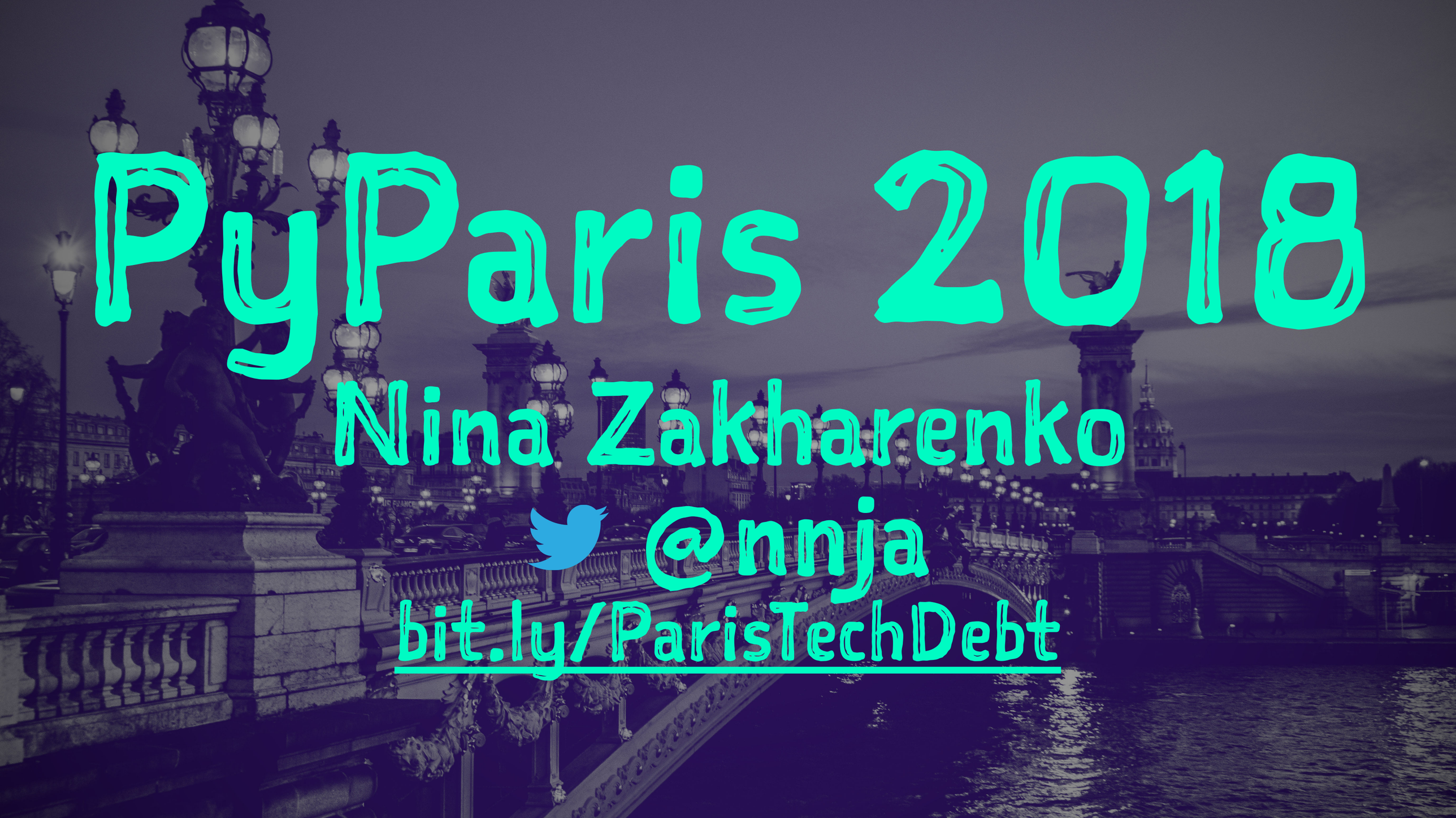

PyParis 2018
Nina Zakharenko
@nnja
bit.ly/ParisTechDebt

Livetweet!

use #PyParis

@nnja

Technical Debt

The code monster in your closet

slides: bit.ly/ParisTechDebt

@nnja

What is technical debt?

