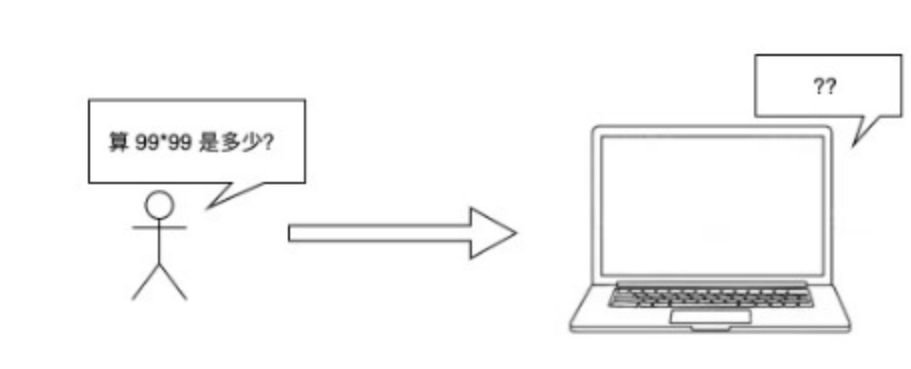
一、语言是信息的载体

语言最重要的价值在于 信息的传递 。现实中通过 汉语 、 英语 、 法语 等都可以实现沟通，不同的语言可以传递相同的信息。语言是什么其实并不是最重要的，它只是信息的 载体 ，一种表达的 工具 。

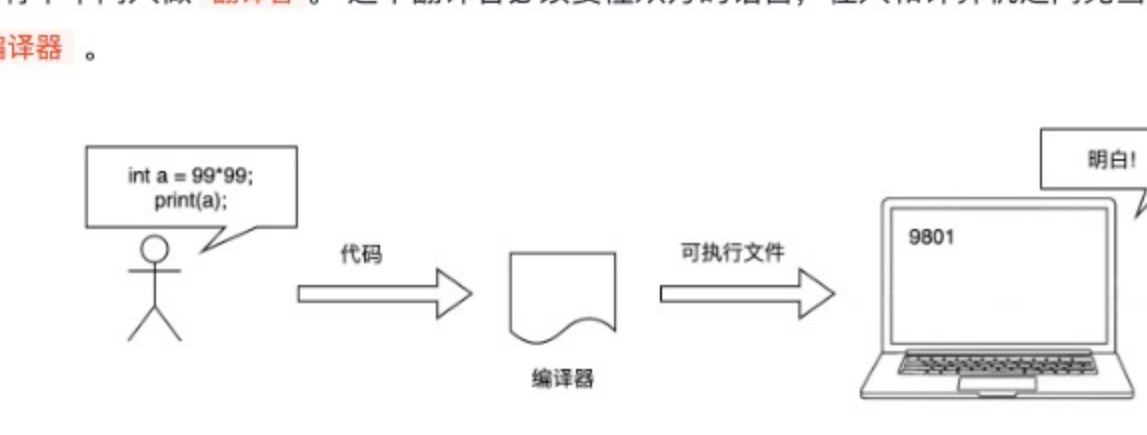
```css
我爱你
I love you
je t'aime
ich liebe dich
愛してる
```

1. 编程语言存在的必要性

编程语言同样是一种表达的 工具 ，只不过我们通过编程语言传递的信息，并不是来抒发个人情感，而是命令计算机 执行任务 。可以想象你是个无情的 “奴隶主”，你的任何命令，计算机都会 毫无反抗 地执行。但是想让计算机这个 “死物” 听懂人话是比较困难的。



在现实生活中，一个人只有学会语言，才有与别人沟通的基础。对于计算机来说也是一样，它听不懂你的语言，你就没有办法直接命令它做事。现实中，当你要和语言不通的人交流时，那就必须有个中间人做 翻译官 。这个翻译官必须要懂双方的语言，在人和计算机之间充当翻译官的就 编译器 。



那问题就来了，如何让编译器 “听懂” 人的语言呢？人类的社会语言自然不堪此负，所以就需要定义 “新的语言” 来折中，作为 “沟通的桥梁” ，而这就是 编程语言 。编译器即可以识别人类写的 编程语言 ，也可以将 源文件（代码） 编译成计算机可以看懂的 “二进制可执行文件” 。这就是编程语言存在的必要性，这个 “无中生有” 绝对可以算作人类创造史上的一大奇迹。

2. 代码是给谁看的?

