《Clean Code》- 第 2 章 有意义的命名

2.1 名副其实

> *“变量取名要花心思想想，不要贪图方便，过于简略的名称，时间长了以后就难以读懂。”*

例如，不应该使用 magic num 或者 i、j、k 这类无意义的变量名。

```
// bad
var d = 10;
var oVal = 20;
var nVal = 100;

// good
var days = 10;
var oldValue = 20;
var newValue = 100;
```

2.2 避免误导

> *“命名不要让人对变量的信息(类型，作用)产生误解。”*

例如，platforms 和 platformList，除非 platformList 真的是一个 List 类型，否则 platforms 会比 platformList 更好。

```
// bad
var platformList = {
    iOS: {},
    Android: {},
    PC: {},
};

// good
var platforms = {
    iOS: {},
    Android: {},
    PC: {},
};
```

2.3 做有意义的区分

> *“用明确的意义去表述变量直接的区别。”*

例如，很多情况下，会有存在 product，productData，productInfo 之类的命名，Data 和 Info 很多情况下并没有明显的区别，不如直接就使用 product。

```
// bad
var goodsInfo = {
    skuDataList: [],
};

function getGoods(){};            // 获取商品列表
function getGoodsDetail(id){};    // 通过商品ID获取单个商品

// good
var goods = {
    skus: [],
};

function getGoodsList(){};        // 获取商品列表
function getGoodsById(id){};      // 通过商品ID获取单个商品
```

2.4 使用读得出来的名称

缩写要有个度，比如像 DAT 这样的写法，到底是 DATA 还是 DATE...

```
// bad
var yyyyMMddStr = eu.format(new Date(), 'yyyy-MM-dd');
var dat = null;
var dev = 'Android';

// good
var todaysDate = eu.format(new Date(), 'yyyy-MM-dd');
var data = null;
var device = 'Android';
```

2.5 使用可搜索的名称

可搜索的名称能够帮助快速定位代码，尤其对于一些数字状态码，不建议直接使用数值，而是使用枚举。

```
// bad
var param = {
    periodType: 0,
};

// good
const HOUR = 0, DAY = 1;
var param = {
    periodType: HOUR,
};
```

2.6 尽量避免使用编码和成员前缀等“旧时代”规则

把类和函数做得足够小，消除对成员前缀的需要。因为长期以后，前缀在人们眼里会变得越来越不重要。

例如，抽象工厂及其实现。

```
// bad
IShapeFactgory / ShapeFactory,

// good
ShapeFactory / ShapeFactoryImp.
```

2.7 避免思维映射

> *“聪明的程序员与专业的程序员之间的区别在于，专业的程序员了解：**明确是王道（Clarity）**”*

例如，i 在编程人员眼中，经常代表 index。但最好用当时有意义的名称来替代 i。

2.8 类名

> *“类名应该是名词或者名词短语，不应该是动词。”*

例如， Customer, WikiPage, AddressParser。 避免使用太通用的词， 如 Data, Info 等。

2.9 方法名

> *“方法名应该是动词或者动词短语。”*

例如， addPayment, removePage 或者 save。 如果是属性访问，则加上 "get", "set", "is" 等前缀。

2.10 别扮可爱（卖萌）

> *“不要使用难以联想到的词，即使你认为很精妙，直接了当的命名。”*

例如，whack() 来表示 kill() 等。

2.11 每个概念对应一个词

> *“Controller、Manager、Driver。英语中有很多语义相似的词语，代码中统一使用一个词来描述，并一以贯之。”*

例如，如果 fetch, retrieve, get 并列出现在方法名里，你怎么知道用哪个?

2.12 不使用双关

双关代表者二义性。比如 add，有可能是 insert 的意思，有可能是 append 的意思。

例如，add 方法，如果用于 collection 时，则应该叫 append 或 insert 更贴切。

2.13 使用解决方案领域名称

> *“如果能用计算机科学领域的术语，则尽量使用。”*

例如，AccountVisitor，程序员一看就知道这是访问者模式。 如果不能用程序员所熟悉的术语，则采用所涉问题领域来的名称。

2.14 添加有意义的语境

> *“对于某些名称，在不同语境下可能代表不同的含义，最好为它添加有意义的语境。”*

例如，看到一个 state 变量，可能不知道它是什么意思，但如果是 addrState, 很容易看出它是地址的一部分，如果 state 是 Address 类的成员，含义就更清晰了。

2.15 不要添加没用的语境

只要短名称足够清楚，就比长名称好，不要一昧的追求长名称，从而造成很多不必要的语境，给人以困扰。

THE END