@nnja

A series of bad decisions

(Both business & technical)

@nnja

Which lead to ->

Error prone code & architecture

@nnja

... and using more

Resources

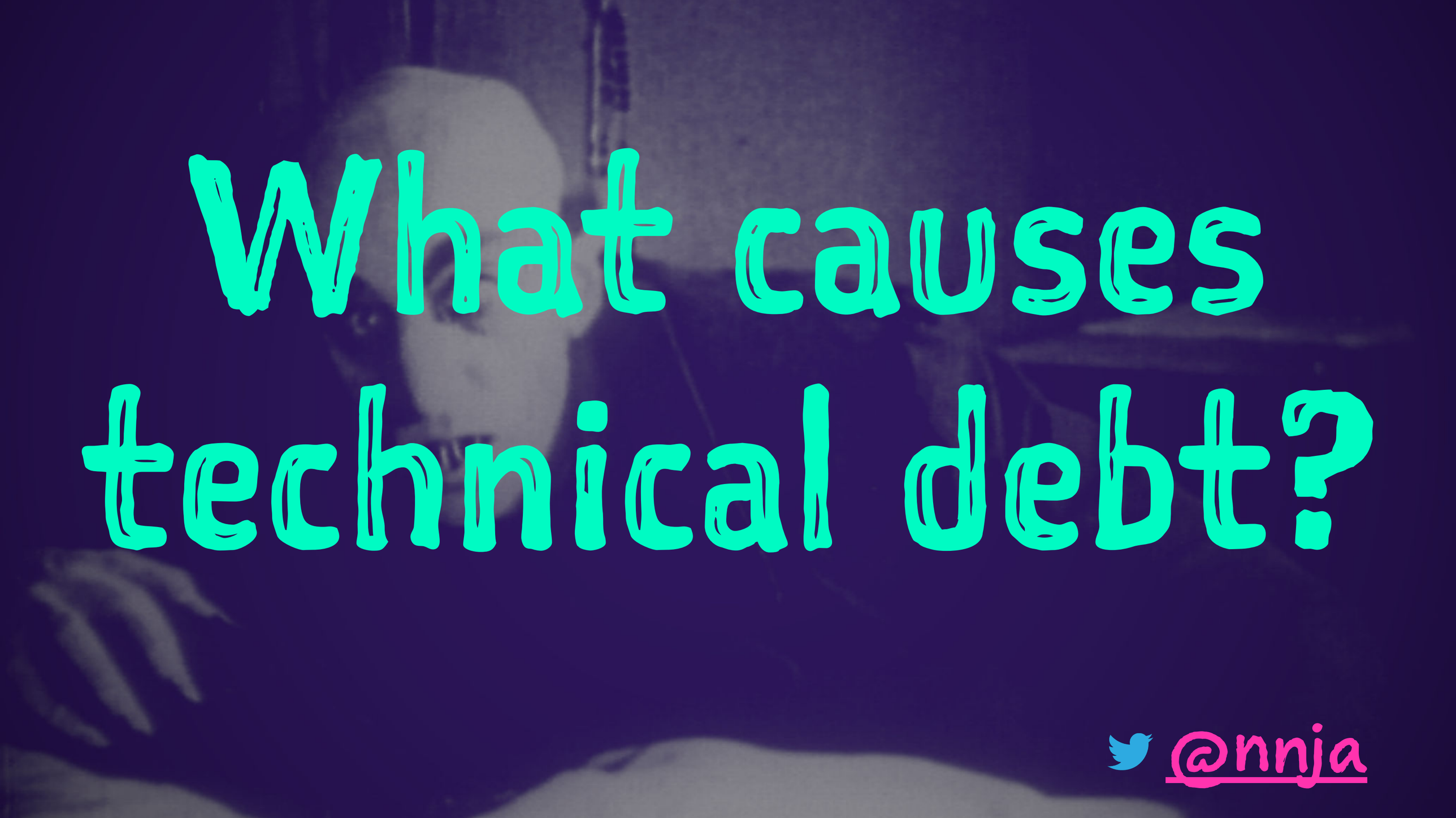
to accomplish

Less

@nnja

What decisions were made in the past that prevent me from getting sh** done today?

@nnja


What causes technical debt?
@nnja

Me.
And you.

@nnja

Mistakes I Made Early On

- Not seeing the value in unit tests
- Not knowing how to say **NO** to features

@nnja

Mistakes I Made Early On

- Overly optimistic estimates
- Putting releases over good design & reusable code

@nnja

Time Crunch

That project was due yesterday!

I'll take a shortcut, and clean up the mess tomorrow.

