# 2nd hardest thing in computer science

# Paweł Lewtak

Developer @ GOG.com

# Definition?

*"There are only two hard things in Computer Science: cache invalidation and naming things."*

Phil Karlton

Source: https://martinfowler.com/bliki/TwoHardThings.html

> *"There are 2 hard problems in computer science: cache invalidation, naming things, and off-by-1 errors"*
>
> Leon Bambrick

Source: https://twitter.com/secretgeek/status/7269997868

# #2 Naming things*

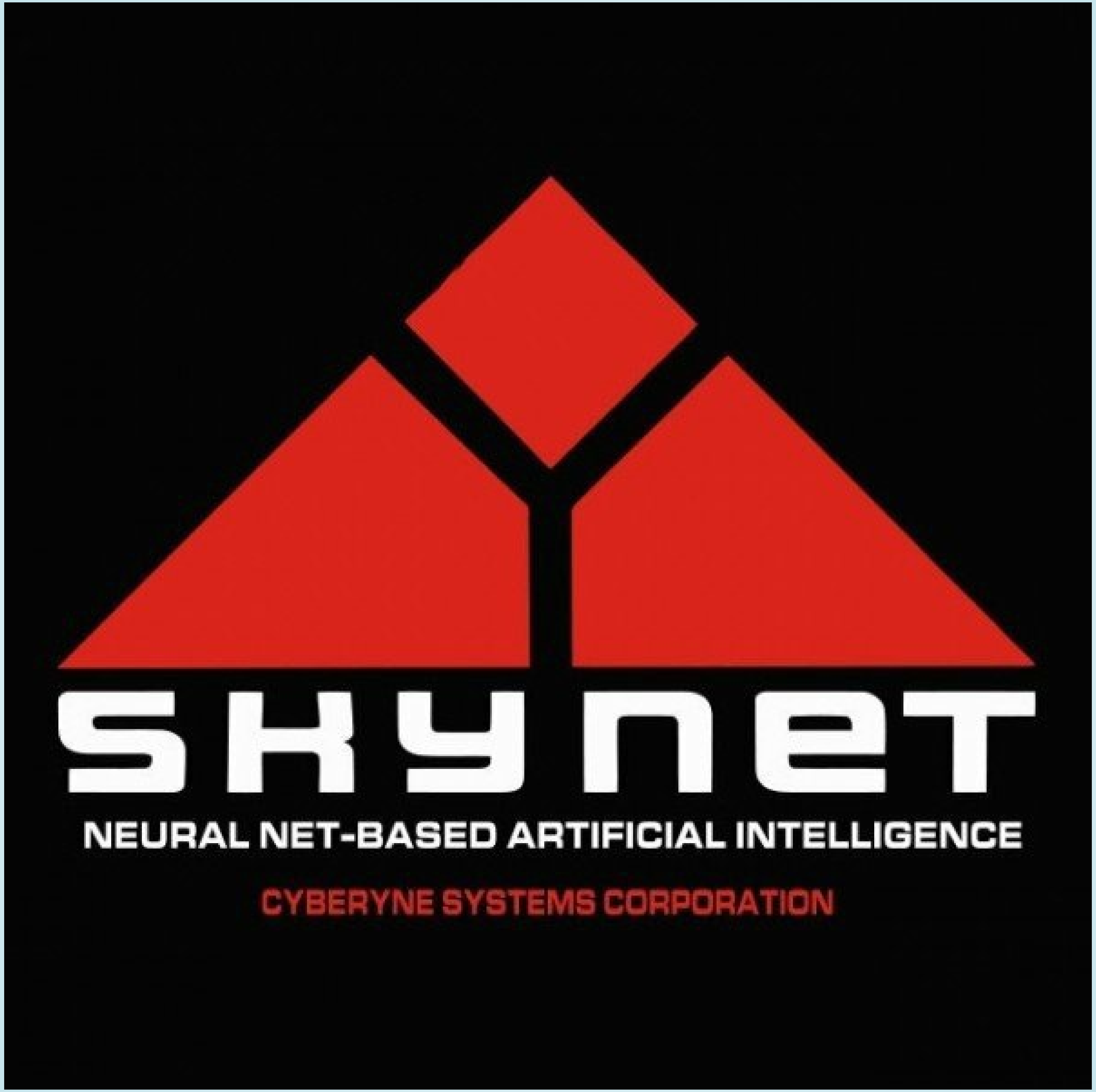
# *things

- variables
- methods
- classes
- modules
- comments
- inline docs
- commit messages