代码 相当于 编程语言 的文字，首先我们要明白一点， 代码 是写给谁看的。代码的最终目的是给计算机发送命令，让它为我们 “办事” 。代码是 因 ，计算机满足人的需求，进行输出是 果 ，而且每个因都对应着唯一确定的 果 。无论是 编译器 还是 计算机 都是 死物 ，而对 因 (代码) 进行维护的是 人 ，所以 代码 是写给人看的。

编程界流传着一个非常真实的梗， 程序员最讨厌有两件事：

```csharp
[1]. 看别人不写注释的代码
[2]. 在自己的代码中写注释
```

代码的 可读性 是非常重要的，我希望新入坑的朋友对这点 铭记在心 。而代码的可读性:

```shell
20% 在于结构
30% 在于命名
50% 在于注释
```

可能很多人在初级阶段都经历过：自己的代码一个星期后再读，就感觉面目全非了。我们要记住，代码是 写给人 看的，只有人能看懂，代码才能长久地维护下去。

二、框架是通用的解决方案

比如一年级的老师布置假期作业，需要每天写一篇一句话的日记。这个作业就是一项 需求 ，而框架就相当于一篇日记模板。它会给出针对某问题的 通用 解决方案，我们只要提供某些必要的信息，即可完成这项需求。

1.框架的核心语言与流程规范

每个框架都会有其对应的核心语言，比如说语文老师布置的日记有汉文模板：

```markdown
____ 年 __ 月 __ 日     天气 __
 正文xxx
```

英语老师布置的日记有英文模板：

```sql
____ Year __ Month __ Day   Weather __
  body xxx
```

从这里也可以看出， 框架模板 在一定程度上可以起到 规范格式与流程 的作用。

2. 框架会简化使用者的操作

模板 除了规定格式与流程之外，还可以封装一些通用的操作，来简化使用。比如把一句话的日记模板简化成如下的模式，这样完成 写日记 的需求只需要在正文中填个名词即可：

```markdown
____ 年 __ 月 __ 日     天气 __
 今天我去 ____ 玩，玩得很开心。
```

日常开发中，我们并不会从 `0` 开始和底层硬件打交道。无论是 `Android` 、 `Web` 、 `iOS` 、 `Windows` 、 `MacOS` 的软件，都是在框架的基础上进行开发，框架层一般称之为 `framework` 。我们作为应用层的开发，就像根据模板来 填空 。 因为一些基本的通用逻辑与流程，并没有必要让每个开发者都自己实现。

而且这些实现逻辑起来也是非常有难度的，为了尽可能简化使用的门槛，框架中会做大量的封装处理。就像是电脑的内部构造是非常复杂的，使用者刚接触时没有必要和里面的零件打交道。

通过外壳的封装之后，我们可以通过 键盘 、 鼠标 、 屏幕 这些 外部暴露 的部分，完成使用电脑的需求。同样，框架的内部封装也是如此，它可以极大地方便使用者的操作，避免直接和复杂的事物打交道。