@nnja

Unneeded Complexity

Lines of code committed != amount of work accomplished

@nnja

Lack of understanding

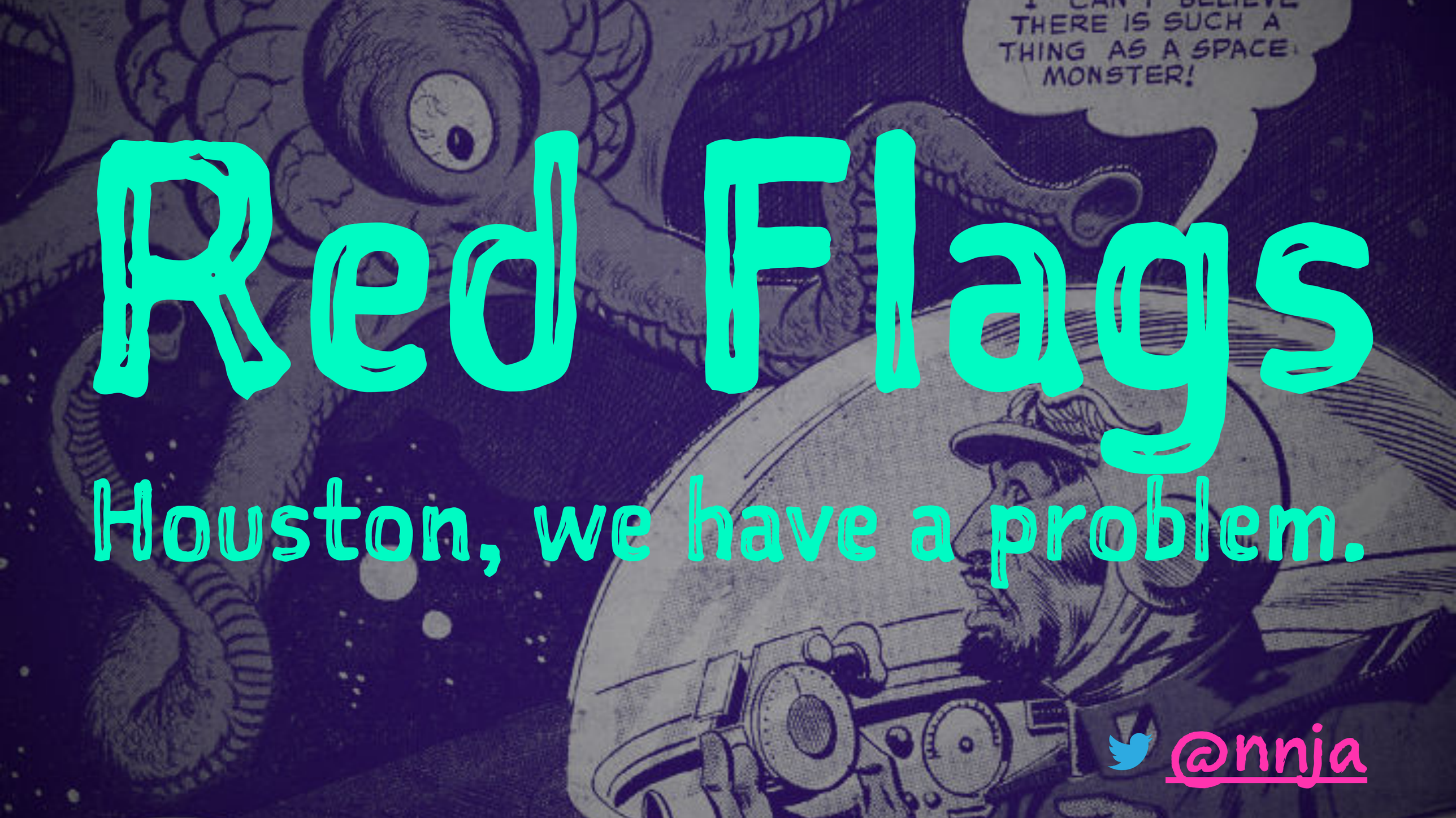
1. Have a problem
2. Look up a solution on stackoverflow
3. Copy & paste it into your code
4. ???
5. **Bugs!**

@nnja

Culture of Despair

This is already a heap of trash.

Will anyone really notice if I add one more thing to the top?



Red Flags

Houston, we have a problem.

@nnja

Code Smells

- Not Bugs
- An indication of a deeper problem

@nnja

Code Smells

- Half implemented features
- No documentation, or poor documentation

@nnja

Code Smells

- Commented out code
- Incorrect comments
- No tests, or worse: broken tests

@nnja

Restore deleted code with git!