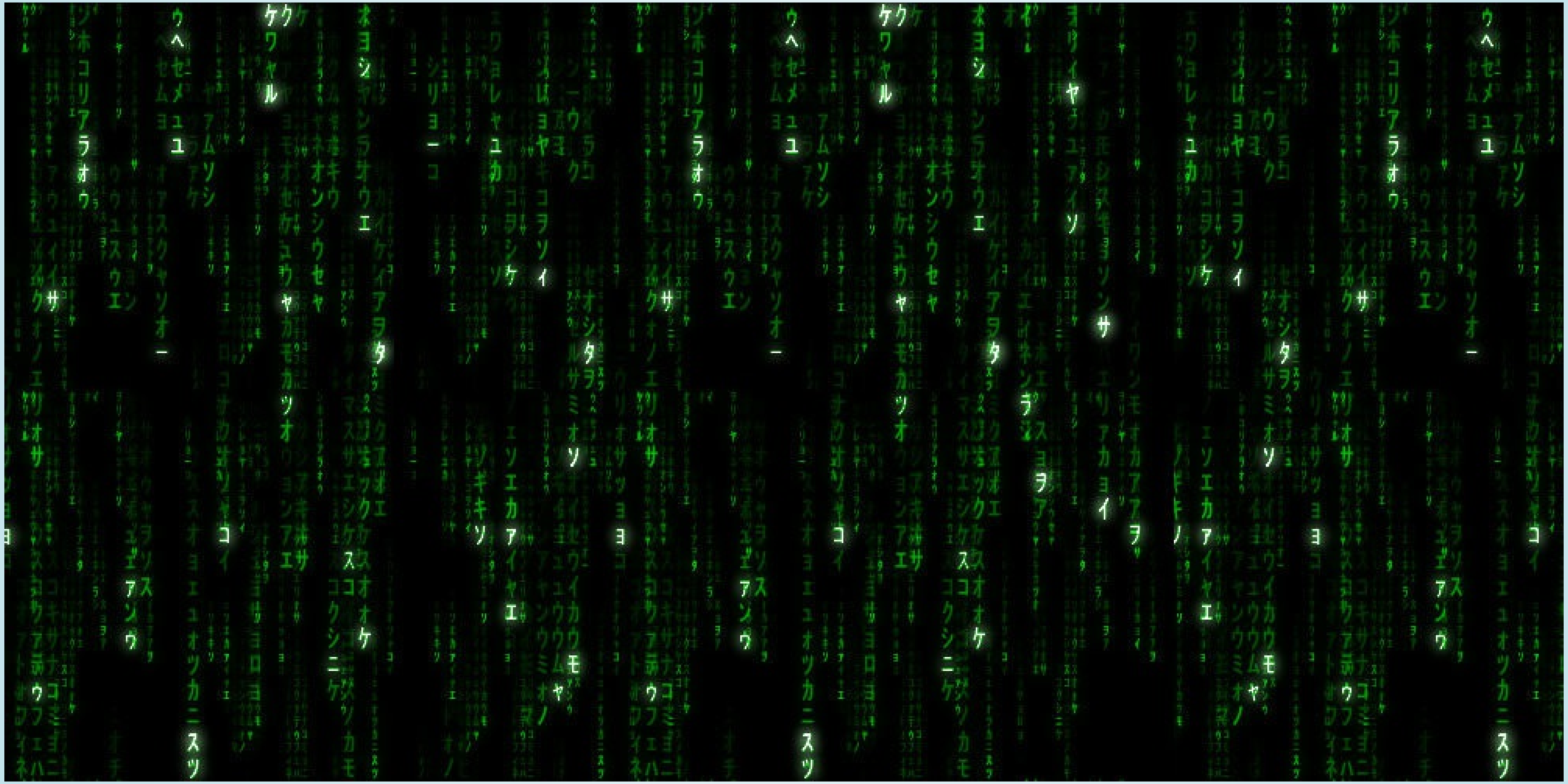
# You don't code for CPU

# You don't code for interpreter

You don't code for compiler



SKYNET
NEURAL NET-BASED ARTIFICIAL INTELLIGENCE
CYBERYNE SYSTEMS CORPORATION



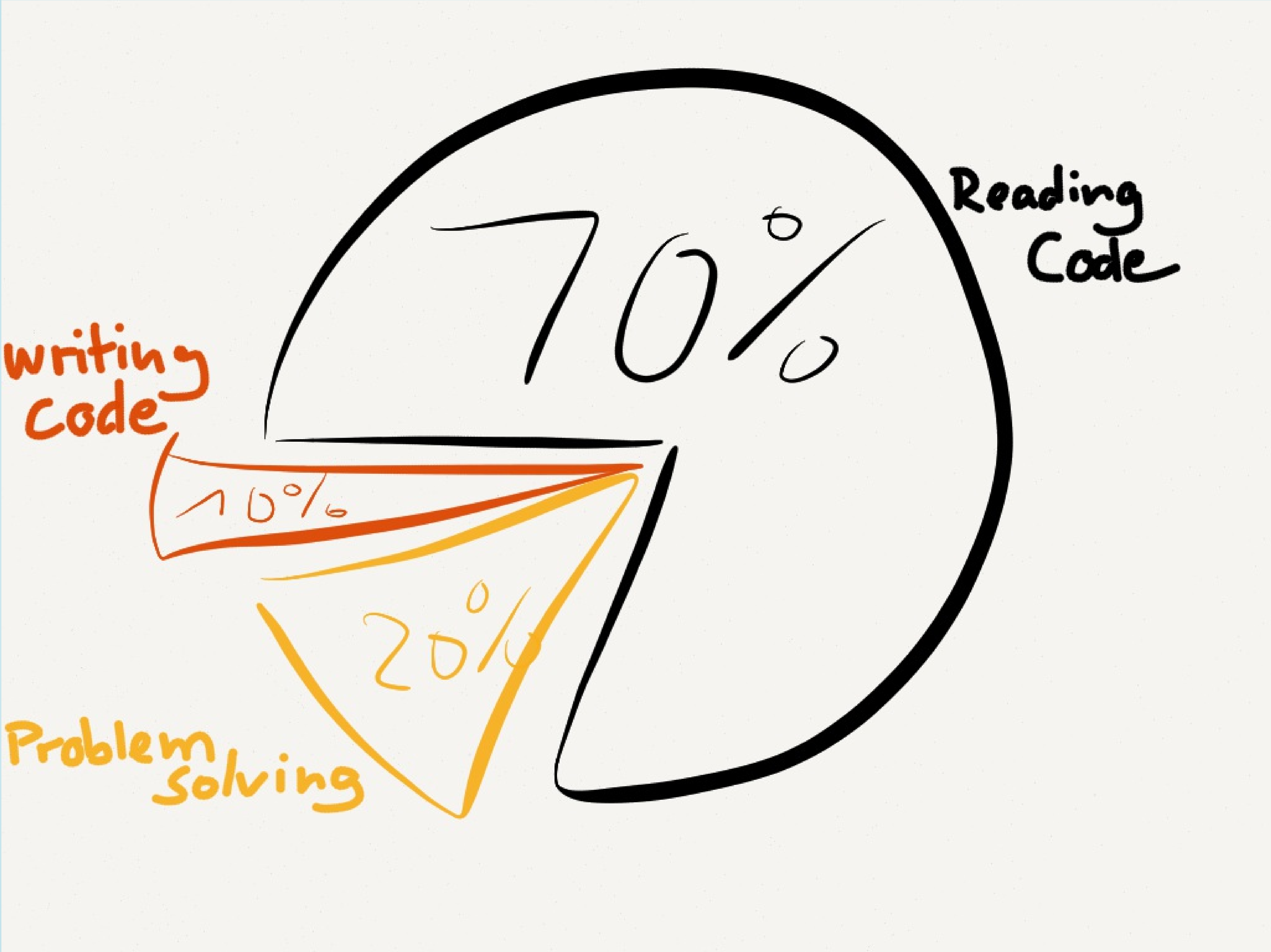
# You code for people

You code for other developers

# You code for your future self

# Don't code, wrote prose

Source: https://www.youtube.com/watch?v=CKONKZLmMwk



Source: https://trustartist.com/2015/01/27/pair-programming-economics/

*"Instead of imagining that our main task is to instruct a* computer *what to do, let us concentrate rather on explaining to* human beings *what we want a computer to do."*

Donald E. Knuth

Source: Literate Programming, 1983

# Comprehension

~70%

```php
function a($b) {
    sort($b);
    $c = count($b);
    if ($c % 2 === 1) {
        return $b[($c - 1) / 2];
    } else {
        return ($b[$c/2 - 1] + $b[$c/2]) / 2;
    }
}
```

```
function median($pool) {
    sort($pool);
    $size = count($pool);
    if ($size % 2 === 1) {
        return $pool[($size - 1] / 2];
    } else {
        return ($pool[$size/2 - 1] + $pool[$size/2]) / 2;
    }
}
```

# Self-documenting code

# Programming is mapping

- from problem domain
- via intermediate domain
- into programming domain

# DDD FTW

# Worst variable name

# data

Source: http://archive.oreilly.com/pub/post/the_worlds_two_worst_variable.html

Thank You,
Captain
Obvious!

