四年前端 2022 再出发

已过去

2022 年已经来到十月,还剩 3 个月数字就要跳到 2023。从事前端行业已经 4 年多，认识和了解这个行业后，依然保持喜爱和热情。但在生活和学习过程中未免会产生一些焦虑感。

7月份时候换了份工作，为什么换工作呢？说来在上家公司呆了差不多三年时间，接触过后台管理项目、移动端、小程序、Node项目。离开原因总结下来主要原因有几点:

1.公司业务简单，重复度高。
2.可以根据自己的喜好做一些相关的基建项目，但是缺乏标准，有时候感觉在单兵作战。
3.对前端理解和认知比较浅，实际项目中也缺乏场景，缺少"有挑战性"的项目。

挑战

目前入职的团队规模翻倍和解决的业务场景复杂度提升不少。总结下来目前遇到的挑战有如下:

1.技术栈转换, 从 Vue 改为 React 及其相关的生态。
2.业务场景复杂且影响面关系复杂，对项目不熟悉、对业务不熟悉，导致开发过程中效率低、容易改错东西。
3.多人协作，随着团队规模变大，共同开发时候难免同事之间会互相修改到彼此的代码，需要解决冲突防止，容易产生副作用。

试用期遇到的问题

起初会有一个练手的 Demo 项目给新人做，主要是给新人熟悉下 React 相关技术栈，代码提交规范以及一些环境依赖的配置的熟悉和使用。(个人感觉这个时间可以缩短，早些接触实际项目)。

到了真正接手公司项目，由于此时对公司产品不熟悉，以及开发过程中对功能的影响面不清楚。在需求评审完感觉工作量不是很大。实际开发过程中，和后续改 Bug 中发现这个和难度是直线上升。

来说说我遇到的问题点吧

1. 最直接的感受是业务不熟悉，pd 说的很多点都不知道入口在哪，所以造成修改时只看到自己做的地方，不知道 影响面 有哪些。
2. 其次是技术栈不熟悉，React、RTK 及相关技术，包括公司内部有些自己的库和工具。开始有种一堆道具扔给你，你却没时间看说明书的感觉。
3. 项目复杂，业务复杂，在看项目时候只看到冰山一角，还未形成整体的框架和脉络。开发过程中效率低，解决方案不够通用，修改时造成多处副作用(改了 A 影响了 B)。
4. 不熟悉 immutable 的更新方式，在 Vuex 中都是直接通过直接操作 state 来修改 store 数据，在 redux 中引入 immutable 更新的方式，包括在公司项目的开发过程中都要采用该方式。带来的好处如下:

    - 方便调试，因为未来的操作不会影响之前创建的对象，所以行为变得可预测。
    - 降低 mutable 带来的复杂度
5. 还有一点我觉得挺关键的，以前开发不需要你非常懂业务，大概了解下功能做什么的即可，属于 完成任务型开发 。但是现在会觉得你得 熟悉业务场景 ，这些功能点的整体使用，毕竟你还有业务群解答问题，以及你可以跟产品去 battle 到底这个功能要不要做。

解决方案

针对上面遇到的问题点目前有如下解决方案。

1. 项目代码复杂，根据具体模块尝试画出对应模块的代码执行的流程图，模块及配置项之间的关系图。熟悉项目整体脉络。

    - 熟悉顶层抽象的通用逻辑。
    - Debug 确认 hooks的执行顺序和影响点。
2. 熟悉产品使用文档，熟悉产品功能点，以及功能点实际解决问题点，能够站在产品和用户的角度考虑功能的实用性。

    - 学习了解 sql 语法，熟悉业务操作对应后端的 sql 具体做了啥。
3. 加强 immutable 的开发方式训练，学习 `lodash/fp` 的使用，能够更好的写出 immutable 的代码。

规划

虽然在新公司呆了两个多月，但是感受也是跌宕起伏，从开始认为简单的换个技术栈应该问题不大，到接触实际项目的压力倍增有些喘不过气，调整心态对项目和产品有了初步的认知后重新开始继续奋

战。

考虑自己短期之内的规划方向有以下几个方面:

1. 加紧完成产品的常用的使用场景的熟悉。
2. 项目区块梳理和自己负责模块的代码深入了解。
3. 有规划的做一些自己的项目，和开始有规律的输出一些自己学习和成长过程中的沉淀和感悟。