Find by content:

```
$ git log --summary -G'(D|d)jango'
```

Find the commit that deleted a file:

```shell
git log --diff-filter=D --summary -- <filename>
```

@nnja

No more commented out code! 

@nnja

Poor Documentation

```
class OrganicGlutenFreePizzaFactory:
    def get_dough(self):
        """
        Return amazing, organic, GMO and Gluten Free Dough
        """
        # ran out of organic gluten free, use the other stuff.
        # return 'organic gluten free dough'
        return 'gmo pesticide processed gluten-full dough'
```

@nnja

Architecture & Design... Smells

- Parts of the code no one wants to touch
- Brittle codebase -- changing code in one area breaks other parts of the system
- Severe outages caused by frequent & unexpected bugs

@nnja

Good Design -> Implementing new features comes easily

Poor Design -> New features are shoe-horned into the system

@nnja


Python Specific
@nnja

Functionality changes, but variable names don't

```
employees = ['John', 'Mary', 'Dale']

employees = 'Bob'

employees[0]
```

@nnja

Monkey Patching 🚫🐒

```
def new_init(self):
    pass

some_library.SomeClass.__init__ = new_init
```

@nnja

What exactly does this decorator do?

```
def decorator_evil(func):
    return False

@decorator_evil
def target(a,b):
    return a + b

>>> target(1,2)
TypeError: 'bool' object is not callable

>>> target
False
```

Circular Dependencies

```
# Circumvent circular dependency warnings
def some_function(x):
    from some.module import some_method
    some_method(x)
```

@nnja

Fig. 22.—Vampire Bat.

Case Studies

@nnja


IRS Chief:
"We still have applications that were running when JFK was President"
Tech at the IRS

50 Year Old Technology

"And we continue to use the COBOL programming language, it is extremely difficult to find IT experts who are versed in this language."

@nnja

It's not just the IRS

- Banks & Financial Institutions
- Universities
- Air Traffic Control
- ... many still use COBOL

@nnja

Story Time

- I used to work in finance.
- At the time I was there, all of the banking systems were run on mainframes.
- The bankers were getting frustrated. They wanted a UI.

@nnja

Big Idea!

- Let's write a fancy new web front end
- It'll do *ALL* the things

@nnja

But

- Rewriting the backend is too expensive
- It already does what we need
- Let's leave the mainframe as the backend

@nnja

Cursors

- The mainframe would output a text screen from a program result, based on a query.
- The results would be parsed by reading variables from the screen in certain positions.

@nnja

Result?

- The new system was incredibly slow
- And error prone
- After months of work, the multi-million dollar rewrite was scrapped

@nnja

You can try to cover up debt...
(but it probably won't work)

 @nnja

The MVP

- (Minimum Viable Product)
- Get the product to market as soon as possible

@nnja

A Great Idea

- A successful project that was created by a lone developer in a coffee fueled 48 hours.

 @nnja

There Was a Problem

- Years went on, but the initial code and design didn't go away.
- Instead, it became the base for an expanding project, with expanding features.
- There was never any time to refactor.

@nnja

@nnja

Scope Creep

- Features that someone thought was a good idea one day, stuck around forever.
- > "In case we need them. Later."

@nnja

Sad Developers

- Minimal working tests (no time to write them).
- When a release was pushed, something was bound to break.
- Made everything feel like it was *your* fault.

 @nnja

Grinding To a Halt

- Development time for new features skyrocketed
- The project was deemed too difficult to maintain
- ... and cancelled.

@nnja


Sometimes you need to
burn it.
With fire.
@nnja


Battling The Monster
@nnja

Don't point fingers

Technical debt is a team-wide problem.

Everybody needs to be part of the solution.

Work Together

- Code Standards
- Pair Programming
- Code Reviews

@nnja

Unless something is on fire,
or you're losing money,
don't merge unreviewed
code into master.

 @nnja

Be Accountable

- Unit & Integration Tests
- Pre-Commit Hooks
- Continuous Integration

@nnja

Make a Commitment

Company tried to fight debt, but they didn't make a commitment.