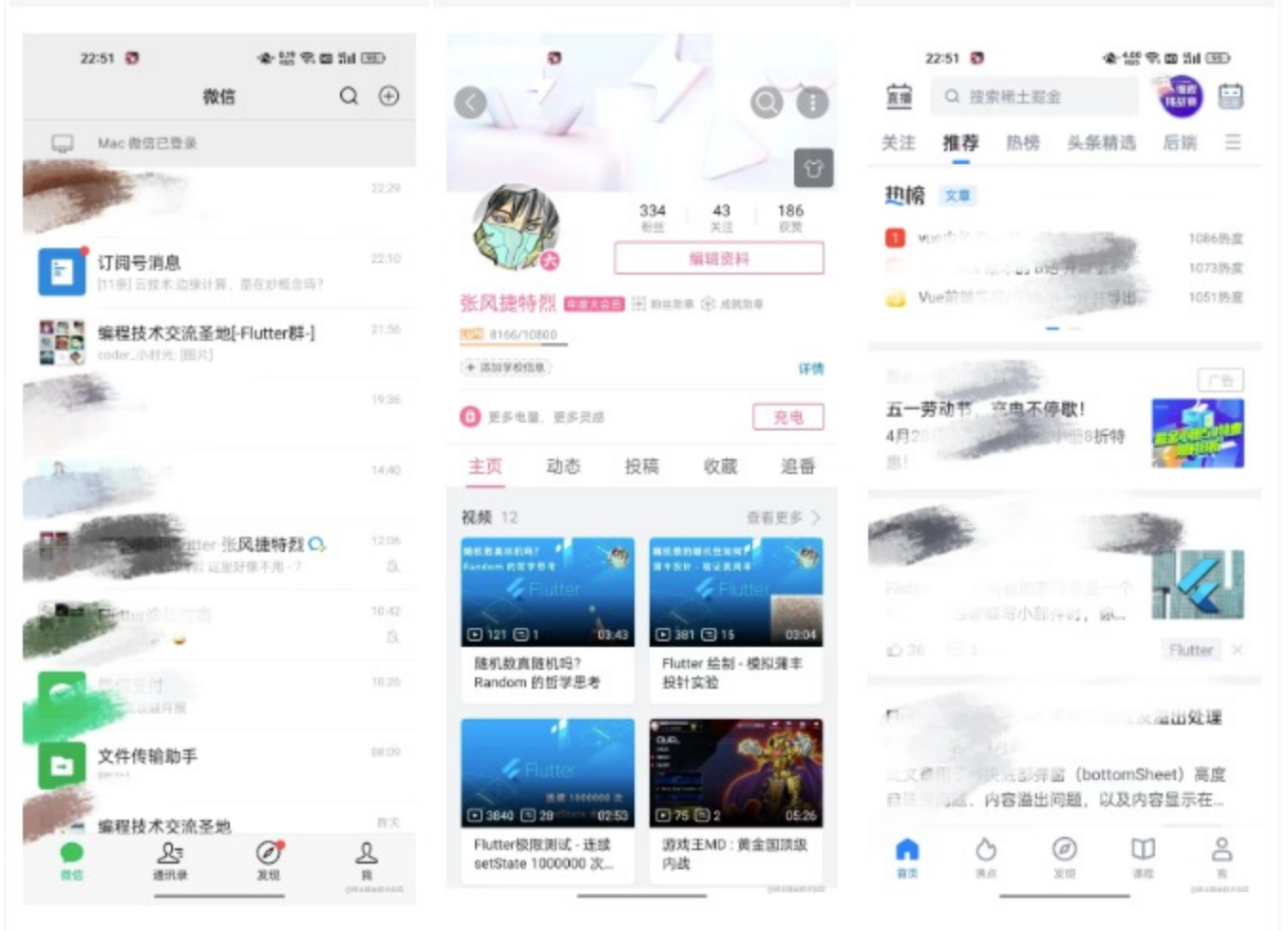
三、应用是满足需求的最终产物

语言 也好、 框架 也罢，本质上都是 工具 。而工具不是目的，而是手段，我们最终的目的是为了做出满足需求的 应用 （或说是 软件 ）。拿上面作文模板的例子类比，应用就是那篇交给老师的日记，而作文模板只是一个中间过程。

1、应用的唯一价值是满足使用需求

打开你的手机，其中的任何应用都有其特定的使用场景，通过 微信 可以实现聊天、通过 B 站 可以看视频，通过 掘金 可以看文章。需求决定着应用存在的价值，只有应用有人用、有使用的场景，它才算是 “活的” 。

微信	B 站	掘金

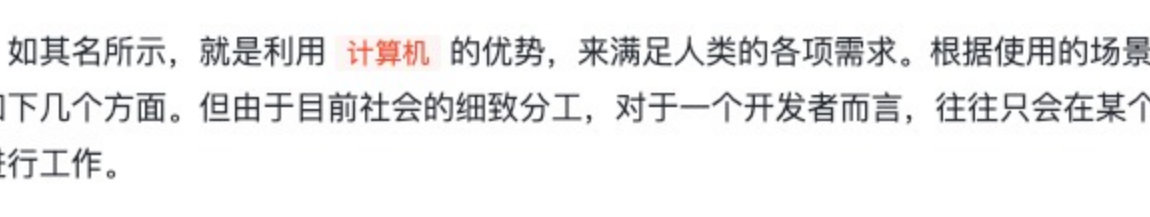


编码能力固然重要，但对应用需求的准确分析也是开发的重要环节之一。知道自己要做什么、在做什么，才能知道该怎么做。就算你语言功底再好，辞藻再华美，高考作文写不到点子上也是白搭。

