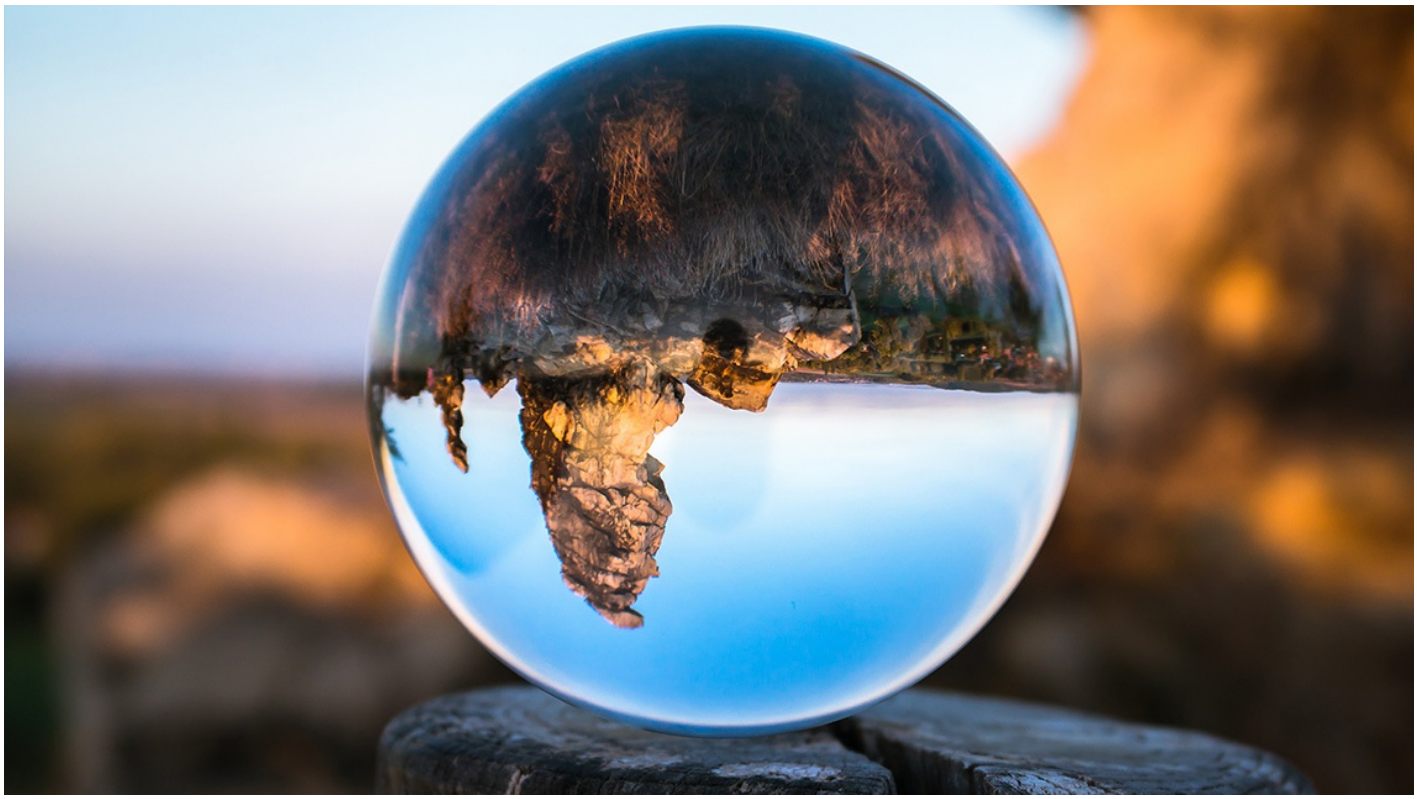
02讲以终为始：如何让你的努力不白费



今天内容的开始，我希望你可以先来思考一个问题：**如果让你设计一个登录功能，你会怎么做?**

我曾在公司内部做过这样一个练习，我扮演客户，让大家帮我设计一个登录功能。同事们一听就高兴了，登录不就是用户名加密码嘛，我熟啊，我还可以设计出验证码、找回密码、第三方登录等等功能。

更有个别动作快的同事，甚至已经开始设计数据库表，考虑用Redis做缓存了。整个过程下来，大家彼此讨论得热火朝天，唯一没人理会的就是我这个“客户”。

讨论结束，扮演客户的我告诉大家，作为一个“土豪”，我打算做一个打车软件，用户可以通过手机号接收验证码的方式进行登录。你可以想见，同事们一副“被套路了”的表情。是的，他们设计那套用户名密码登录完全是文不对题。