@nnja

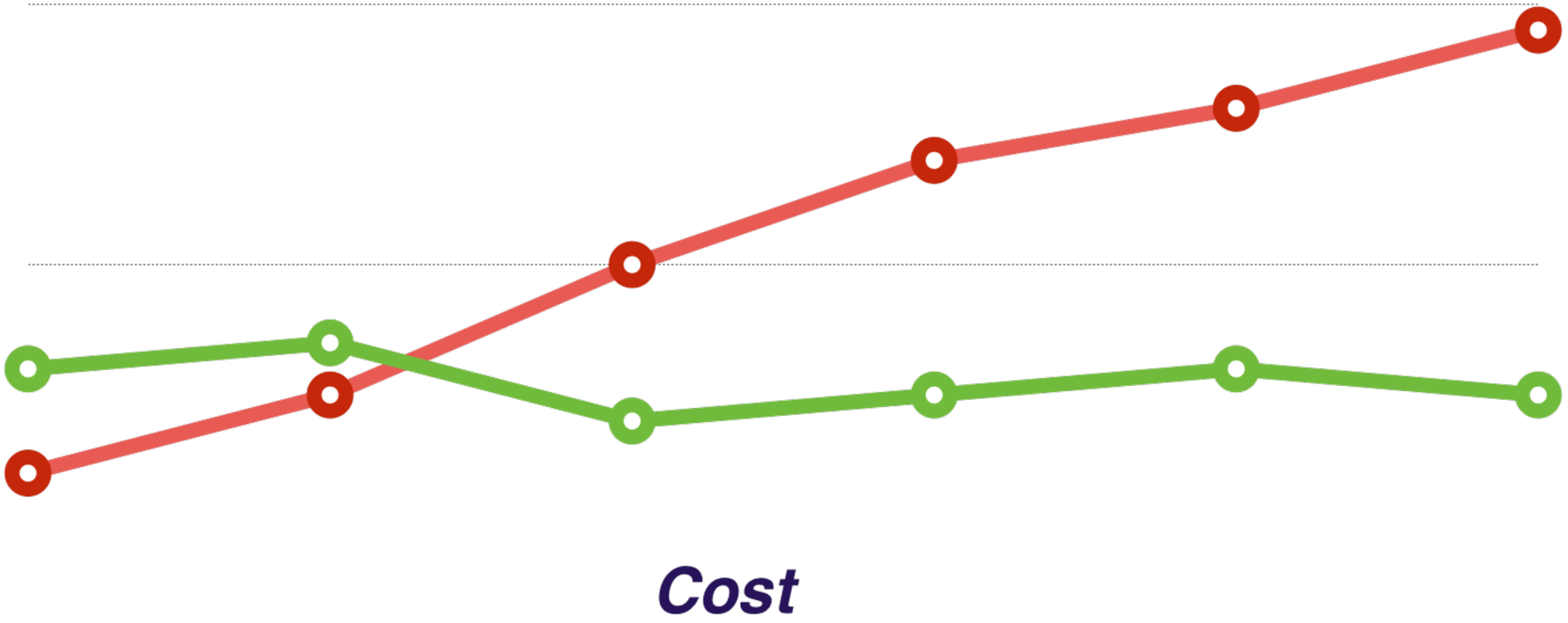
Ended up with twice as many technologies in their stack as needed, and twice as big of a mess.

Sell It To Decision Makers

By allocating project time to tackling debt, the end result will be less error prone, easier to maintain, and easier to add features to.

@nnja

Not broken, why fix it?



Source

Ski Rental Problem

You're going skiing for an unknown number of days.

It costs $1 a day to rent, or $20 to buy.

Source

Hiring developers is hard.

Technical debt frustrates developers.

Frustrated developers are more likely to leave.

 @nnja

Some lingering debt is inevitable.

Don't be a perfectionist.

Figure out the project tolerance, and work with it.