# Second worst name?

# data2

```php
$total = $price * $qty;
$total2 = $total - $discount;
$total2 += $total * $taxrate;

$total3 = $purchase_order_value + $available_credit;
if ($total2 < $total3) {
    echo "You can't afford this order.";
}
```

```
$order_total = $price * $qty
$payable_total = $order_total - $discount
$payable_total += $payable_total * $taxrate

$available_funds = $purchase_order_value + $availble_credit
if ($payable_total < $available_funds) {
    echo "You can't afford this order.";
}
```

> *"No-one sets out to write legacy code"*
>
> Rachel Willmer

Source: https://twitter.com/amokleben/status/868377283496751104?s=09

# Broken window theory

# Code will decay

# Design patterns

> *Misapplied Java design patterns are the root of all AbstractWordFactoryFactory("evil")*
>
> HN comment

Source: https://twitter.com/tmmx/status/865308678903267328

# Naming conventions

# TL;DR

- CamelCaseClass
- methodName
- someVariable
- CAPITAL_CONSTANT

syntax < semanthics

# Common issues

# Pseudo getter

# get_data()

with extra operations inside

# get_create_object()

- fetch
- find
- lookup
- create
- calculate

# Not really a boolean

# is_active()

```
function is_active() {
    if (cond) {
        return 'false';
    }

    return 'true';
}
```

# is_valid()

```
function is_valid() {
    if (input_is_valid) {
        return true;
    }
}
```

# Plural / singular names

```
function get_person() {
    return ['John Doe', 'Jane Doe'];
}

function get_employers() {
    return 'John Doe';
}
```

# Misleading docs

```
function get_lowest_price($user) {}
```

```
/**
 * Actually it returns the highest price.
 */
function get_lowest_price($user) {}
```

```php
/**
 * Actually it returns the highest price.
 */
function get_lowest_price($user) {
    if ($user === null) {
        return _get_highest_price();
    } else {
        return _get_lowest_price($user);
    }
}
```

