产品从-1到0

> 本文是写给那些希望了解求是潮产品运营部门在做什么，同时希望加入我们的同学。您可能并不懂什么是"产品"，什么是"运营"。甚至，你可能对它们有错误的理解。所以本文的标题是从-1到0.我们希望你在看完之后能有一个对产品、运营的初步印象，同时也对我们的工作有一些基本的了解。

编撰说明

- 有参考资料的地方务必标注（比如加括号，像这样
- 图像在**src/img/你的名字** 文件夹里，否则无法共享。起名规范什么的之后再说吧...暂时自由发挥
- 有其他需要本地存储的资源在src中新建文件夹即可

前言

什么是产品

首先让我们从**产品**开始。

产品，顾名思义，你所见到的几乎所有都可以抽象的被称之为“产品”。但是对我们来说，我们口中的产品更偏向于互联网产品。互联网产品的概念是从传统意义上的产品延伸而来的，是在互联网领域中产出而用于经营的商品，它满足互联网用户的需求。互联网产品是功能与服务的集成，可能是具象的一个网页，移动应用（手机App），也可以是互联网服务的一种思路，如互联网共享经济。

说到这里你可能已经有些晕了，让我们先从一个大家最熟悉的产品出发——微信。

产品的组成部分

打开微信，首先映入我们眼帘的就是：这个App长什么样。

一般来说，我们把它叫做UI（User Interface），即用户界面的整体设计。

balabala

UI

为什么要先跳过UI呢，因为按盒子精的PPT顺序来的。

UI Stack

UI Stack: 盒子精PPT16页

这里我也没太懂，先跳过

Surface

Skeleton

Structure

Scope

Strategy

接下来开始参考波霸的PPT了

页面布局

主要介绍一些布局方式，参考从0到1，瀑布流什么的。

组件设计

阴影

颜色

尺寸

动效

###

UX

在UI之外，我们还有一个常说的就是UX（User Experience），用户的体验。作为一个产品经理，这也是十分重要的。用户是否觉得好用，体验如何，直接决定了用户的粘度。

尽管我们还不知道UX具体有什么，但我们可以说说怎样才是一个好的UX，一般来说，有以下三个要素：

- Useful - 有用的
    - UX设计一定要能真正满足一些需求，而不是产品经理的自娱自乐。
    - 举个例子：扫码共享电话的设计就显然是一个Useless Design.
- Usable - 易用的
    - UX设计应尽可能减轻用户的负担，让用户觉得“没那么复杂”
    - 举个例子：如果借书需要出示身份证-> 手机二维码 -> 手动输入书的编号 -> 付款 -> 完成借书。用户一定会觉得非常复杂而出于畏难情绪不愿使用。但如果你仅仅需要“刷脸”，借了哪些书会由摄像头判断，身份验证与付款都通过支付宝人脸识别完成，那么用户的负担就被我们降低了。
    - 此外，易用的UX设计可以降低用户的学习成本，使得我们的产品即使是对一个小白也非常容易上手。
- Desirable - 令人渴望的
    - 产品的宿命是被用户使用。因此用户如何想，决定了UX设计的质量如何。只有当用户真正需要，或渴望去使用时，那么你的产品才称得上是一个好产品。

- 当然，Useful, Usable都是决定产品是否Desirable的关键要点。除此之外，是否有足够的新意，是否在竞品中存在优势，甚至当前的用户市场份额，都会影响用户的决断。

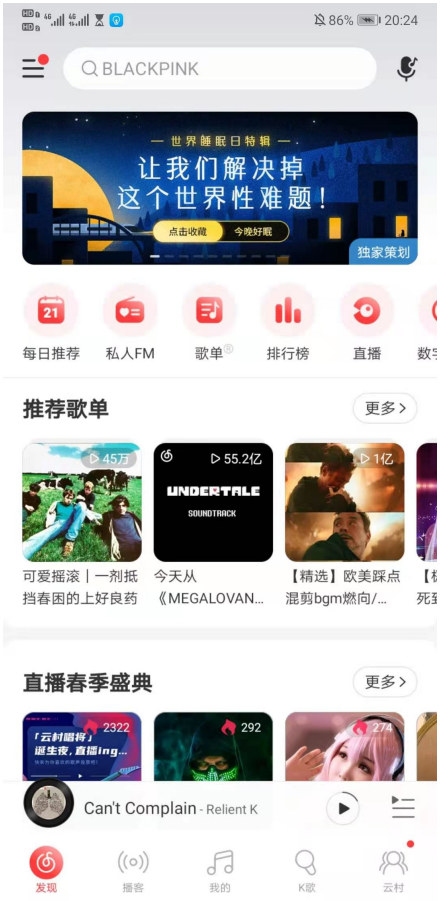
这三个概念足够High Level，囊括了非常多的概念。比如你可能觉得某个产品是不好用的，那么它究竟不好用在哪呢？接下来我们再给出一些更具体的判据。