@nnja

Use these arguments to justify the additional time it takes to do things right

@nnja


To Win The Fight, Pay Down Your Debt
@nnja

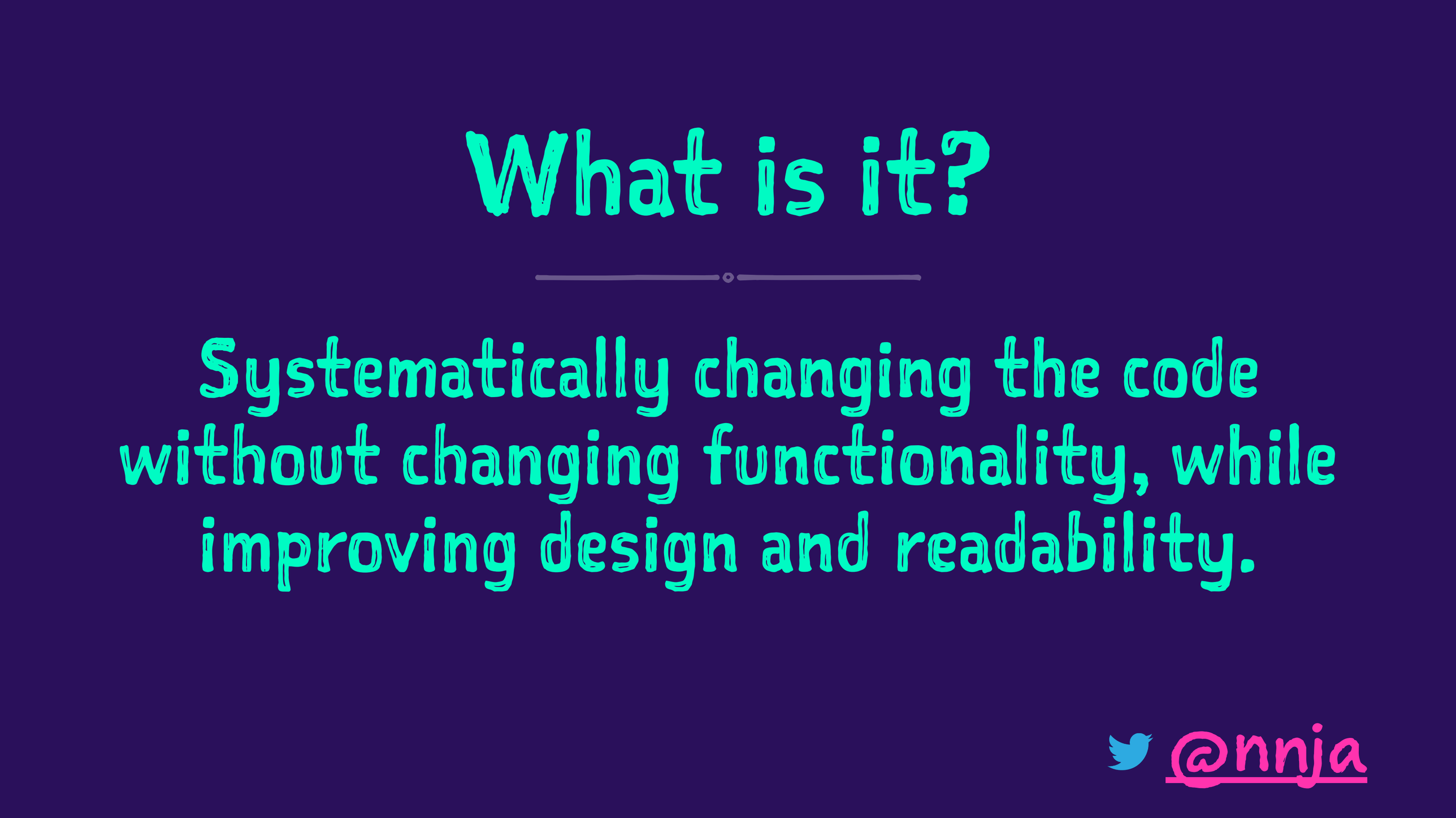
Refactoring

The single greatest tool in your toolbox

@nnja

What is it?

Systematically changing the code without changing functionality, while improving design and readability.



Refactoring

- Slow and steady wins the race.
- The end goal is to refactor without breaking existing functionality.

@nnja

Refactoring

- Replace functions and modules incrementally.
- Test as you go.
- Tests are *mandatory* at this step.

@nnja

yelp Engineering

Undebt: How We Refactored 3 Million Lines of Code

Evan H., Software Engineering Intern

Aug 23, 2016

github.com/Yelp/undebt, yelp refactoring

Use proper design patterns

README.md

python-patterns

A collection of design patterns and idioms in Python.

github.com/faif/python-patterns

Use depreciation patterns

Like openstack debtcollector

```
class removed_property(object):
    """Property descriptor that deprecates a property.
    This works like the ``@property`` descriptor but can
    be used instead to provide the same functionality
    and also interact with the :mod:`warnings` module to
    warn when a property is accessed, set and/or deleted.
    """
```

@nnja

Use vulture.py to find dead or unreachable code

```
$ pip install vulture
$ vulture script.py package/
```

or

```
$ python -m vulture script.py package/
```

github.com/jendrikseipp/vulture

sample code

```
def foo():
    print("foo")

def bar():
    print("bar")

def baz():
    print("baz")

foo()
bar()
```

vulture.py output

```
› python -m vulture foo.py
foo.py:7: unused function 'baz' (60% confidence)
```

Prioritize

What causes the biggest & most frequent pain points for developers?