# More than one responsibility

# Abbreviations

```
pos
mod
abs
auth
```

# Synonyms

# <ThatThing>Manager

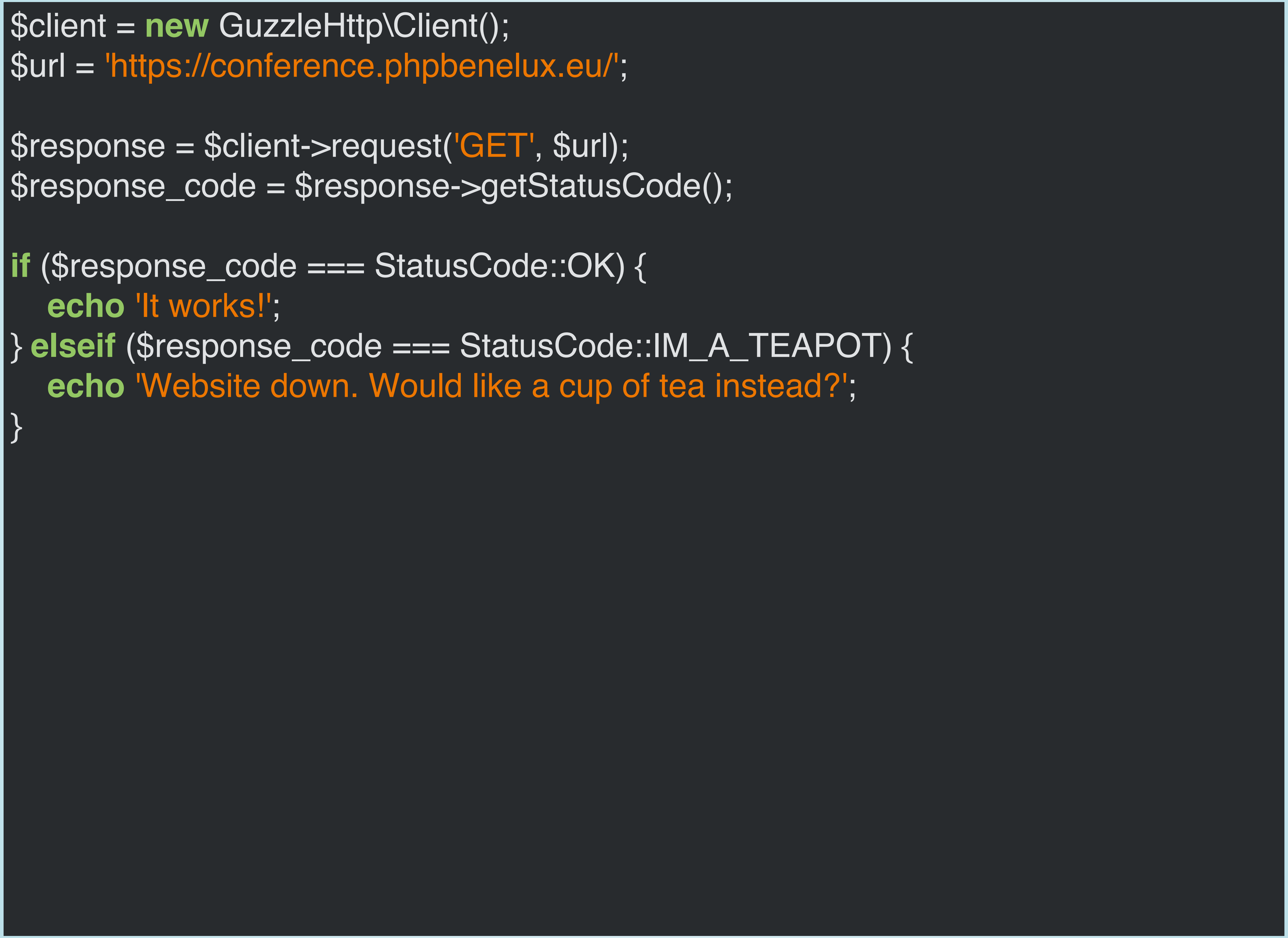
- UserManager
- StringManager
- ProductManager
- etc.

# Alternatives

- Builder
- Writer
- Adapter
- Factory
- Handler
- Provider
- Converter

# Magic numbers

```php
$client = new GuzzleHttp\Client();
$url = 'https://conference.phpbenelux.eu/';

$response = $client->request('GET', $url);
$response_code = $response->getStatusCode();

if ($response_code === 200) {
    echo 'It works!';
} elseif ($response_code === 418) {
    echo 'What is that?!';
}
```



```php
$client = new GuzzleHttp\Client();
$url = 'https://conference.phpbenelux.eu/';

$response = $client->request('GET', $url);
$response_code = $response->getStatusCode();

if ($response_code === StatusCode::OK) {
    echo 'It works!';
} elseif ($response_code === StatusCode::IM_A_TEAPOT) {
    echo 'Website down. Would like a cup of tea instead?';
}
```