UX评价

Learnability

当用户第一次使用时，需要多大的上手难度？

我们都知道，人的第一印象非常重要，对产品也是如此。如果用户第一次打开发现不会用，那么他大概率不会反思是不是自己太笨了，而是会愤怒的大喊"垃圾App，根本用不了"并且愤怒地卸载App。举个例子:



这是2021年3月21日的网易云首页，如果你是一位想使用听歌识曲的用户，你能知道需要如何操作吗？

如果你迅速的找到了——没错，就是页面右上角的那个话筒Logo.那你一定是网易云音乐的忠实粉丝了。至少对于编者一个使用了5年网易云音乐的用户来说（以前听歌识曲不在这里），在改版后我需要百度才能找到听歌识曲的位置。这就是上手难度的一个典型例子，对于第一次使用的用户来说，他太不友好了。

一种常见的提高Learnability的方法是——使用新手教程。对于游戏他或许有效，但对于一个希望主动抓住用户的产品来说，用户看完后大概率会忘记。就像一道数学题，当时你会了，再过一会，你还会吗？尤其是对于功能复杂繁多的App来说更是如此。

所以，更多的，我们希望在UX设计的本身就提供给用户更高的Learnability. 如更多的文字提示，让用户看了一眼就知道的Icon（**我们会在icon部分着重讨论这个问题**）。我们并不打算在这里深入展开Learnability的设计思路，那可能需要你海量体验产品后才能有较多的感悟。

但我想你现在有一个测试Learnability的初级想法——找一个没用过的人试一下。这显然可以，但成本过高，毕竟需要提供一个给测试者的demo. 所以更多的还是需要我们产品经理来把握——从一个小白的心态。

Efficiency

这些概念本身也是循序渐进的。当我们已经解决了小白上手的问题，那么对于已经学会使用的用户来说，需要多久他才能完成一个任务呢?

如果你对算法有一些了解，这就像是不同的时间复杂度，我们希望尽可能降低用户完成一个任务的时间。如果你不懂，也可以看我在Usable部分写的例子。完成同样的功能，显然用户会倾向于用时最少的。

如何改善Efficiency呢，有一些值得注意的思路：

如果操作A使用频率是90%，操作B使用频率是10%。你会花更多的时间在优化哪一个操作上？答案是显然的（A，不会有人选B吧）。在产品开发过程中，每一点时间都是宝贵的，把钢用在刀刃上就是这个道理。简单来说——**优先高频需求**

以下的数学题可以用来加深你的印象。

> 三个任务A/B/C要被分配在三个页面，玩家打开第k个页面需要k步，而A/B/C使用的频率大概是50%,30%,20%.如何安排三个任务可以最大化Efficiency呢?

看到这里你可能会觉得我在侮辱你的智商。在现实的场景中，当任务一个接着一个，页面也越来越复杂时，很多年轻的产品经理都会花大量的经历在不那么重要的需求上，我只是为了加深你的印象，以免你换个马甲就不认识它了。