@nnja


Just like with
monetary debt,
pay off the high interest
loan first.
@nnja

Shelf Life

What's the life expectancy of this project?

Longer shelf life -> higher debt interest

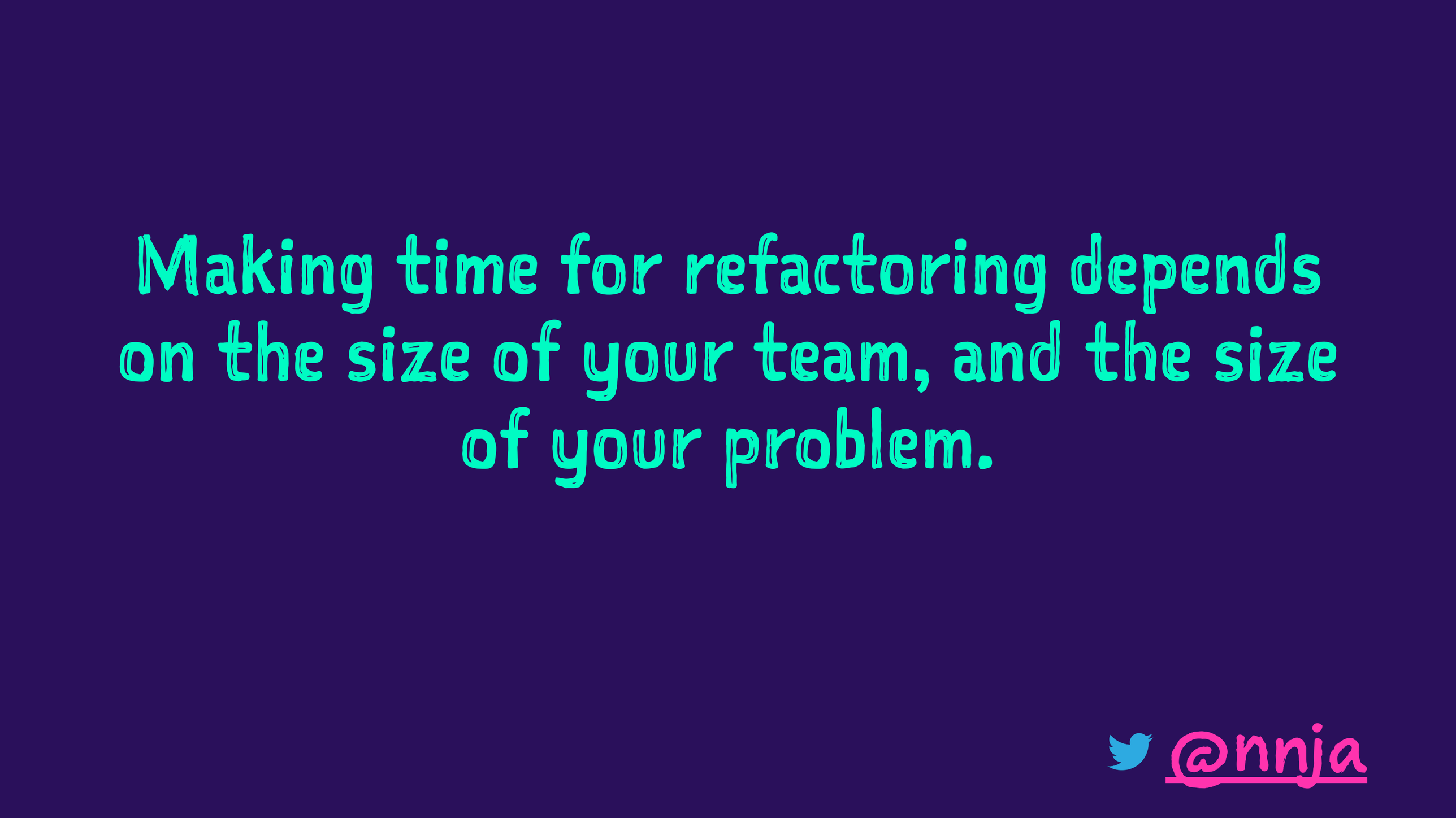
@nnja

Technical debt can be strategic

If you don't have to pay it off, you got something for nothing.

 @nnja

Making time for refactoring depends on the size of your team, and the size of your problem.