# Useless comments

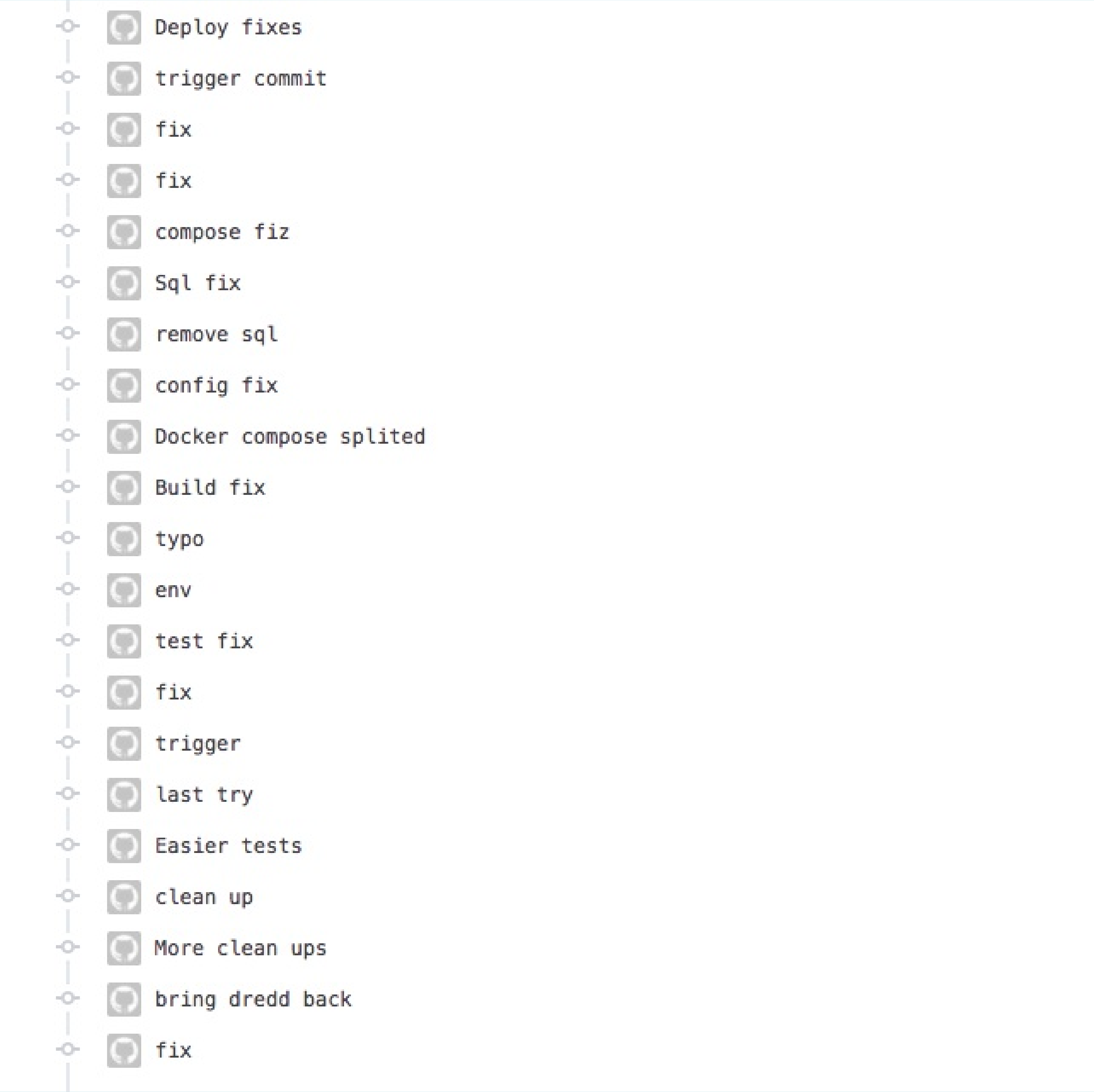
```
/**
 * Returns the data.
 */
function get_data() {}

/**
 * Return maximum ID value from the database.
 */
function get_max_id_from_db() {}
```

Explain why,
not what or how

*"Code should have comments, but if your file is more than 25% comments, that's a red flag: you may be explaining bad code"*

Adam Culp

# Commit messages



```
Deploy fixes
trigger commit
fix
fix
compose fiz
Sql fix
remove sql
config fix
Docker compose splited
Build fix
typo
env
test fix
fix
trigger
last try
Easier tests
clean up
More clean ups
bring dredd back
fix
```

# Bad name:

- Does more that what is says
- Says more than what it does
- Does the opposite
- Contains more than what it says
- Says more than what it contains
- Contains the opposite

# Good practices

# Specific names

## No generics

# Short names

# Do not use negation

```
is_not_enabled()
```

is_disabled()