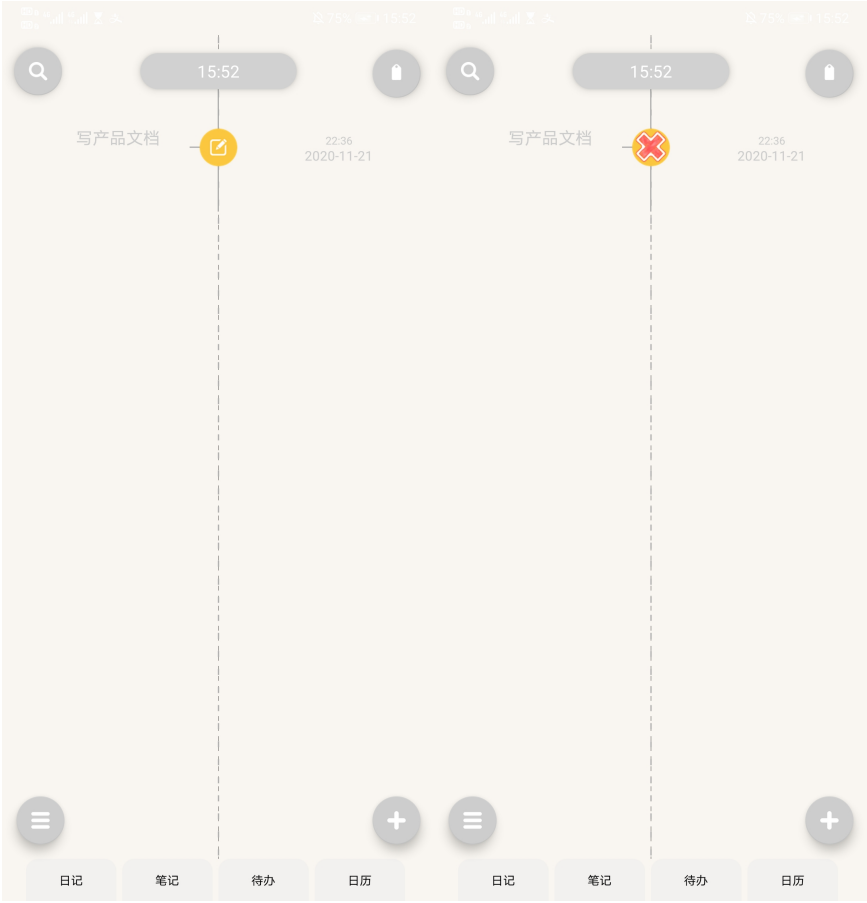
Memorability

并非所有用户一旦使用我们的App就会一直高频使用，挽留那些偶尔使用的用户对我们来说还是十分重要的。那么紧跟而来的问题就是：

当用户一段时间不使用你的产品后再次回归，他需要如何重建熟练度?

首先想到的就是我们用于解决一般Learnability的思路——"让操作本身变得更简单"，如果第一次上手就很容易掌握，那么再次回归只会更容易。此外，Memorability还可以通过增强操作的记忆点，特点来实现，即让用户一旦学会，就再也忘不掉。

以pendo为例，pendo是一款典型的为了交互高效、简洁，舍弃了一部分Learnability的产品。



左图是pendo有一个日程后的界面，就Learnability来说，它并不那么合格，因为没有任何引导告诉我们——右滑这条信息就可以出现删除的选项（如右图）。但就Memorability，右滑也是一个符合人的习惯的交互，且本身比较自然，你甚至不用记住右滑是删除，左滑是收藏。你只要记住每一条message都是可以滑来滑去的这一并不复杂的事情，就可以掌握Pendo的使用技巧了。

Error Rate

我们无法假设用户一定具有粘性，更无法保证用户都是“聪明”的。在我们向用户传达设计理念的时候，一方面我们需要尽可能清晰的传达，即告诉用户他们该怎么做；另一方面我们也需要容错控制，因为总有人会“不按套路出牌”。用户更容易在那些地方犯错？我们是如何处理错误的？以及我们与用户如何可以迅速地从错误中恢复出来？

这部分内容可能有些多，因此我会介绍一些其他概念帮助大家理解。Error Control背后蕴含着更深层次的思想——考虑所有情况。

用户带给你的“惊喜”远远超出你的想象。由于不同用户习惯、地域等等诸方面的不同，他们的行为差异也十分巨大。我们的App考虑的更全面，所留住的用户就更多一分。并且即使是一个如你一般的用户，也有可能有手抖，误触（手机放兜里没锁屏）等情况的发生。因此Error Control并不是一个出现概率低、无意义的工作，与我们先前所说的“优先高频需求”并不矛盾。因为Error Control就是每个人交互过程中不可或缺的一环。

首先我们先明确：什么才是错误的行为。你想聊天的时候点到了付款，这时付款是错误行为。你想付款的时候点到了聊天，这时聊天又是错误行为。可见错误不是针对行为本身的，而是与我们的目的有关的。与我们想做的行为不同的，都可以理解为一种错误。

接下来回答第一个问题：用户更容易在哪些地方犯错？犯错的原因一般有两种

1. 我们没有正确传达正确的使用方法，而这是Learnability, Memorability需要着重考虑的。
2. 手抖、误触等等无意的错误操作，这是我们这里着重讨论的

我们需要明白什么情况下会出现第二种情况。设想你的微信每个对话框旁就有一个大大的支付按钮，点击支付以后立马就付了钱。那么很显然，这种情况下，哪怕你知道这个点了没什么好处，但还是非常有可能出现错误。以上的例子太过极端，但是却涵盖了所有的情况，下面我们将一一展开