虽然这是一个简单的练习，但反映的却是我们日常面对的真实工作场景：许多人都是刚刚听到别人要求做的一个功能，就开始脑补接下来的一切。导致的结果，就是付出的努力毫无意义。

那么问题出在哪呢？因为我们欠缺了“以终为始”的思维习惯。

一种反直觉的思维方式

以终为始，就是在做事之前，先想想结果是什么样子的。

说起来很简单，但做到并不容易。因为我们习以为常的思维模式是线性而顺序的，第一步做完，做第二步；第二步做完，做第三步。

这也情有可原。我们人类都是从远古时代演化而来，在那个食不果腹的时代里，倒着思考的用途并不大，人们甚至不确定自己能否见到明天的太阳。几十万年的进化留给我们很多短视的行为和思考习惯，因为这样的做法最为节省能量，把目光放长远是需要额外消耗能量的。

“以终为始”是一种反直觉的思维方式，是大多数人不具备的。所以，日常生活中，我们看到很多有趣的现象。

比如，大学毕业时，有很多人想考研，如果你问他为什么要考研，得到的理由通常是为了找个好工作。但考研真的能帮他找个好工作吗？不一定，因为找工作和考研根本就不是同一棵技能树。

如果真的是想找个好工作，那你就应该了解工作的要求是什么，怎样才能掌握工作要求的技能。

从后面这个角度出发，你会发现考研只是通往工作诸多道路中的一条，其他的路径也是可以到达的。比如，你应该找个实习的地方锻炼一下职业技能。这就是“以终为始”思考问题的方式。

回到前面“设计登录功能”的例子，对比“以终为始”的思维，你也许会替我的同事抱不平，他们或许也有“以终为始”的思路，只不过，他们的“终”和我这个客户的“终”不一样罢了。这就要说到做软件，本质上是在构建一个“集体想象”。

想象的共同体

如果你读过尤瓦尔·赫拉利的《人类简史》或《未来简史》，有一个说法你一定不陌生：想象的共同体。作者认为，人类历史发展的一个重要因素是“集体想象”，无论是国家、宗教，还是法律、习俗，都是人们达成的“集体想象”。人类就是认同了这些“集体想象”的一个共同体。

我们这些做软件的人其实就是一个想象的共同体，这个“集体想象”就是我们要做的软件，任何想象都需要一个载体将其展现出来，我们编写软件的过程就是将这个“集体想象”落实的过程。

既然是“集体想象”，那么在载体将想象呈现出来之前，我们的想象很难统一起来，都或多或少存在差异。

所以，**任何事物都要经过两次创造：一次是在头脑中的创造，也就是智力上的或者第一次创造（Mental/First Creation），然后才是付诸实践，也就是实际的构建或第二次创造（Physical/Second Creation）。**

我们在工作中遇到的很多问题，其实就是在于第一次创造没有做好，就进入到第二次创造。所以，我们在工作中会遇到很多“惊喜”，准确地说，是惊吓。

相比于第一次创造，第二次创造是一件成本很高的事。我们知道，软件开发最费时费力，一旦投入大量精力做出来，却发现与理解偏差甚大，所有人都会欲哭无泪。

所以，在动手做事之前，我们要在第一次创造上多下一些功夫，将相关各方的“集体想象”统一起来。以建筑为例，就是先在图纸上构思各种细节。对应到做软件，我们也可以做很多事，比如：

- 要给用户看产品的样子，可以用原型工具把它做出来，而不是非得把完整功能开发出来；
- 要呈现服务接口的样子，可以用模拟服务器搭出一个服务，而不用等后端全部开发完毕；
- 要让程序员知道要开发产品的细节，可以在任务上描述出软件各种场景给出的各种行为。

再回到前面“设计一个登录功能”的例子上，我的同事们在构建的其实是他们自己的想象，而不是我们共同的想象。这其中最大的一个区别就在于，没有人会为他们自己的想象买单的。

所以说，他们看到的“终”不是真正的终，只是一个自我的“终”，至于看到什么样的“终”，这取决于每个人的见识。

对做软件的人来说，我们应该把“终”定位成做一个对用户有价值的软件，能够为别人带来价值，自己的价值才能体现出来。

至此，你对“以终为始”已经有了一个初步的认识，有了这种思维方式，我们可以在工作中怎样运用它呢？

规划和发现

软件行业有很多英雄传说，一个人或者一个团队连续奋战一段时间，写好了一个软件，在上线前夜发现了一个问题，然后冒着“不成功便成仁”的风险，通宵达旦解决了问题，一战成名。