# Consistent names

# Code & docs

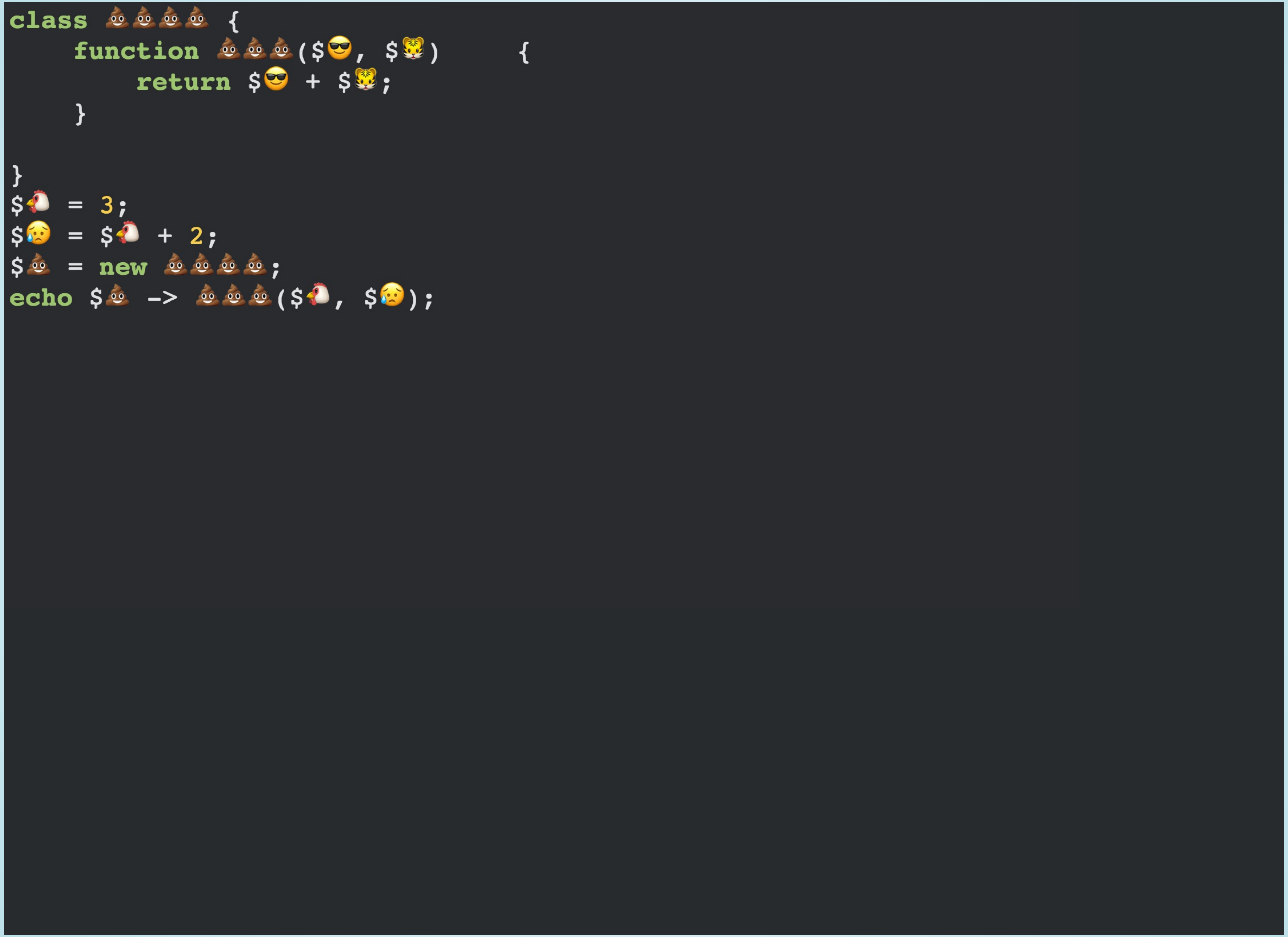
# Single responsibility

# Domain terms

# Think about it

# ASCII only

```
$Δ = 1;
++$Δ;
echo $Δ;
```



```
class 💩💩💩💩 {
    function 💩💩💩($😎, $🐯)     {
        return $😎 + $🐯;
    }

}
$🐔 = 3;
$😥 = $🐔 + 2;
$💩 = new 💩💩💩💩;
echo $💩 -> 💩💩💩($🐔, $😥);
```

