

How to name things:

the hardest problem in programming

@PeterHilton

<http://hilton.org.uk/>

15 years as
the dev team's
only native
English speaker...



A blue-tinted photograph of George Orwell sitting at a typewriter, smoking a cigarette. The background shows a complex network of cables and equipment, suggesting a wartime or intelligence setting. On the left side of the image, there is a vertical bar with three colored segments: teal at the top, yellow in the middle, and orange at the bottom.

George Orwell's rules for naming

How to name things, by G. Orwell

‘What is above all needed is to
let the meaning choose the word,
and not the other way around.

... the worst thing one can do with
words is surrender to them.

‘When you think of a concrete object, you think wordlessly, and then, if you want to describe the thing you have been visualising you probably hunt about until you find the exact words that seem to fit it.

‘When you think of something abstract you are more inclined to use words from the start, and unless you make a conscious effort to prevent it, the existing dialect will come rushing in and do the job for you, at the expense of **blurring** or even **changing your meaning...**

1. Never use a **metaphor**, simile, or other figure of speech which you are used to seeing in print.
2. Never use a **long word** where a short one will do.
3. If it is **possible to cut a word** out, always cut it out.
4. Never use **the passive** where you can use the active.
5. Never use a foreign phrase, a scientific word, or a **jargon** word if you can think of an everyday English equivalent.
6. Break any of these rules sooner than say anything outright barbarous.

‘These rules sound elementary,
and so they are, but **they demand
a deep change of attitude** in
anyone who has grown used to
writing in the style now
fashionable.’

Politics and the English Language

(1946)

1. ‘Never use a metaphor, simile, or other figure of speech which you are used to seeing in print’

(beware of over-using design patterns, and using their names just because you’re used to seeing them in code)

e.g.

AbstractConfigurationFactory

2. 'Never use a long word
where a short one will do'
(prefer concise variable names,
use longer names for a good reason)

e.g.

company_person_collection

VS

staff

3. 'If it is possible to cut a word out,
always cut it out'

(avoid additional words that don't add any
meaning to a name)

e.g.

AbstractObjectFormatterProxy

...

`org.springframework.web.servlet.support.
AbstractAnnotationConfigDispatcher
ServletInitializer`

‘This is like homeopathy.
What you’ve done is
you’ve diluted the meaning
until it’s all gone.’

`@KevlinHenney`

4. 'Never use the passive
where you can use the active'
(respect grammatical rules for identifiers)

e.g.

class PlanEvents

vs

class EventPlanner

or even

class Scheduler

5. ‘Never use a foreign phrase, a scientific word, or a jargon word if you can think of an everyday English equivalent’

(don’t let technical jargon from a library pollute your domain model)

(beware libraries that import ‘foreign’ naming from one language to another)


e.g. **ShipmentMonad**

6. ‘Break any of these rules sooner than say anything outright barbarous’ (don’t blame me if your code is featured on The Daily WTF)

Note: a lot depends on context; publishing library code is not the same as maintaining private application code

It sounds like
writing prose
is as hard as
writing code.

Who knew?



