






全栈营 (/dashboard)


学习中心 (/dashboard) Meetup (/meetup_groups) 直播室 (/live) 编程比赛 (/competition_season_2)

帮助文档 (<http://docs.qzy.camp/>) 交流论坛 (<http://forum.qzy.camp/>)

 (/conversations)  江帆 ▾

VIP入学手册 (/courses/28/syllabus)  搜索本课程教材

 概览 (/courses/28)  教材 (/courses/28/syllabus)

 作业 (/courses/28/assignments)


 FAQ (/courses/28/faqs)

 动态 (/courses/28/activities)

 积分榜 (/courses/28/leaderboard)

有 0 位同学正在浏览当前页面

元学习框架

 (/favorites?post_id=507)

在第一周开始学习编程之前，我想跟各位分享这套课程背后的学习原理。

在变成全职 Rails 教练前，我是 10 年开发经验的超级资深程序员。当然，在行业里面，很多人知道我甚至自己不仅程序写得好，我还擅长培训出优秀程序员。

关于训练程序员，一直以来我有一套自己的方法。我用这套方法过去几年来训练出了几十个一流的徒弟。这套方法总的来说有点反程序员界的「直觉」与「政治正确」，但却非常的有效。后来我将这套方法改良，做成了公开课。

我的公开课学员背景组成非常多元：有司机、烘焙师、药师、医师、建筑师，等等.....各式你想像不到的背景，但他们共同的特征就是：这辈子从来没有写过代码，最后他们却学会编程了。