Guidelines

- Small
  - Devote a week every 6-8 weeks
- Medium
  - Devote a person every 1-4 weeks, rotate
- Large

@nnja


A Few Last Tips
@nnja

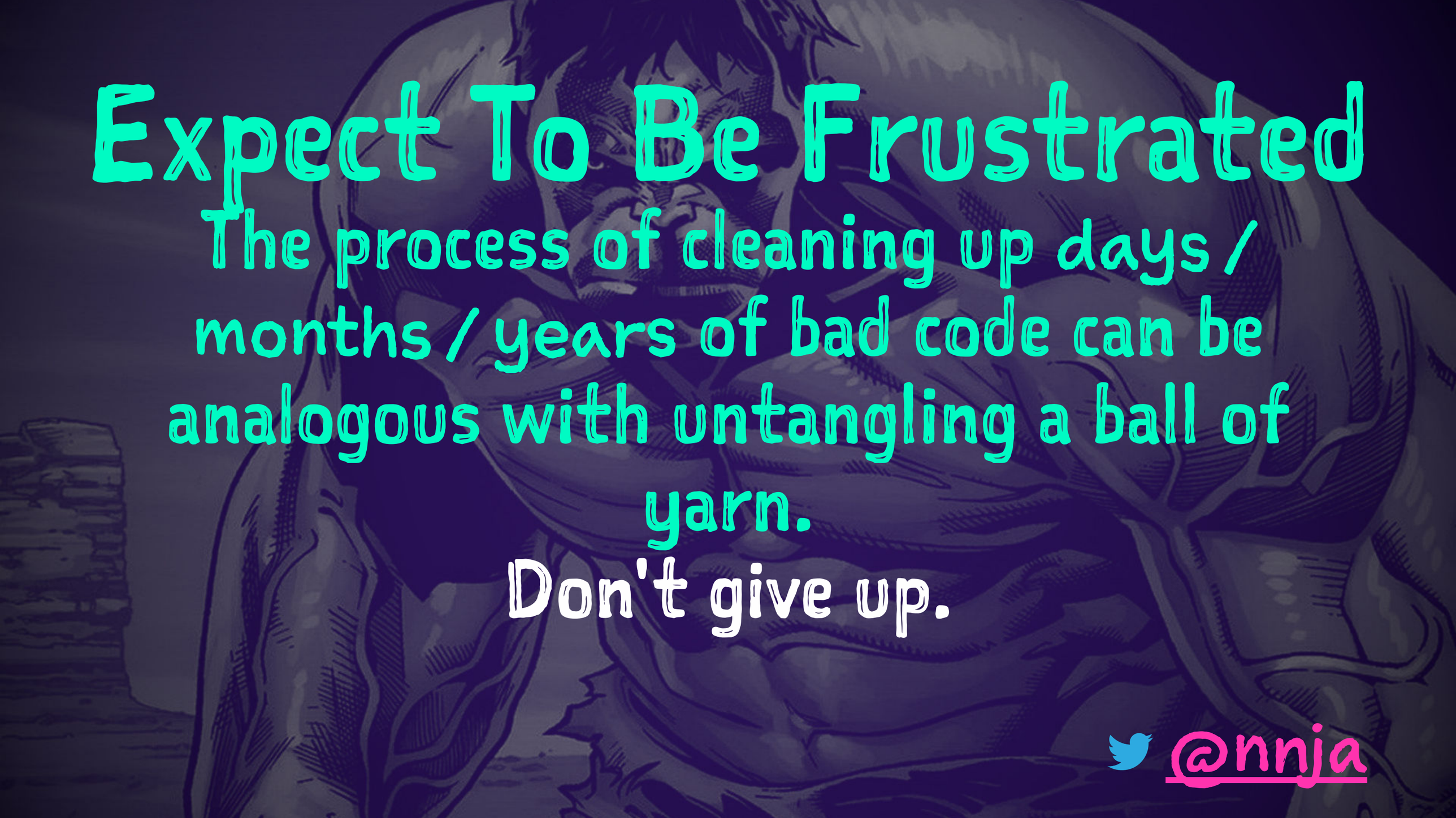
Code should be for humans

@nnja

Boy Scout Rule

"Always check in a module cleaner than when you checked it out."

Source


Expect To Be Frustrated
The process of cleaning up days / months / years of bad code can be analogous with untangling a ball of yarn.
Don't give up.
@nnja


YOU CAN
DO IT!
memegenerator.net

Thank You!

Python @ Microsoft:
bit.ly/parispython

@nnja

 @nnja