Advice from other writers



Stephen King on pair programming

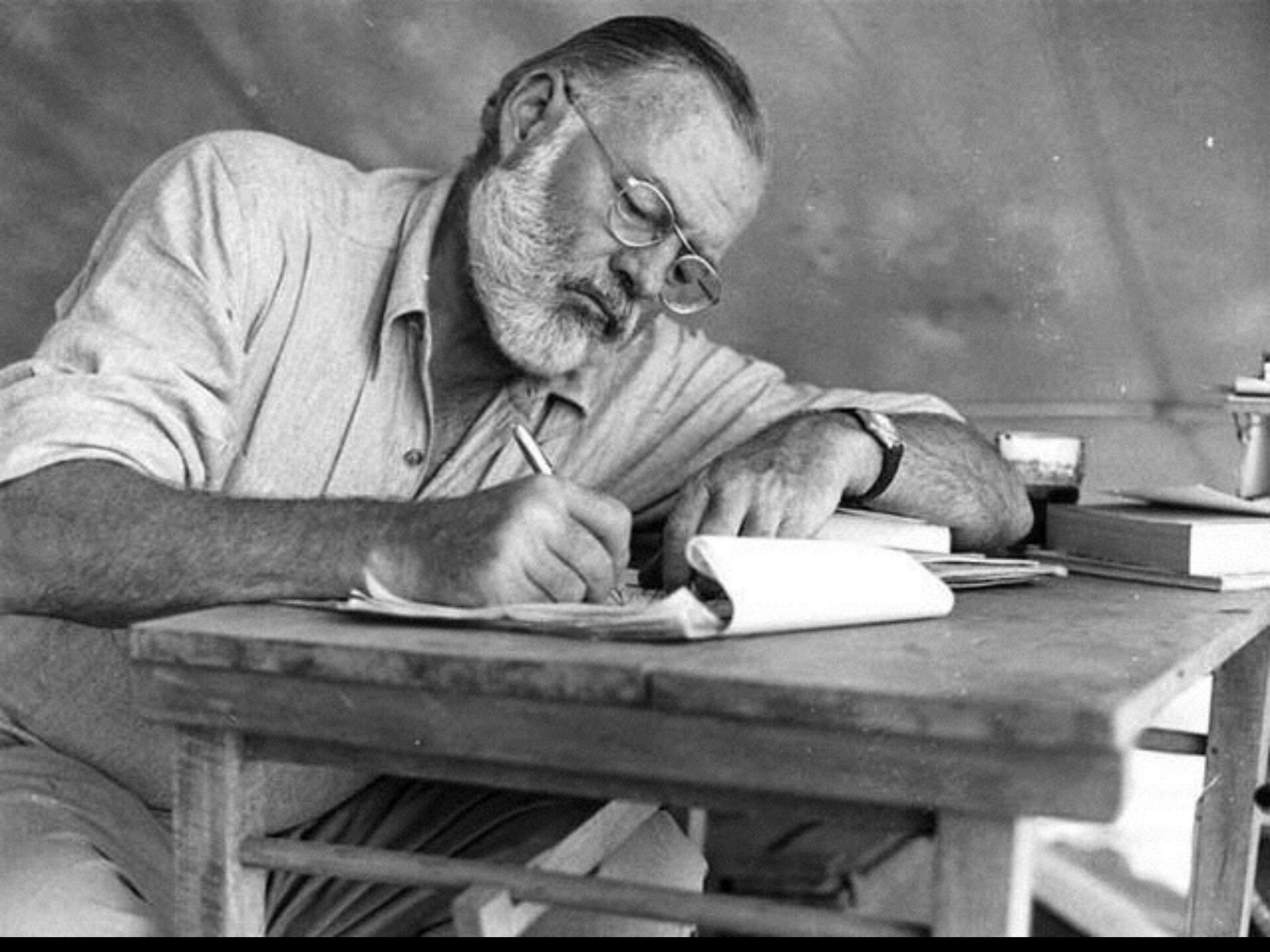
‘Write with the door closed, rewrite with the door open.’





Anne Rice on development hardware

**‘I find the bigger
the monitor,
the better the
concentration.’**



Ernest Hemingway on user personas

‘When writing a novel a writer should create living people; people not characters. A character is a caricature.’





W. Somerset Maugham on
enterprise architecture

**‘There are three rules for
writing the novel.
Unfortunately, no one
knows what they are.’**



Neil Gaiman on productivity

‘Write.

‘Put one word after another.

Find the right word, put it down.

‘Finish what you’re writing. Whatever you have to do to finish it, finish it.’

Neil Gaiman on code review

‘Put it aside.

**Read it pretending you’ve never read it
before.**

**Show it to friends whose opinion you
respect’**

Neil Gaiman on review feedback

‘When people tell you something’s wrong or doesn’t work for them, they are almost always right.’

‘When they tell you exactly what they think is wrong and how to fix it, they are almost always wrong.’

Neil Gaiman on refactoring

‘Fix it.

‘Remember that, sooner or later, before it ever reaches perfection, you will have to let it go and move on and start to write the next thing.

**‘Perfection is like chasing the horizon.
Keep moving.’**

Neil Gaiman on humour in code

‘Laugh at your own jokes.’

Neil Gaiman on open source

‘The main rule of writing is that if you do it with enough assurance and confidence, you’re allowed to do whatever you like.’

