对齐

# 列表

## 无序列表

- 列表项1
- 列表项2
- 列表项3

## 有序列表

- 列表项1
- 列表项2
- 列表项3

## 任务列表

- ☐ 列表项1
- ☒ 列表项2
- ☐ 列表项3

# 代码块

```
console.log("Welcome to Docaemon~");
```


```
#include <stdio.h>
int main()
{
    printf("Welcome to Docaemon!");
    return 0;
}
```





# 引用


React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

# 提示

 这是一条普通提示。

 这是一条警告！

 这是一条危险提示！！

 这是一条成功提示～

# 数学公式

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i \xi x} d\xi$$

表格

	服务一		服务二	
任务描述	召回率	精确率	召回率	精确率
1. 分类	100%	100%	100%	100%
1. 分类				
1. 检查				


图片



团建



# 视频

	视频	mp4 - 4.7MB
---	----	----------------

# 文件



文章示例——优秀软件工程师的自我修养.pdf

pdf -  
586.7KB

# HTTP API

GET

请求用户信息

https://xxx.com /api/user

请求用户信息

Request

Path Parameters

REQUIRED

Authorization

string

Bearer xxx

Response

200: OK

return user data

```
{
  "userId": "",
  "username": ""
}
```