这种故事听起来让人热血沸腾，但仔细想想，为什么总在最后一刻发现问题？除了时间压力确实大的情况以外，大多数情况，他们还是一开始没有想好就动手了。

在团队内部，我一直坚持“以终为始”，让大家在执行任务之前，先倒着想想再动手规划，这样规划出来的工作更能瞄准真正的目标。举一个之前做产品的例子，当年在创业的时候，我们打算做一个物联网开发平台，但具体应该做成什么样子呢？

有了“以终为始”的思维，我们考虑的是别人会怎么用我们的平台。我们设计的方式是，用户到我们的网站，阅读相关文档，然后参考文档一步一步照着做。

这其中的一个关键点是：文档，特别是《起步走》的文档，这是用户接触我们这个平台的第一步，决定了他对我们产品的第一印象。

所以，我们决定从写《起步走》这个文档开始，这个文档描绘了用户怎样一步一步使用我们的开发平台，完成第一个“Hello World”级别的应用。**请注意，这个时候，我们一行代码都没有写。**

写好了这个《起步走》文档，团队的所有人对于我们的平台要做成什么样子，已经有了一个比较初步的认识。更重要的是，我们可以拿着这个文档，去和外部的人讨论这个尚未出世的平台。

人类是一个擅长脑补的群体，一旦有人看到了这个文档，他就已经可以构想出这个平台已经存在的样子，进而给出各种各样的反馈：“我认为这个地方可以这样做”“我觉得那个地方可以改改”。

所有这些反馈都是真实的，因为他们已经“看到了”一个真实的东西。正是这些真实的反馈，让我们逐渐地锁定了目标。之后，我们才开始动手写代码。

“以终为始”的方式，不仅仅可以帮我们规划工作，还可以帮我们发现工作中的问题。

有一次，我的团队在开发一个大功能，要将现有的系统改造成支持多租户的系统。也就是说，别的商家可以到我们的平台上发起申请，拥有和我们现有平台一样的能力。

功能来了，各个团队将任务分解，然后就各忙各的去了。但我有着习惯性的不安，总担心丢点什么，于是催着项目经理梳理一下上线流程。

是的，上线流程，虽然我们的代码还没开发完，但是本着“以终为始”的态度，我们就假设各个部分已经开发好了，来想一想上线应该怎么做。

果不其然，一梳理上线流程，我们便发现了问题：怎么识别不同的租户呢？有人给出的方案是设置一个HTTP头。但谁来设置这个HTTP头呢？没人仔细想过。于是，一个潜在的问题就这样被发现了，至少不用在未来为它加班了。至于解决方案，作为程序员，我们有的是办法。

事实上，在今天的软件开发实践中，已经有很多采用了“以终为始”原则的实践。

比如测试驱动开发。测试是什么？就是你这段代码的“终”，只有通过测试了，我们才有资格说代码完成了。当然，测试驱动开发想做好，并不是先写测试这么简单的。

比如持续集成，我们是要交付一个可运行的软件，倒着来想，最好的做法就是让软件一直处于可运行的状态，那就是持续地做集成。

概括地说，**践行“以终为始”就是在做事之前，先考虑结果，根据结果来确定要做的事情。**

这是“以终为始”这个内容版块的开篇，后面我会给你介绍这个原则在不同场景下的应用，也会引入一些现在行业内的最佳实践进行解析。相信会对你的实际工作有帮助。

总结时刻

有一段时间，网上流传着一个帖子，亚马逊 CTO 介绍亚马逊是如何开发一项产品的，简单来说，他们采用向后工作的方法，开发一项产品的顺序为：

1. 写新闻稿；
2. 写FAQ（常见问题解答）；
3. 写用户文档；
4. 写代码。

今天我带你了解了“以终为始”的做事思路，回过头再来看这个帖子，相信你不难理解为什么亚马逊要这么做事情了。

人们习惯采用顺序思考的思维方式，几十万年的进化将这种思考模式刻在了我们的基因里。要成为更好的自己，我们要克服自身的不足，而这个做法很简单，那就是“以终为始”，做事倒着想，先考虑结果。

人类是一个想象的共同体，做软件的团队更是如此，而我们写出来的软件是我们将“集体想象”落地的载体。

任何事物都要经过两次创造：一次是在头脑中的创造，也就是智力上的或者第一次创造（Mental/First Creation），然后才是付诸实践，也就是实际的或第二次创造（Physical/Second Creation）。我们应该在第一次创造上多下功夫，统一集体想象，让目标更明确。

