

NEW! STRONGER THAN DIRT

ENTER.

tech with a heartbeat



CAUTION: EVEN BAD CODE CAN FUNCTION
40 FL OZ (1.25 QT) 1.18L

INTRODUCTION

- Names are **essential** in software development.
- The names of variables, functions, classes and package should **reveal their intent**, avoid **disinformation**, be **pronounceable**, be **searchable**, avoid **encodings** and avoid **mental mapping**.
- ***“Say what you mean, and mean what you say!”***

INTENTION-REVEALING NAMES

- Names of variables, functions, classes (etc.) should answer all the big questions:
 1. Why do they exist?
 2. What are they supposed to do?
 3. How it is does it?
- It is even possible to tell a variable's, function's or classe's scope by properly choosing a name (***without suffixes, though ;)***).

! REMEMBER !

If a comment is needed,
then the name **is probably not good!**

! REMEMBER !

If a comment is needed,
then the name **is probably not good!**

Good names work as a kind of “compilable documentation”!

AVOID DISINFORMATION

- Do not leave **false clues** that obscure the meaning of the code and not use names which **meaning vary from context to context**.

```
1 double hp; //vs  
2 double hypotenuse;
```

- Avoid choosing names that give **incorrect/inaccurate** information about how a variable works or is meant to be used.

```
1 auto valuesList = Values.GetValuesStack();
```

AVOID DISINFORMATION

- Beware of using names which vary in small ways;

1	ConvertGlobalToLocal3DCoordinates(); //vs
2	ConvertGlobalToLocal2DCoordinates();

EXAMPLE #1

```
1  uint8_t d;  
2  char myVariable;  
3  static theObject sr_rlrtnk;  
4  vector<int> listOfEntriesNotFoundInTheQueryResult;  
5  // Pointer to an orthogonality-calculation function  
6  // accepting a set of *NORMALIZED* stokes vectors  
7  void (*callback)(vector<stokes>, bool, int);
```


MEANINGFUL DISTINCTION

- Avoid **adding number series or differentiation suffixes** to distinct two or more variables:

```
1 string word1;  
2 string word2;  
3 // If the names are different, they should  
4 // also mean something different!
```

- Do not use **noise words**. Those are words added to a base name to simply make them distinct, without making the names mean anything different.

```
1 Product product; Product productData; Product theProduct;
```

! REMEMBER !

Suffixes and Prefixes can still be added to names though, but they should add **meaningful information!**

BAD DISTINCTION EXAMPLE

```
1 getActiveAccount();  
2 getActiveAccounts();  
3 getActiveAccountInfo();
```

What is the difference between those three functions?

BAD DISTINCTION EXAMPLE

```
1 getActiveAccount();  
2 getActiveAccounts();  
3 getActiveAccountInfo();
```

What is the difference between those three functions?

How would you know which one to call?

ENCODING

- Usually encoding provides **extra information** about a **variable type or scope**;
- Encoding adds an extra burden of deciphering the name;
 - Unnecessary mental burden when trying to solve a problem;
 - Above it all: encoded names are hardly pronounceable and are easy to mis-type;

ENCODING: HUNGARIAN NOTATION

- Adds a **1 letter prefix** to a variable name, representing its type.
- Useful (back in the day) when compilers did not check type. Nowadays, though...
- Similar to the Hungarian Notation are those OOP prefixes:
 - Add m_ to member attributes;

```
1 m_dailyEventsCounter;
```

- Add I- before the name of an interface;

```
1 IRenderer
```

UNCLE BOB'S NAMING GUIDELINES

- **Class names** should be noun or noun phrases;
 - Customer; WikiPage; Account; AddressParser;
- Avoid using buzz words like **Manager**, **Processor**, **Data** or **Info**;
- ***A Class name should not be a verb!***

UNCLE BOB'S NAMING GUIDELINES

- **Method names** should have verbs or verbal phrases;
 - `postPayment`; `deletePage`; `save`;
- **Accessors**, mutates and predicates should be named from their values and prefixed with `set`, `get`, `is`, etc.
 - `setContext`; `getServerStatus`, `isClientConnected`;

UNCLE BOB'S NAMING GUIDELINES

- **Booleans** should have predicates:
 - isDirty; isUpdated; isEmpty
- Those read nicely along with if statements:

```
1 If (userWallet.isEmpty)
```

UNCLE BOB'S NAMING GUIDELINES

- **Enumerations** are usually named after adjectives:

```
1  enum state {  
2      headless,  
3      inactive,  
4      detached,  
5      connected,  
6      halted,  
7      Offline,  
8  };
```

THE SCOPE LENGTH RULE

- The **longer the scope of a variable, the longer its name must be!**
 - Variables declared and used in a small scope can **have very small names** (even one letter);
 - Member attributes, on the other hand, **should have longer and descriptive** names.
- Functions and classes **follow the opposite rule!**
 - Public functions should have **short names**;
 - Derived classes have longer names, because they should have **adjectives** added to their names!