Summary of advice from writers

Advice from writers is useful, and not only about naming.

Writers have been at it for centuries; programming is merely decades old.

Also, their advice is better written.

And funnier.

Getting it right
means a struggle
for every single
word.



Naming things

badly

Phil Karlton on naming

‘There are only two hard things
in Computer Science:

0. off-by-one errors

1. cache invalidation and

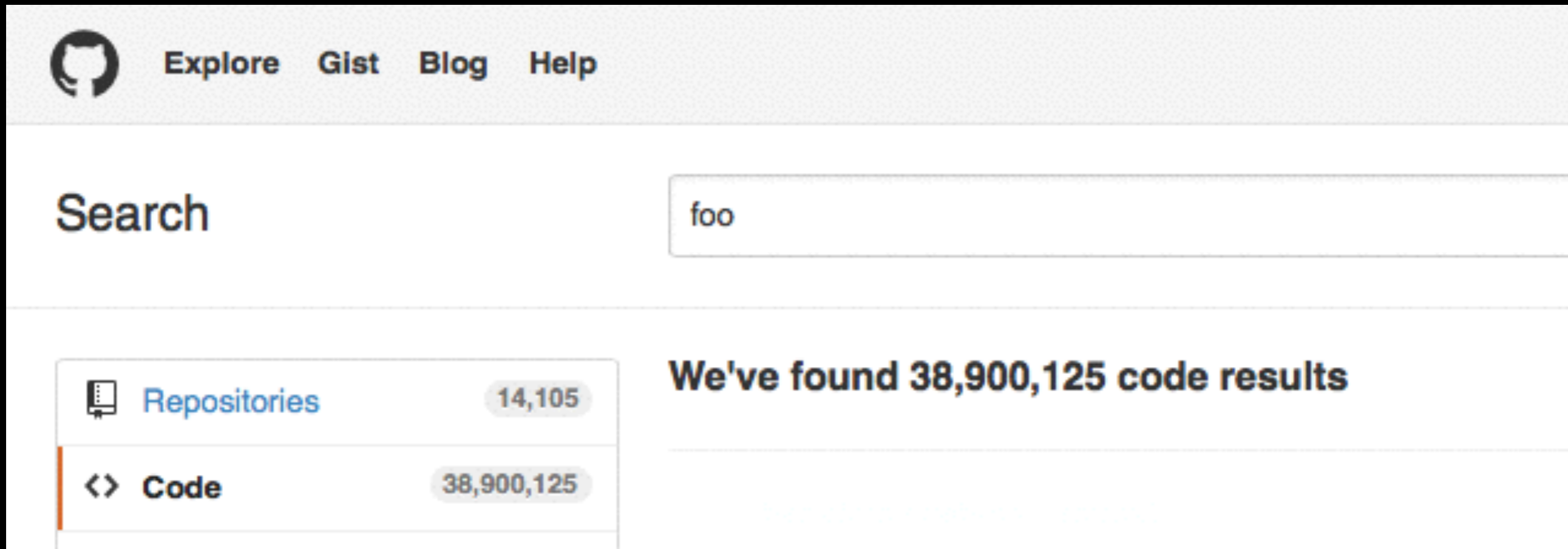
2. naming things.’

Lewis Carroll on bad naming

‘When I use a word,’
Humpty Dumpty said,
in rather a scornful tone,
‘it means just what I choose it to mean -
neither more nor less.’

*Through the Looking-Glass,
and What Alice Found There (1871)*

Deliberately meaningless names



The screenshot shows the GitHub search interface. At the top left is the GitHub logo. To its right are navigation links: "Explore", "Gist", "Blog", and "Help". Below the logo is the word "Search". To the right of "Search" is a search input field containing the text "foo". Below the search bar is a summary of results: "We've found 38,900,125 code results". On the left side, there is a filter menu with two items: "Repositories" with a count of 14,105, and "Code" with a count of 38,900,125. The "Code" item is currently selected, indicated by an orange vertical bar on its left.

In theory, **foo** is *only* used as a placeholder name (because it doesn't mean anything)

Sam Gardiner on naming

‘If you don’t know what a thing should be called, you cannot know what it is.