“以终为始”的思维可以帮助我们更好地规划我们手头任务，也可以帮助我们发现过程中的问题。

如果今天的内容你只能记住一件事，那请记住：**遇到事情，倒着想。**

最后，我想请你思考一下，在实际的工作或生活中，你有运用“以终为始”的思维方式吗？帮助你解决过哪些问题？欢迎在留言区写下你的想法。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

精选留言

每天晒白牙

我们在生活中经常遇到这样的场景:许多人听到别人要做的一个功能，然后自己开始脑补后面的东西。导致的结果，付出的努力没有意义。同样在面试的过程中也会发生类似的事，面试官谈到一个点，还没等面试官说完，就开始吧啦吧啦说一堆，然而并不是面试官想要问的。

这里面的问题是缺少”以终为始”思维习惯。

任何事物都要经过两次创造:一次在头脑中的创造也就是智力上的或者第一次创造，然后才是付诸实践即第二次创造。

第二次创造的成本往往相对较高，所以需要在动手之前，要在第一次创造上多下一些功夫。对于做产品来说，就是我们想给用户看原型或做各种问卷调查，充分理解用户喜欢的，只有符合用户的使用习惯，这样的产品做出来才有卖点。

拿开发举例子，我们一般会经过需求评审－技术方案评审－写代码－测试－上线－测试。

在需求评审阶段，pm给我们呈现的是最后的结果，这次需求想要达到的效果。

技术方案评审就是rd根据需求给出设计方案，怎么用代码来实现这个需求，最开始我接触出技术方案就是在58同城，之前的公司没有这个过程。我们在出技术方案的时候，要考虑好需求的实现过程可能出现的各种情况，这点往往是最考验rd的，我还需要努力呀，团队内其他小伙伴做的还是很棒的。然后叫上小组内的同学，一起对这个方案的可行性进行评估，或者给出优化方案，总之技术方案评审这个过程是个思维碰撞的过程，也是一个架构的过程，不是说非得架构师才可以玩架构，不知道这样说会不会被架构师揍。

写代码就是实践嗯过程即第二次创造的过程。这个过程需要在前面的基础上进行，如果需求理解有偏差，或技术方案设计的不好，在这个过程中往往是比较痛苦的。

测试阶段很重要，我们听说过测试驱动开发(TDD)，qa会根据这次需求想要的结果进行测试，这个过程很重要，就是再检验是否践行3了以终为始这个原则。其实我们的流程还有点欠缺，更好的流程是在开发前加上测试用例评审，之前待的一家公司有遇到过，把测试用例放到开发前面，会更好践行TDD，也更符合以终为始。

说了这么多废话，总之就是践行以终为始，就是在做事情前，先考虑后果，以结果为导向来确定要做的事情。

2018-12-28 07:51

Edward

开头的“套路”确实很生动。做技术的人会有这么样的倾向，听到要做一个功能，第一反应就不是去梳理功能的细节，而是把熟悉的技术（也有是自己业余时间研究的，想在项目里用上）拿出来，粗略地构想解决方案。也是那句话：“手里拿着锤子，看什么都像钉子。”

2018-12-29 11:31

jueyoq

当前处于什么水平?

优势：年轻(今年刚毕业)，没对象(充足学习时间)，善于钻研底层原理，对编程有热情，善于总结方法论，技术表达力强(会吹牛逼)

劣势：完美主义，英语差，算法一般，计算机原理不成体系(非科班)，稳定性差，数学功底一般，缺乏工作经验，工作效率低。

有什么目标?

想在技术上快速积累，达到主力开发程度。

如何实现?

吃透项目代码。

看开源项目源码。

写开源项目。

输出博客，打造职业品牌。

每天练习算法。

总结方法论，提高工作效率。

2018-12-28 21:42

pyhhou

感觉“以终为始”这样的思考模式其实在生活中经常出现，例如说要做菜给别人吃，得先考虑别人爱吃什么？南方人还是北方人？是否有饮食顾虑？自己会的菜当中哪道菜做出来更吸引人眼球？先是考虑好要做什么，再去买菜，然后做菜；还有就是买一个东西会先想买回去预期有一个怎样的效果，然后再决定买不买；

“能够为别人带来价值，自己的价值才能体现出来”，感觉这才是做软件的初衷，也是一个程序员的目标，受教了；

另外就是还请老师在后期的课程中指明程序员职业发展的最佳路径，或者授人以渔，道明如何去思考出属于自己的一个最佳路径，期待。

2018-12-29 01:50

David Mao

1.开发采用“以终为始”的原则有利于认清目标，提高开发效率。

2. 根据自己的经历，在产品的整个生命周期，业务的部分如销售在前期需求评审及开发阶段介入，会更好的支撑后续的销售工作。实际遇到的情形，产品开发完成后，由于产品没有市场竞争力，导致销售人员销售时遇到很大的压力，站在全局的角度，销售也应提早介入，销售也可以提产品的需求。目前提的持续交付业务（销售）也应是里面的一个环节。总结起来就是业务驱动开发，业务驱动技术。