1. 交互过于相似
    - 事实上这已经不能算一种情形，而是错误出现的原因。如果A操作是把手机扔到天上，那你大概率不会弄错这个功能。因为除了当你需要使用这个功能时，很难想象你为什么要把手机扔到天上。
    - 大多数错误发生都是由于交互过于相似，比如点击和长按，一不留神就按过头了；两个距离很近的点击，一不小心就会点错……
2. 高频功能与易错功能过近
    - 如果每次手抖的概率是一定的，那么功能越高频，手抖的概率也就越高。而上述例子中，聊天是微信最高频的功能，在对话框旁放上一个容易出错的地方，那么这个错误出现的频率就会较高。
    - 由此引出的一种设计思想就是——不要将高频功能与其他功能的交互设置地过分相似，否则错误率就会显著的提升。

3. 错误触发过于容易
    - 在例子中，仅仅一次点击就触发了“付款”的结果。如果误触一次的概率是p，那么连续误触n次的概率就是p^n。如果我们需要一系列复杂的操作才能达到一个“结果”，就可以显著降低错误的概率。

上述的思路主要是“错误如何发生”，第二与第三个问题则注重于“错误如果发生了我们应该怎么办”。

上述的思路似乎并不能给我们带来太好的思路，因为我们在定义错误的时候，每个功能都可能被认作“错误”，我们又怎么能够针对性设计呢。在这里不妨想想，如果聊天出了错，你会如何？撤回，解释，好像没什么大不了的；但如果支付出错了呢，可能就会给你带来经济损失。相比之下，尽管二者都有可能出错，但我们可以粗略的把“支付”定义为易错功能——因为它的后果严重。基于此，我们就可以回答第二第三个问题了——努力通过设计让错误的后果不那么严重。

综上，以下是一些常见的设计思路

1. 提高各交互的“距离”
    - 这个思路就是针对上述的1，希望从源头上解决问题。
    - 0-1二进制信号的电压标准就是很好的例子。比如我们把0.3V看做低电平，记做0而1.2v看做高电平记做1，那么我们可能可以很好的通过电压得出0-1的数据流。但如果用0.3v来表示0，0.4v来表示1，那么两个之间的“距离”过近，就难以区分，容易发生错误。交互设计也是同理，我们希望尽量明确两个功能在交互上的区别。
    - 考虑上述2提到的情况，尤其是需要与高频功能的交互分离。
    - 但有时为了提高交互的效率，我们不得不安排较多的交互方式，这时我们仍有其他的方法需要考虑
2. 降低错误后果
    1. 解耦合
        - 有时候一个操作的结果是多个结果的耦合。比如支付，实际上是支付—对方收到的耦合。如果我们可以解开中间的这条链，那么付错了但对方却不会收到，错误的后果就被降低了。微信转账的延迟到账就是这样的思路。
    2. 可撤销
        - 如果你的错误是可撤销的，比如微信消息的撤回。那么即使你出了错，后果也会降低很多。一些文档自动保存（误关闭）也是此类的道理。
3. 提高触发难度
    - 对于易错的操作，我们可以通过增加交互难度来降低错误率。也就是对标前述的3。尽管在交互涉及的部分我们将会提到——要尽量减少每个操作用户点击的次数。但对于这种较为重要，出错结果严重的功能也可以适当的增加中间过程，常见的方式如：
        - 二次确认
        - 交叉验证
            - 如使用手机验证码+密码 之类

一个小练习：下述Error Control机制适用了哪些思想？

微信付款时，需要点击确认付款后才能指纹支付。值得一提的是：对于有实体按键的手机，指纹支付和返回是完全重合的交互（轻按指纹识别器）

Satisfaction

Satisfaction更多的是对上述因素的总述：用户的整体体验是否愉快。在这一部分，你可以把UX与UI与交互结合起来考虑。反馈感，icon设计，微动效，页面层次。我们可以在其他部分详细讨论这些内容，这里就不多做展开了。

Interaction Design 交互设计

在这之前，我们更多的是在讨论一些静态的设计。现在我们进入到交互设计，让我们的页面动起来。

交互是一个比较抽象的概念，简单来说，交互就是我们和页面相互交流的互动。你的点击，滑动，拖拽，长按都属于交互的范畴。

交互设计的共识

盒子精PPT

在了解交互之前，我们先达成一个共识：

> **Miller's Law**
>
> 大部分人在他们的短期记忆中只能保存大约七条信息

> Fitts' Law

> Fat Finger Problem

> Hick-Hyman Law

5 dimentional

Words

Visual representations

Physical objects or space

Time

Behavior

Principle

Discoverability

Consistency

Learnability

Readability

硬件交互

Some Tips

反馈感

用户的等待成本

盒子精ppt

具体的平台开发

Android

这里可以参考波霸的PPT写一点Material Design

iOS

网页

求是潮产品线

历史

现在

工作流

产品是如何工作的

如何高效地与其他部门沟通与合作

沟通技巧

敏捷开发