If you don’t know what it is, you cannot sit down and write the code.’

<http://97things.oreilly.com/wiki/index.php/>

[A_rose_by_any_other_name_will_end_up_as_a_cabbage](http://97things.oreilly.com/wiki/index.php/A_rose_by_any_other_name_will_end_up_as_a_cabbage)

What is the worst ever
variable name?

data

What is the second-worst name?

data2

What is the third-worst name ever?

data_2

Abbreviations are ambiguous

Is **char** a **character** or **characteristic**?

Does **mod** mean **modify** or **modulo**?

What about **acc**, **pos** or **auth**?

Sadly, **fab** was just a function $f : A \rightarrow B$
(not **fabulous**)

Allow one exception: **id** for ‘identity’

One letter is too short

Local variable: what is the meaning?

```
var a = 42;
```

The exception that proves the rule?

```
for (int i = 1; i < 42; ++i)
```

Not an improvement:

```
ii, jj, kk
```

Functional programming: one letter is *still* too short

```
def modp[C]
```

```
  (f: B1 ⇒ (B2, C), a: A1):
```

```
  (A2, C) = {
```

```
    val (b, c) = f(get(a))
```

```
    (set(a, b), c)
```

```
  }
```



()


```

func 🙄😭🎃🐍(🐎👻📀📺: [[String]]) -> Void {
    var 🐳🐱🎒🇺🇸🎵 = [Int](count: 🐎👻📀📺.count, repeatedValue: 0)
    var ✨ = 0
    🚑🐎(🐎👻📀📺, 🐳🐱🎒🇺🇸🎵)
    while true {
        if ✨ >= 🐎👻📀📺.count {
            return
        } else if 🐳🐱🎒🇺🇸🎵[✨] < 🐎👻📀📺[✨].count - 1 {
            🐜👾🐧🐵(&🐳🐱🎒🇺🇸🎵, ✨)
            🐳🐱🎒🇺🇸🎵[✨] += 1
            ✨ = 0
        } else {
            ✨ += 1
            continue
        }
        🚑🐎(🐎👻📀📺, 🐳🐱🎒🇺🇸🎵)
    }
}

```

```

func 🚑🐎(🐎👻📀📺: [[String]], 🐳🐱🎒🇺🇸🎵: [Int]) -> Void {
    for (🐣, 🚑🐎) in enumerate(🐎👻📀📺) {

```


Multiple words can be replaced by
more specific words

What's an **appointment_list**?

a **calendar**

What's an **company_person**?

an **employee** or perhaps an **owner**

What's a **text_correction_by_editor**?

just an **edit**

Vague words are vague

Alan Green wrote* about vague words, e.g.

InvoiceManager

TaskManager

‘Manager’ is very imprecise; one of its meanings may be the word you want:

Bucket, Supervisor, Planner, Builder

* http://www.bright-green.com/blog/2003_02_25/naming_java_classes_without_a.html

Vague words are vague

get at the start of a method name is appropriate only for returning a field value.

If it does anything else, or gets the data from anywhere else, use another name:
fetch, find, lookup, create, calculate, derive, concoct,

Wrong words are wrong,
Synonyms are confusing

order \neq shipment

carrier \neq broker

shipment \neq transport leg

shipment = consignment

carrier = transporter

transport leg = journey

Apache Camel - <http://camel.apache.org>

**(Java enterprise
middleware example)**



```
// Not enough jokes in code
```

```
/** Configure and start Apache Camel. */
```

```
def mountCamel() {
```

```
    Logger.info("Starting Camel...")
```

```
    val context = new DefaultCamelContext()
```

```
    configuredRoutes foreach { route =>
```

```
        context.addRoutes(route)
```

```
    }
```

```
    context.start()
```

```
}
```



Programming Puns

Writing code, comments, variable names, etc., in the manner that they create a pun.

This is a common and fun practice, but it can be an [AntiPattern](#). Making a pun can detract from the readability of the code. (See [MeaningfulNames](#).) Plus, if you work with the code long enough, some jokes can tire to the point of becoming completely annoying.

Alternately, simply a class of puns which require programming knowledge to be understood.