2.广义的应用

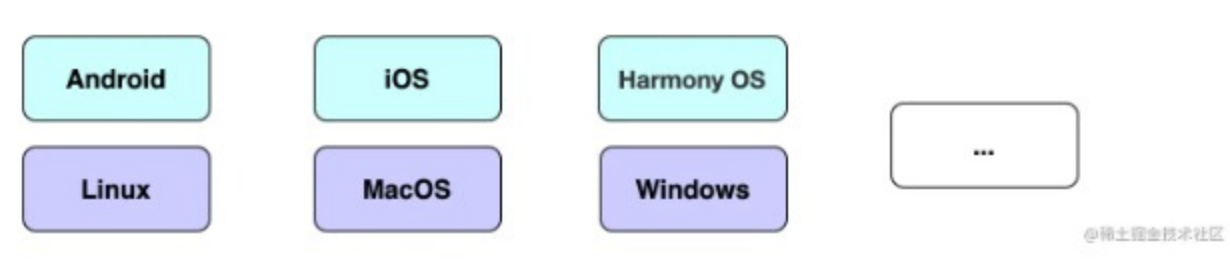
也许很多人觉得 应用 就是手机里面装的软件，其实把视野放宽一点：对于一切设备、在操作系统中运行的、满足人类生产、生活需求的，通过代码进行编写的软件都可以称为广义的应用（ `Application` ）。

应用，如其名所示，就是利用 计算机 的优势，来满足人类的各项需求。根据使用的场景，主要分为如下几个方面。但由于目前社会的细致分工，对于一个开发者而言，往往只会在某个固定的领域进行工作。



目前主流的操作系统如下所示，随着计算机技术的发展，移动设备逐渐摆脱了曾经性能的瓶颈，各种嵌入式的智能设备也层出不穷。操作系统、平台间 应用需求 的差异性被逐渐缩小。各种跨平台技术应运而生， `Flutter` 作为一种跨平台应用开发的解决方案，也已经开源三四年了，目前生态环境已经算是不错了。

以前，每个操作系统都需要使用对应的语言、框架进行开发，可想而知一个应用需要满足在多个操作系统上运行，是多么困难的事，需要耗费大量的人力、物力。就像是战国中，每个国家有自己的语言，写本书在各国流通，需要有 7 国 版本。



这也是 跨平台开发 想要解决的痛点，但跨平台也不是那么简单的，涉及到各个平台特性时，也需要单独进行处理，这是不可避免的。有人说， `Flutter` 只是个 `UI` 框架，这只能说明他对 `Flutter` 的认知是浅薄的。当了解 `Flutter` 与各平台间的 通信系统 之后，就会明白 `Flutter` 不仅是 `UI` ，还可以连接平台进行操作。这也就是常说的 平台插件 ，只不过绝大多数 `Flutter` 开发者都不会自己写而已。

3. 碎碎念

如果说， 代码 就是 文字 ，那 应用 就相当于产品的 作文 或说是 书籍作品 。我们从小开始学习语文时，是从 字 、 词 、 句 、 段 、 篇 、 章 逐步进行学习的。编程也是类似，一开始认识简单的 关键字 ，学习 语句 的书写，通过 函数 将若干个语句进行封装，通过 类 对成员和函数进行封装；通过 包 对类文件进行封装；然后各个部分的运行使得 应用 得以完成需求。所以，应用的开发，本质上就是在用代码 “写命题作文” 。

不同的社会语言中有着语法 特性 和 共性 ，其实不同的编程语言也是如此。对于语言的学习，千万别拘泥于语法的条条框框，也不要什么一开始都尽善尽美，没有掉过坑的成长是不完整的。没有哪个文学大师一开始就出口成章，没有哪个编程大佬不是从最基本的语法开始入门。优秀的语句，流芳的佳文是永远也背不完的，我们需要的是理解和思考。

要时刻记住，语言只是表述信息的工具，特别是 编程语言 ，它的目的性非常强，就是为了让计算机解决问题而存在的。对于一个事物，我们要学会抓主要矛盾，这样才不会陷入细节的深渊之中。对于无法立刻明白的概念，也许只是时机未到，你需要更多其他方面的知识基础。这时要学会浅尝辄止，曲线救国，而不是死磕到底，撞碎南墙，牛角钻透。

也许当你在研究其他方面的知识时，会突然对之前的某个痛点豁然开朗。我曾遇过几次瓶颈，在其他领域中被突破，毕竟万物之间的联系是普遍的，有时在局外去思考棋局，就会灵感袭来，豁然开朗。别人的三言两语，也可能让你茅塞顿开。