2018-12-28 08:49

WTF

“以终为始”，最常见的一个实践就是计划倒排了。先定时间，然后看功能是不是做不过来得砍掉一些，人力是不是不够需要补充一些，提前预知规避风险。

2018-12-28 00:14

作者回复

你说得对，倒排时间表不是错，倒排时间表却不调整需求范围和资源配置就是问题了。

2018-12-28 08:38

喜悦

今日概念：

1. 想象共同体：集体对同一个目标达成的共同想象；
2. 任何事物都需要经过两次创造：一次在头脑中，一次是付出实践；
3. 遇到问题倒着想，再用工具模拟“想象共同体”发现潜在问题；

今日总结：

以终为始就是终局思维，这种思维方式会强迫我们的大脑去做计划。借用工具将想法表达出来更是有利于发现潜在问题，创造出“想象共同体”；

2019-01-01 22:20

长期规划

以终为始，挺有新意的想法。定好目标后，倒推时间表，安排好每一步要做的事情。我的观点是先要通盘梳理每个节点要做的事情，最后考虑节点之间的依赖关系，再调整节点要做的事。有些时候，我们到了某一步才发现之前的做法有误，再纠正的成本会有些高。提前考虑这种情况很有必要

2019-01-12 19:26

Bacon

以终为始，先考虑清楚结果，再反过来想怎么出发，我觉得就是一种逆向思维，另外，前期的构思确实是十分的重要！

2019-01-11 16:11

sprinty

以终为始的思想让我想到了<石头汤>的典故。

2019-01-10 18:00

王维

这确实是一种思维方式，从结果反推前导，这条原则不仅适用于工作，而且还适用于生活。在具体的开发工作中，我也使用过这个原则，例如在做一个功能之前，我会画一个原型图，先让要使用的人确认，如果ok我们就动手开发。

其实我们在生活或者工作中有意或者无意都用到了这个方法。

2019-01-08 17:06

作者回复

赞！

2019-01-09 08:51

MarksGui

以终为始的思想给了我很大启发！最近发版遇到很多问题，大部分都是发版前才发现。这样再进行二次改造成本是非常高的。非常感谢，我想是时候改变一些策略了

2019-01-03 00:30

彩色的沙漠

本节总结

1、任何事物都要经过两次创造:一次是在头脑中的创造，也就是智力上或者第一次创造，然后付出实践，也就是实际的构建或者二次创造。第二次创造的成本很高，需要在第一次创造花费更多的时间和精力。但是如何把握第一次创造的时间成本的度?期待后面的课程

2、做软件团队也是想象的共同体，而我们做的软件就是将我们的"集体想象"落地的载体。

3、遇到事情倒着想，"以终为始""可以帮助我们在软件开发中规划工作和发现工作中的问题

2019-01-02 10:38

0bug

总感觉以终为始这个词很别扭，目标驱动感觉更容易理解

2019-01-02 09:33

Apollo

大道至简，其实就是充分讨论后再实施。把所有过程在大家充分讨论后再实施，而不是一言堂，个人拍板，企业的扁平化管理也是配合这个目标，国家议会的形成也是这个目标

2019-01-02 08:40

davidce

"软件行业最大的浪费是做出来东西没人用"，作者能否讲讲如何避免这种情况发生?

2019-01-01 01:34

作者回复

后面讲的内容就是在一步步展开这个话题，敬请期待。

2019-01-01 21:34

阿狸爱JAVA

反直觉思维方式：遇到问题，倒着想。

2018-12-31 15:54

诺

老师的这节课让我受益匪浅，茅塞顿开，

也许是水平有限还沉浸在技术代码阶段，没有跳出到业务场景，

所以希望老师能多结合实际开发对接场景来讲解，同时也希望同学们能结合当天老师的课程列出自己遇到实际开发的问题，老师在选举几个有代表性的场景结合当天知识做详细讲解，

以上是我的愚见，望采纳！！！

2018-12-30 12:22

作者回复

感谢你的建议，这篇文章属于“以终为始”主题下的综述。后续更新的内容，很多都是结合具体应用场景展开的，欢迎持续关注。

2019-01-03 18:45

Eric

坚持先写测试case是蛮好的，没想到自己也是以终为始。后续需要多思考，在更多场景用起来。

2018-12-29 20:42

李祥乾

TDD也是一种以终为始。

2018-12-29 19:58