# Hungarian notation

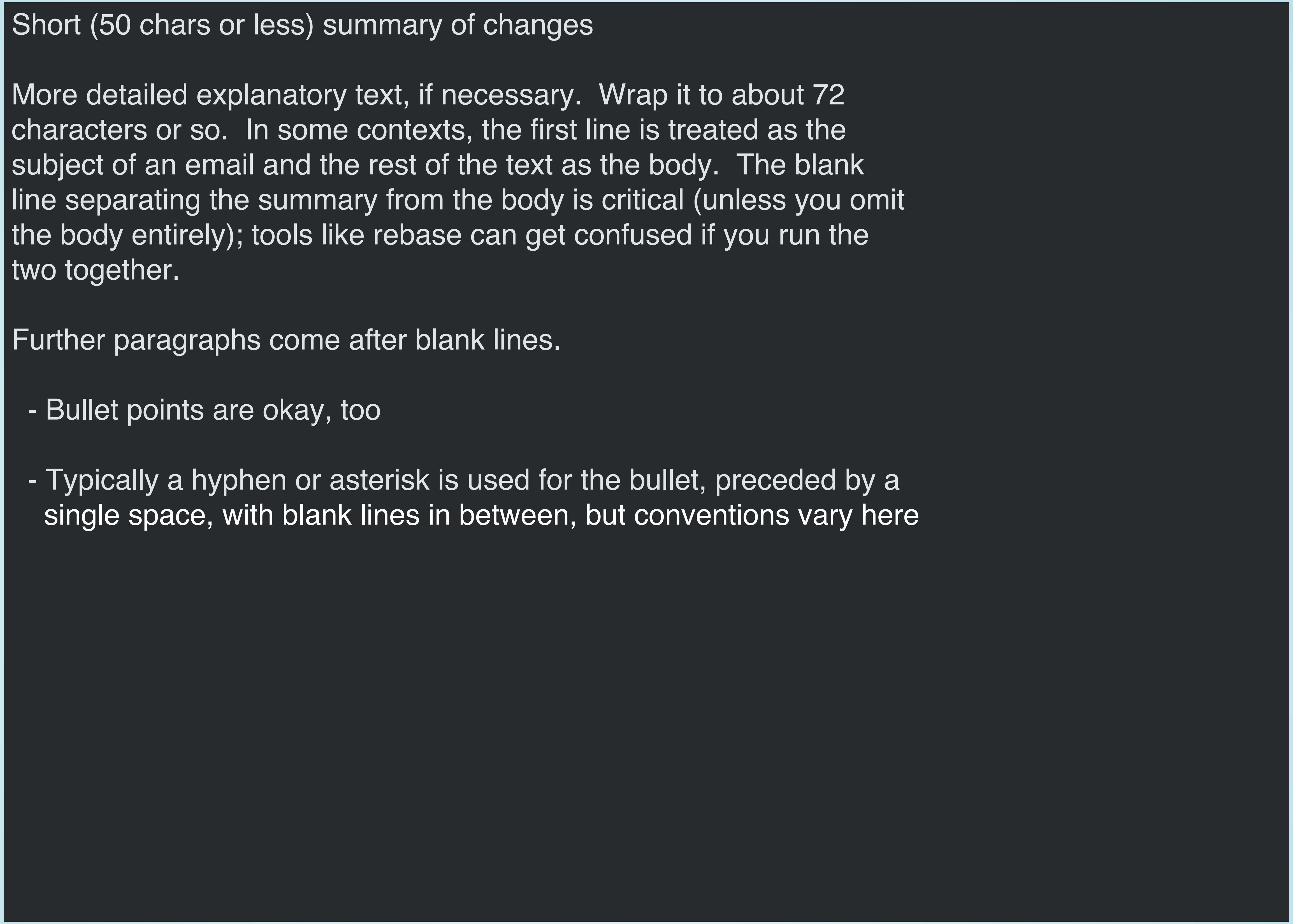
```
hostList, hostSet => hosts, validHosts
valueString => firstName, lowercasedSKU
intNumber => accountNumber
```

# Tests!

# Commit message

# Good commit message

- Speeds up review process
- Helps write release notes
- Helps future maintainers



```
Short (50 chars or less) summary of changes

More detailed explanatory text, if necessary.  Wrap it to about 72
characters or so.  In some contexts, the first line is treated as the
subject of an email and the rest of the text as the body.  The blank
line separating the summary from the body is critical (unless you omit
the body entirely); tools like rebase can get confused if you run the
two together.

Further paragraphs come after blank lines.

  - Bullet points are okay, too

  - Typically a hyphen or asterisk is used for the bullet, preceded by a
    single space, with blank lines in between, but conventions vary here
```

Source: Source: http://git-scm.com/book/ch5-2.html

# How?

# Agree on standards

# Boy Scout Rule

# Practice

# Improve vocabulary

# Refactor

# Code reviews

Short, bite size, single logical change

# Code ownership

# Commit messages

# Research papers

- https://www.cqse.eu/publications/2005-concise-and-consistent-naming.pdf
- http://www.cs.loyola.edu/~lawrie/papers/lawrieJese07.pdf
- https://www.researchgate.net/publication/224079441_Relating_Identifier_Naming_Flaws_and_Code_Quality_An_Empirical_Study
- http://www.veneraarnaoudova.com/wp-content/uploads/2014/10/2014-EMSE-Arnaodova-et-al-Perception-LAs.pdf

# Thank you!

# Questions?

# Feedback?

https://joind.in/talk/76157

@pawel_lewtak