《黑客与画家》读书笔记

推荐理由：

本书是硅谷创业之父 Paul Graham 的文集，主要介绍黑客即优秀程序员的爱好和动机，讨论黑客成长、黑客对世界的贡献以及编程语言和黑客工作方法等所有对计算机时代感兴趣的人的一些话题。书中的内容不但有助于了解计算机编程的本质、互联网行业的规则，还会帮助读者了解我们这个时代，迫使读者独立思考。

本书适合所有程序员和互联网创业者，也适合一切对计算机行业感兴趣的读者。

本书要点：

(1) 优秀的黑客养成了一种质疑一切的习惯。这是肯定的，因为如果你不得不同一台机器打交道，而这台机器全部由文字组成，像机械式手表一样复杂，并且规模大出 1000 倍，那么你也会养成这种习惯的。

(2) 我认为，真实世界的关键并非在于它是由成年人组成的，而在于它的庞大规模使得你做的每件事都能产生真正意义上的效果。

(3) 你不能盼望先有一个完美的规格设计，然后再动手编程，这样想是不现实的。如果你预先承认规格设计是不完美的，在编程的时候，就可以根据需要当场修改规格，最终会有一个更好的结果。

(4) 与历史上别的年代一样，我们的思想几乎肯定也是一张有错误的地图。如果你也犯下与别人一样的错误，那么这个错误不太可能完全来自于你自己。

(5) 只有深入了解当前的技术，黑客才能构想下一代技术。知识产权的拥有者也许会说，不，谢谢，我们不需要你的帮助，我们自己就能开发下一代技术。他们错了，在计算机工业的历史上，新技术往往是由外部人员开发的，而且所占的比例可能要高于内部人员。

(6) “你的电脑”这个概念正慢慢成为过去时，取而代之的是“你的数据”。你应该可以从任何电脑上获取你的数据。或者更准确地说，在任何终端设备上获取你的数据，终端设备不一定是电脑。

(7) 通过创造有价值的东西而致富，这种方法的优势不仅仅在于它是合法的，还在于它更简单。你只需要做出别人需要的东西就可以了。

(8) 技术无法使其变得更便宜的唯一东西，就是品牌。这正是为什么我们现在越来越多地听到品牌的原因。富人与穷人之间生活差异的鸿沟正在缩小，品牌是这种差距的遗留物。

(9) 我以前曾经认为，那些相信更严格的法律会遏制垃圾邮件的人真是太天真了。我现在认为，更严格的法律或许无法减少我们收到的垃圾邮件的数量，但是肯定有助于减少逃过过滤器拦截的垃圾邮件的数量。

(10) 把品味说成个人的偏好可以有效地杜绝争论，防止人们争执哪一种品味更好。但是问题是，这种说法是不正确的。只要你自己开始动手设计东西，就能明白这一点。

(11) 有意思的是，劫持飞机与“缓冲区溢出攻击”有类似之处。在一般飞机上，乘客区与驾驶舱是想通的，就好像 C 语言中数据区与代码区是相邻的一样。劫机者一旦进入驾驶舱，实际上就相当于把自己从数据提升为代码。

(12) 效率低下的软件并不等于很烂的软件。一种让程序员做无用功的语言才真正称得上很烂。浪费程序员的时间而不是浪费极其的时间才是真正的无效率。随着计算机速度越来越快，这会变得越来越明显。

(13) 除了某些特殊情况，你就是应该使用目前最强大的语言。不过在现实中这个结论很少能落实。到了一定年龄之后，程序员极少主动更换自己的编程语言。不管习惯使用的是哪一种语言，他们往往认为这种语言已经足够好了。

(14) 事实上，选择更强大的编程语言会减少所需要的开发人员数量。因为：(a)如果你使用的语言很强大，可能会减少一些编程的工作量，也就不需要那么多黑客了；(b)使用高级语言的黑客可能比别的程序员更聪明。

(15) 实体书并没有过时，它们读起来很方便，而且出版社对书籍内容的审核是一种很有用的质量保证机制。书店则是程序员发现和学习新语言的最重要的场所之一。

(16) 为了写出优秀软件，你必须同时具备两种互相冲突的信念。一方面，你要想初生牛犊一样，对自己的能力信心万丈；另一方面，你又要像历经沧桑的老人一样，对自己的能力抱着怀疑态度。