Examples (of the first kind)

```
byte me;  
long john_silver;  
char broiled;  
string me_along;  
float valve;  
double jeopardy;  
struct by_lightning { ... };  
Object strongly;  
class warfare { ... };  
String cheese;  
Exception taken;  
Graphics ex: // XXX
```


Property accessors revisited

In a numeric library, these method names would be irresistible, but inadvisable:

getEven

getReal

getAround

getRoundTo

getRichQuick

getJoke

Summary of naming things badly

Meaningless: **foo**

Too general: **data**

Too short: **a**

Too long: **text_correction_by_editor**

Abbreviated: **acc**

Vague: **InvoiceManager**

Wrong: **order**

Just not funny: **startCame1**

Better naming

How to solve the naming problem

Become a better writer.

Improve your vocabulary.

Adopt better naming practices.

Work on it.

Become a better writer

Naming is just one part of writing,
and is mostly about vocabulary.

You may remember working on vocabulary
as part of learning a foreign language.

Not having had to learn a foreign language
is a mixed blessing.

Improve your general vocabulary

Read books, especially funny novels.

Play word games with someone who always wins, until they don't.





Peter H.
199

Karen V.
347



Karen played **JOUST** for 44 points

			D ²	E ¹	N ²	TL		R ¹	I ¹	N ²	G ³				
		DL		R ¹	O ¹	C ⁴	K ⁵	E ¹	T ¹			DL			
		DL			DL			T ¹			DL			C	
TW			TL					DW	E ¹			V ⁵	R ¹	O ¹	W ⁴
		DL				DL		A ¹				DL		L ²	
			J ¹⁰	O ¹	U ²	S ¹	T ¹		M ⁴	TL				L ²	
TL					P ⁴			I ¹	S ¹		DL			I ¹	D ²
			S ¹	E ¹	V ⁵	E ¹	N ²					DW		D ²	A ¹
TL					W ⁴			P ⁴	A ¹	N ²	F ⁴	R ¹	I ¹	E ¹	D ²
		DW				TL	A ¹	H ³		TL					R ¹
		DL					Z ¹⁰	A ¹	X ⁸				DL		
TW			TL				O ¹	S ¹				TL			TW
		DL				DL			E ¹			DL			DL
		DL					DW					DW			DL
			TW			TL		TL				TW			

T ¹	Q ¹⁰	R ¹	H ³	I ¹	I ¹	A ¹
----------------	-----------------	----------------	----------------	----------------	----------------	----------------

Improve your general vocabulary

Use your dictionary and thesaurus...

sure

surely

surer

surest

sureties

surety

sure

adjective

- I am sure that they did not have an affair:* CERTAIN, positive, convinced, definite, confident, decided, assured, secure, satisfied, persuaded, easy in one's mind, free from doubt; unhesitating, unwavering, unfaltering, unvacillating, unshakeable, unshaken. ANTONYMS unsure, uncertain, doubtful.
- he was **sure of** finding a way around the difficulties:* CONFIDENT, certain, assured; with no doubts about.
- someone was sure to cop it before the day was out:* BOUND, destined, fated, predestined, very likely. ANTONYMS unlikely.
- this is a very attractive way of presenting fruit and is a sure winner with the children:* GUARANTEED, unfailing, infallible, unerring, assured, certain, inevitable, incontestable, irrevocable; informal sure-fire, in the bag, as sure as eggs is eggs.
- he could have thrown his servant into the street in the sure knowledge that it would be put down to robust good humour:* UNQUESTIONABLE, indisputable, incontestable, irrefutable, incontrovertible, undeniable, indubitable, beyond question, beyond doubt; undoubted, absolute, categorical, true, certain, well grounded, well

**A sandwich walks
into a pub.**

**The barmman says,
'I'm sorry, we
don't serve food.'**

Tell jokes

Many jokes rely on word-play.

It takes practice to think of puns quickly.

Puns are important for naming, because they rely on double-meanings.

Spotting double-meanings is the essential skill for avoiding ambiguous names.

Adopt better naming practices

Start with *meaning* and *intention*.

Use words with precise meanings.

Prefer fewer words in names.

No abbreviations in names, except `id`.

Use code review to improve names.

Remember: 'rename' is the simplest but most effective refactoring. Use it.