学习编程失败的原因

不过，这套学习方法虽然有效，虽然不论背景都能学会，但是成功机率却不是百分之百。于是我也花了很多时间去分析学生的样板，想找出：

- 为什么有人学得会编程，为什么有人死都栽在同一个坑里面
- 学会一套编程语言并且独立开发程序，需要多久的时间
- 及一般人自学编程为什么低效

我发现来来去去，学习低效甚至失败者，无非栽在对于入门编程的四条错误认知上，他们是这样认为的：

- 学习编程必须要有天赋
- 必须在学习编程中「看」懂每一个步骤
- 学习必须要扎实，要从技术基础原理学起
- 学习时复制代码是可耻的

这样的认知造成了「学习初速度低下」。

一旦学习者不能快速地进入「小获胜」状态，找不到成就感，很快的就放弃了。其实这四条固有认知，都是没必要的伪概念。

「初学」方法绝对与「进阶」方法不同

当然，有些程序员朋友，听到这样的理论，应该会反口说这是我编出来的胡说八道。但是你也也许不知道：

- 必须在学习编程中「看」懂每一个步骤
- 学习必须要扎实，要从技术基础原理学起
- 学习当中禁止使用捷径，如模仿以及复制代码

这些是「进阶」「学习编程」应该要有的认知啊！是在「熟悉基本框架」后的「复习」方法，而非「初学编程」时用的。

该如何「初学」编程？

那么，该如何学习编程呢？

Step 1: 建立自信

进入陌生的领域，第一件事就是要建立自信，相信你自己学得会，而且并不孤独。所以我们安排了作业

- 第一课：环境建置
- 第二课：初级练习

让各位第一时间，就感受到进入编程，搭个小网站，你也能迅速上手！

Step 2: 在大脑内建立足够的基础框架

刚开始学习一门程式语言

- 这时候就算有前辈，拼命想解释让你理解
- 或者是自己尝试上网 Google 找资讯

但还是感到「一头懵」，这绝对是绝对正常的，因为大脑无法以「未知解释未知」。

而这往往却是一般人「从入门到放弃」当中的最大一个关键点。

正常人一般人来说遇到「很多知识不懂」的解决方法，是「花更多时间想办法解释弄懂未知的知识」，或者是「想办法去补自己都看不懂的基础知识」，直到自己筋疲力尽放弃为止。

但是，其实最好的方式，是找到行业中「好的高频小套路」，背下来。即使很多不理解也没有关系。

只要了解「输入」什么，可以「输出」什么就行了。

只有在大脑内储存了够多的已知资讯，其他的资讯才能够有效附着，进而用「已知解释已知」。

这也是我们在课前让各位练习背诵：

- 第三课：中级练习（ Rails 101 的最新版）

的原因。

这套课程其实是写 Rails 程序内，最常见的 101 个场景的共有模板。以足球来比喻，就是射门基本动作。

Step 3：小迁移（二周）

而开学之后，我们的第一套课程

- 「招聘网站」，将会是第一个「小迁移」练习。

所谓的「迁移」，就是我们将会带领各位脱离「只会抄模板」的新手学步车，带领各位搭建出类似场景的第一个小应用。

这一个课程，将会慢动作重放，一个小网站是如何被构思制作出来的。而且，你在日常生活中就能具备这个能力。

招聘网站的结构，会与课前的中级练习，非常的类似。甚至你自己就可以依样画葫芦模仿造出来。

Step 4：中迁移（三周）

而第二套课程：

- 「购物网站」，将会是第二个「中迁移」练习。

在这个阶段，难度将会比小迁移复杂上数倍。但是各位将可以在这个项目体验到，程序员

- 是如何从零到有，将一个网站灵感，拆成数十条执行细项的
- 如何按部就班，把这些执行细项化为程式码
- 并且一步一步找到资源，写出未知的功能

各位将会在中迁移中有极大的进度进展。

Step 5：提取练习（代码）

在开学前，我有请各位做几件事，以这样的方式学习：

- 第一遍复制代码。（观察输入与输出）
- 第二遍手敲代码。（观察自己哪里错误）
- 第三遍以后，凭自己的记忆写出代码。（提取练习）

其实，这完全是科学根据的。这样做的原因是：

- 复制代码，就不容易有错误。同学可以观察到「输入」是什么，「输出」又是什么。
- 手敲代码。你就可以比较出「自己打了什么」，又「为什么这样错了」。（但也不容易掉入坑里，因为你不需要去理解太多的知识）
- 凭自己记忆写出代码。目的是为了常用功能直接进入人类的「长期记忆」。

但是，如果各位在这当中，还是蛮横地去查太多未知的东西，就会把自己带入恐慌的坑里。所以千万不要跳入这个坑里。

Step 6：提取练习（文章 / 错误修正）

在这三遍里，我们还要求各位写日记。日记是一套科学模板（ORID）。

这套科学的模板，是为了让各位近距离观察自己的情绪。写了几天之后，你就可以「客观」的观察到，自己的学习进度其实有很大的进展，而不是「懵懵懵懵懵懵懵懵懵」。

不会再被自己的「情绪」影响「进步」。

Step 7：提取练习（错误修正）

再来呢，「错的部分」是一定要记下来的。也要写到博客中。

虽然刚开始，你可能觉得「记错的」而「不记对的」是很傻的一件事。

但是，这其实是一套非常科学的「进步方法」，当你试着这样记了五六个错误以后。你会发现自己傻逼的错误，不可能再错了，进步的效果会变得非常非常的明显。

Step 8: 提取练习（教就是学）

各位应该在 Slack 上看到，有些进展得比较好的同学，开始在教比较落后的同学。「教就是学」，这句话并不是古人唬弄我们的，而是也有科学根据的。

「教」其实就是一部分的「大脑提取练习」。当我们经由感官刺激接受到讯息后，其实大脑里面的记忆痕迹并不深。如果几天内不应用，其实资讯根本无法留得下来。


所谓「教」，就是在讲述过程中，透过创造「自己版本的新记忆」，将这道资讯在大脑上面再刻深三遍。

总结

所以虽然课前作业，貌似我要求各位做的作业，似乎都是非常反常理的。但全栈营所有的学习步骤，都是走在认知科学的最前沿所设计的。若各位同学「放下学习的傲慢」，并「只字不差的执行练习」。

你将会在这个学习过程中，感受到自己巨大的进步。

对本页内容的感受如何？

 我要吐槽



So easy



还OK



崩溃了

← 上一页 (/posts/505)


🔍 可以使用 ← → 键进行翻页

下一页 → (/posts/508)

全栈营

课程资源

关于我们

 在线客服（非技术答疑，工作日10:00-19:00）

课程介绍

学习中心

公司介绍

(/pages/course_intro)(/dashboard)

(/pages/about)

教学团队

帮助文档

常见问题

(/pages/teachers) (http://docs.qzy.camp//pages/faq)

学员心得

交流论坛

联系方式

(/pages/students) (http://forum.qzy.camp//pages/contact)



新生大学 - 软件学院是李笑来对未来世界的实验计划，旨在改变中国的计算机教育。
(http://www.xinshengdaxue.com/)