(17) 要记住一点，怎么理解编程语言？你不要把它看成那些已完成的程序的表达方式，而应该把它理解成促进程序从无到有的一种媒介。

精编书摘：

(1) 没错，成年人不知道孩子们内部发生的事。认识到这一点很重要。在抽象意义上，成年人知道孩子的行为有时是极端残酷的，这正如我们在抽象意义上知道贫穷国家的人民生活极端艰难。但是，像所有人一样，成年人不喜欢揪住不放这种令人不快的事实。你不去埋头探寻，就不会发现具体的证据，就会永远以为这件事是抽象的。

(2) 在读研究生期间，我潜意识里一直有一种很不舒服的感觉，觉得自己应该多学一点理论，不应该期末考试结束还不到三个星期，就把所有东西忘得一干二净，那样真是不可饶恕。现在，我意识到自己错了。黑客搞懂“计算理论”的必要性，与画家搞懂颜料化学成分的必要性差不多大。

(3) 我们可以自以为是地相信，当代人比古人更聪明、更高尚。但是，了解的历史越多，就越明白事实并非如此。古人与我们是一样的人，他们既不是更勇敢，也不是更野蛮，而是像我们一样通情达理的普通人。不管他们产生怎样的想法，都是正常人产生的想法。

(4) 不要被微软吓到。你能做到它做不到的事情，正如它能做到你做不到的事情一样。开发互联网软件不需要得到任何人的许可，没有任何能够阻止你。你不需要去申请许可证，不需要在零售店的货架上谋得一席之地，也不需要卑躬屈膝地求人家，将你的软件与操作系统捆绑在一起。你能够通过浏览器发布软件，没有人能在你和浏览网站的用户之间插上一脚。

(5) 小团队天生就适合解决技术难题。技术的发展是非常快的，今天很有价值的技术，几年后可能就会丧失价值。小团队在如今这个时代可谓如鱼得水，因为他们不受官僚主义和繁琐管理制度的拖累。而且，技术的突破往往来自非常规的方法，小团队就较少受到常规方法的约束。

(6) 在一个剥夺个人财产的社会，财富创造活动中所有那些没有乐趣的事情都会急剧地放慢，乃至停顿。

(7) 喜欢一件东西，却不知道为什么自己喜欢它，原因可能是这件东西是美的，但也可能因为他们的母亲也拥有同样的东西，或者杂志上某个明星使用它，或者仅仅因为它的价格很昂贵。人类的思想就是没有经过整理的无数杂念的混合。

(8) 关于面向对象编程优劣的争论并不像静态类型与动态类型之争那样壁垒分明，因为编程的时候你只能在静态类型和动态类型中选一种。但是，面向对象编程只是程度不同的问题。事实上有两种程度的面向对象编程：某些语言允许你以这种风格编程，另一些语言则强迫你一定要这样编程。

(9) 对速度的追求是人类内心深处根深蒂固的欲望。当你看着计算机这个小玩意，就会不由自主地希望程序运行得越快越好，真的要下一番功夫才能把这种欲望克制住。设计编程语言的时候，我们应该有意识地问自己，什么时候可以放弃一些性能，换来一点点便利性的提高。

(10) 编程语言的特点之一就是它会使得大多数使用它的人满足于现状，不想改用其他语言。人类天性变化的速度大大慢于计算机硬件变化的速度，所以编程语言的发展通常比 CPU 的发展落后一二十年。

(11) 认为所有语言都一样的看法的缺点是自欺欺人，但是优点是可以使许多事情变得很简单。我想这就是为什么它被广泛接受的主要原因。它是一个令人舒服的想法。

(12) 有一个笑话说，黑客动手写程序之前，至少会在心里盘算一下哪种语言的打字工作量最小，然后就选择使用该语言。这个笑话其实与真实情况相差无几。

(13) 做一个好的设计师就像做一个好医生一样。你不能头痛医头，脚痛医脚。病人告诉你症状，你必须找出他生病的真正原因，任何针对病因进行治疗。大多数优秀设计都是这样产生的，它们关注用户，并且以用户为中心。