Replace vague words with more specific synonyms

Manager

do

Object

execute

Data

perform

Thing

operate

Info

manage

Amount

handle

Details

get

Overcome fear of renaming

The only thing harder than naming is *renaming*.

Renaming requires change, a conversation, and new understanding.

‘Refactor’ is the safest refactoring.

Copyrighted Material

Microsoft

CODE COMPLETE

2

Second Edition



A practical handbook of software construction

Steve McConnell

Two-time winner of the *Software Development Magazine* Jolt Award

Copyrighted Material

Chapter 10: The power of variable names

Clean Code

A Handbook of Agile Software Craftsmanship



Chapter 2: Meaningful Names

O'REILLY®



Becoming a Better Programmer

A HANDBOOK FOR PEOPLE WHO CARE ABOUT CODE

Pete Goodliffe

Chapter 2: Names

Gather domain-specific vocabulary

Scan the domain model entities' Wikipedia pages for names of related concepts.

Read novels set in your customer's domain, to learn their jargon.

Find out what they *really* mean.

Domain-Driven

DESIGN

Tackling Complexity in the Heart of Software



Eric Evans

Foreword by Martin Fowler

Chapter 2:
Communication
and the use of
language



PROGRAMMERS

QUESTIONS TAGS USERS BADGES

TAGS

A tag is a keyword or label that categorizes your question with other, similar questions. Using the tag you can find other questions and answers your question.

Type to find tags:

× 338

× 45

Give meaning and explanation with the fewest number of characters in a form that is most accepted by your team or

21 asked this year

5 asked this month, 67 this year

tag synonyms

Are there good techniques or tests for naming types?



20



4

An awkward, open question, but it's a problem I'm always bumping against:

Software that's easy to maintain and work with is software designed well. Trying to make a design intuitive means naming your components in such a way that the next developer should be able to infer the function of the component. This is why we don't name our classes "Type1", "Type2", etc.

When you're modelling a real-world concept (e.g. customer) this is generally as simple as naming your type after the real-world concept being modelled. But when you're building abstract things, which are more system-oriented, it's very easy to run out of names which are both easy to read and simple to digest.

Are there any good techniques for choosing a well-meaning name for a component, or how to build a family of components without getting muddled names?

Are there any simple tests that I can apply to a name to get a better feel for whether the name is "good", and should be more intuitive to others?

design

naming

share edit flag

asked Jan 13 '12 at 13:20



Tragedian

403 2 11

For naming, there are **six techniques** that were proven to work for me:

1. spend a lot of time on inventing names
2. use code reviews
3. don't hesitate to rename
4. spend a lot of time on inventing names
5. use code reviews
6. don't hesitate to rename

Additional benefits

**If you become a better writer,
you could use your new skills**

... for writing



Writing whole
sentences in code

‘Most of the things programmers say about comments in code are excuses for not writing any comments at all.’

@PeterHilton

Comments: the basics

1. Don't say what the code does
(because the code already says that)
2. Don't explain awkward logic
(improve the code to make it clear)
3. Don't add too many comments
(it's messy and they'll get out of date)

Explain why the code exists

Even perfect code cannot explain its own existence.

When should I use this code?

When *shouldn't* I use it?

What are the alternatives to this code?

Discover which comments are hard to write, and why

If a comment is easy to write, then that code doesn't need a comment.

Write a one-sentence comment, for every class and method, to start with.

‘A common fallacy is to assume authors of incomprehensible code will somehow be able to express themselves lucidly and clearly in comments.’

@KevlinHenney

**Acknowledge that writing
(comments) is a specialist skill**

**On a cross-functional development team,
not everyone is good at visual design.**

The same goes for writing about code.

Work out who is a better writer.

Get help with writing comments.

How to write good comments (summary)

1. Try to write good code first.
2. Try to write a one-sentence comment.
3. Refactor the code until the comment is easy to write.
4. Now write a good comment.
5. Don't forget the rules of good writing (e.g. remove unnecessary comments).



Summary

Summary

1. Naming is hard
2. Get inspiration from great writers
3. Read novels, tell jokes, play games
4. Expand your vocabulary
5. Try actual writing;
start with comments,
try blogging, or even write a book

@PeterHilton

<http://hilton.org.uk/>