学习方式

- 温故而知新
  - 整理笔记（跟着**系统**教程（老师）学）
  - 不断的强化复习（每天）
    - 慢慢的把所学的知识，在脑海中形成一个知识图谱。
      - 最起码，知道前端，如`js`中有哪些内容。能够对`js`，对前端开发有一个系统性的认知
      - 有了知识体系后，逐渐的将每个知识点串联到一起，会梳理出一条主干和若干分支。每条分支和主干的关系，每条分支和其他分支的关系做到心中有数。那个时候别人问你什么都能答出来，问一个东西引申到另一个东西也可以，这样才算是基础知识扎实
      - 只有基础知识扎实了，以后提一个需求，会有`n`中方案实现这个需求，然后通过实践、尝试或经验发现某一种方案，从性能、结果等角度，各方面是最好的。这样的话，所谓的技术能力就达到了一个很高的地步了
  - 总结梳理后，输出博客（掘金之类），提高个人影响力
- 学而不思则罔
  - 学习过程中，一定要多思考，脑子绝对不能懒
  - 自己拓展、其他人分享
  - 多尝试
    - 不管对和错，哪怕有一丁点想法，都要写出来
  - 多练习
    - 多敲代码
      - 前三年以内，薪资和敲代码的行数成正比
      - 练就出“手速”和“代码感觉”（不用动脑子了，也可以敲代码，不也会出错，形成肌肉记忆）
- 设定一个切合实际的目标，以激励自己
  - 自律问题
  - 学习的过程不是单纯的学习技术，也需要让自己养成良好的习惯（生活的节奏感）
  - 培养自己的学习、研究能力
  - 培养自己的静心、专注能力

- 对于从来未实现的功能，如果借助第三方库实现，一定要核对清楚安装的版本是否和当前开发环境匹配，样例去`npm`或`github`上找官网案例，百度只能告诉你，这个库能干嘛，具体怎么干，还得看官方案例。
  - 以及基于第三方库，如果还想实现其他功能，`npm`搜索下关键字，`90`概率是已经有了这样的功能库了

入门学习方法

> Crash Course < 3h + 文档 + 实战

学习一种技术

- `Crash Couse`是什么
  - 时长不是太长
  - 并不是教会你这个技术
    - 或者即使教了，但不一定要完全吸收这些
    - 具体技术可以不从这个教程里学
  - 而是告诉你这个技术的蓝图、大纲，告诉你应该怎么学这个技术
    - 比如学`Vue`，这个视频得告诉你，要学`Vue`的基础语法、`VueRouter`、`Vuex`
    - 主要帮助你开拓视野，让你有一个这个技术栈的大局观，当后面真正学的时候，能明白自己学到哪一步了
    - 如`Youtube`上搜索`React Crash Course`，一个多小时的视频肯定是学不会`React`的，但是通过视频大纲可以知道，学会`React`需要掌握哪些点
    - 也就是上一小结所说的，了解一下知识树，再去学知识点
- 文档 + 文字教程
  - 细节，快速上手
  - 官方 + 社区，文档不局限于官网
  - `google`关键字`how to learn ...`
  - 可以去看技术贡献者的博客
- 实战

学习目标

学东西，我一直都强调要首先明确学习目标，这样我们才能去拆解目标，找到一个个最小的学习单元。接下来就看看我给大家学习数据结构与算法确定的学习目标：

确定学习目标

- 知道有哪些常用的数据结构和算法
- 能够写出高性能的底层轮子
- 知道如何进行复杂度分析、性能分析
- 通过不断训练，具备“算法思维”，提高分析和解决实际问题的能力
- 能够自如地应付大厂面试

怎么就知道，某个知识点就是我们要掌握的学习目标呢，得去了解一下这个领域的概览，从整体角度去了解一下所有的内容，各自的模块都有哪些

学习重点