Coding with Confidence

Adding TDD to Your Toolset

image: un-sharp @ flickr

why test?

reduce defects

verify expectations

why test first?

"The metric I, and others I know, have used to judge unit testing is: *does it find bugs?*"

Bill Moorier

kinda.

"When you write unit tests ... you scrutinize, you think, and often you prevent problems without even encountering a test failure."

Michael Feathers

Test-Driven Development

Test-Driven Design

1. Design the API

2. Make it work

3. Make it clean

↑ cohesion

↓ coupling

“As long as you don't change that component's external interfaces, you can be comfortable that you won't cause problems that ripple through the entire system”

The Pragmatic Programmer

avoid complicated solutions to simple problems

TELEPHONE 393

FRED J. ROWLANDS & CO.

Successors to the Butte Insurance Agency

FIRE
PLATE
GLASS...

INSURANCE

LIFE
AND
ACCIDENT

BEN J. CORNELIUS, MANAGER

51 East Broadway

regressions

mess

image: locket479 @ flickr

"Consider a building with a few broken windows. If the windows are not repaired, the tendency is for vandals to break more windows"

The Atlantic Monthly (1982)

fear

image: troyholden @ flickr

“Test-driven development is a way of managing fear during programming”

Kent Beck

documentation

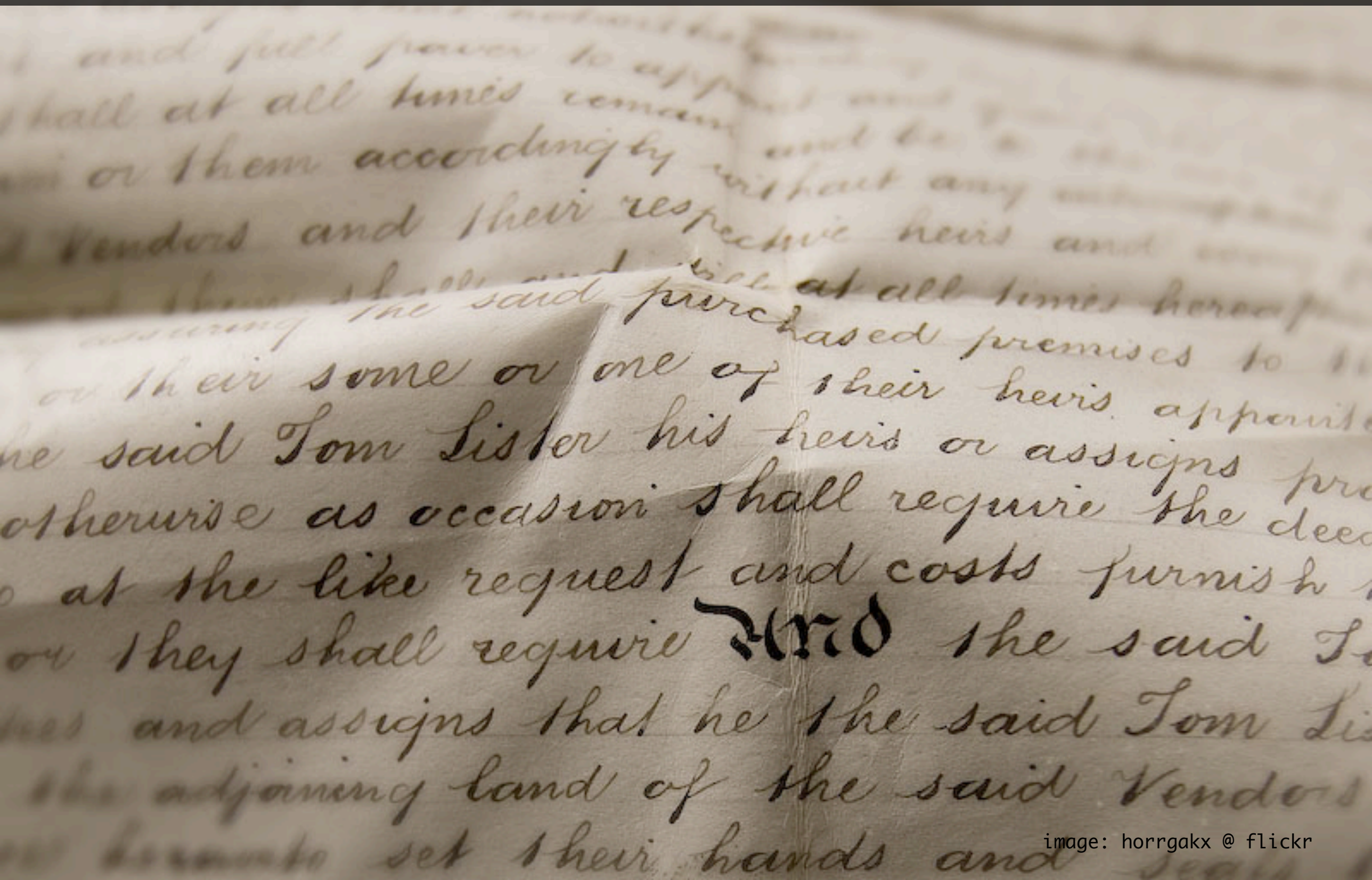
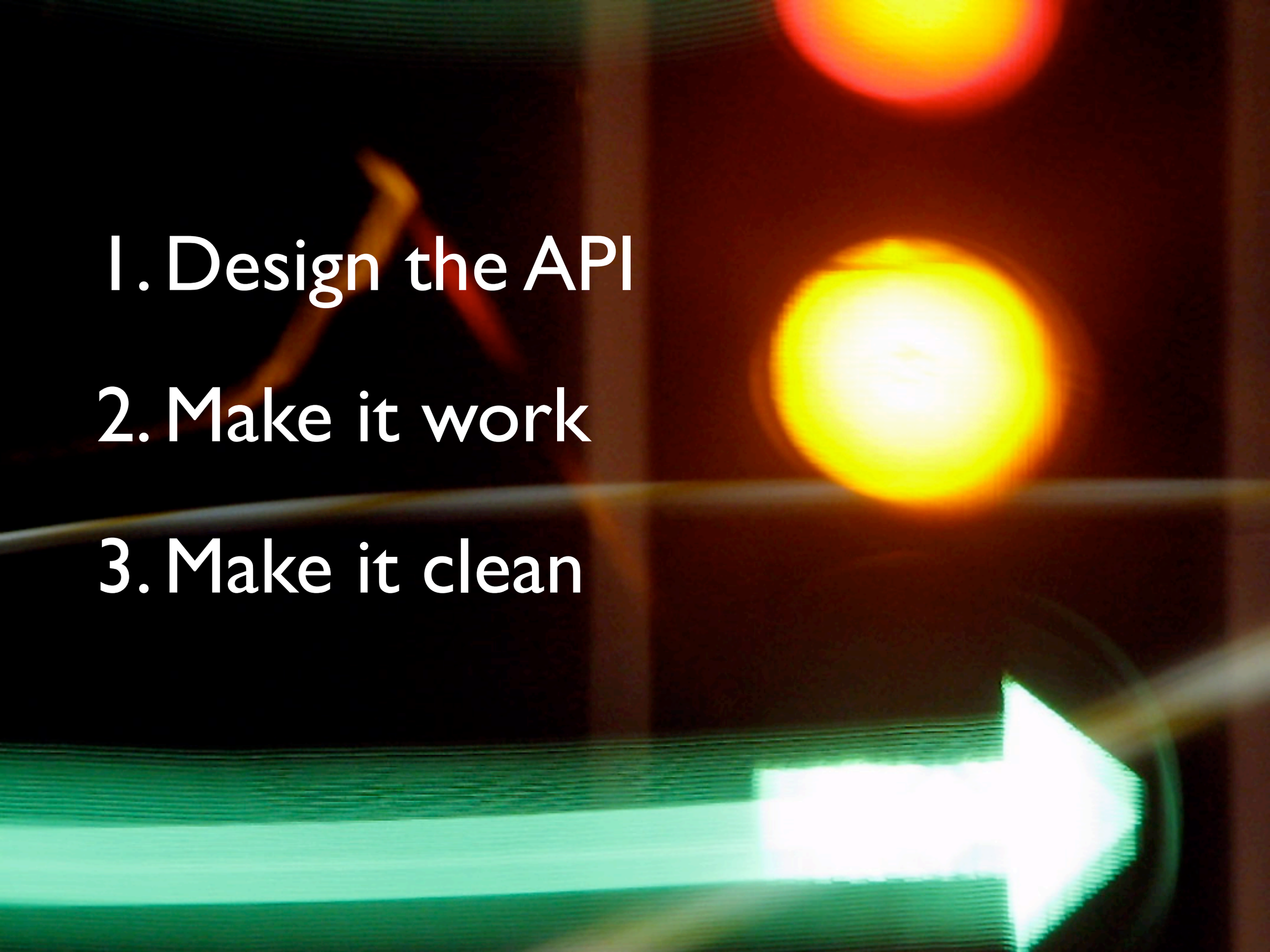
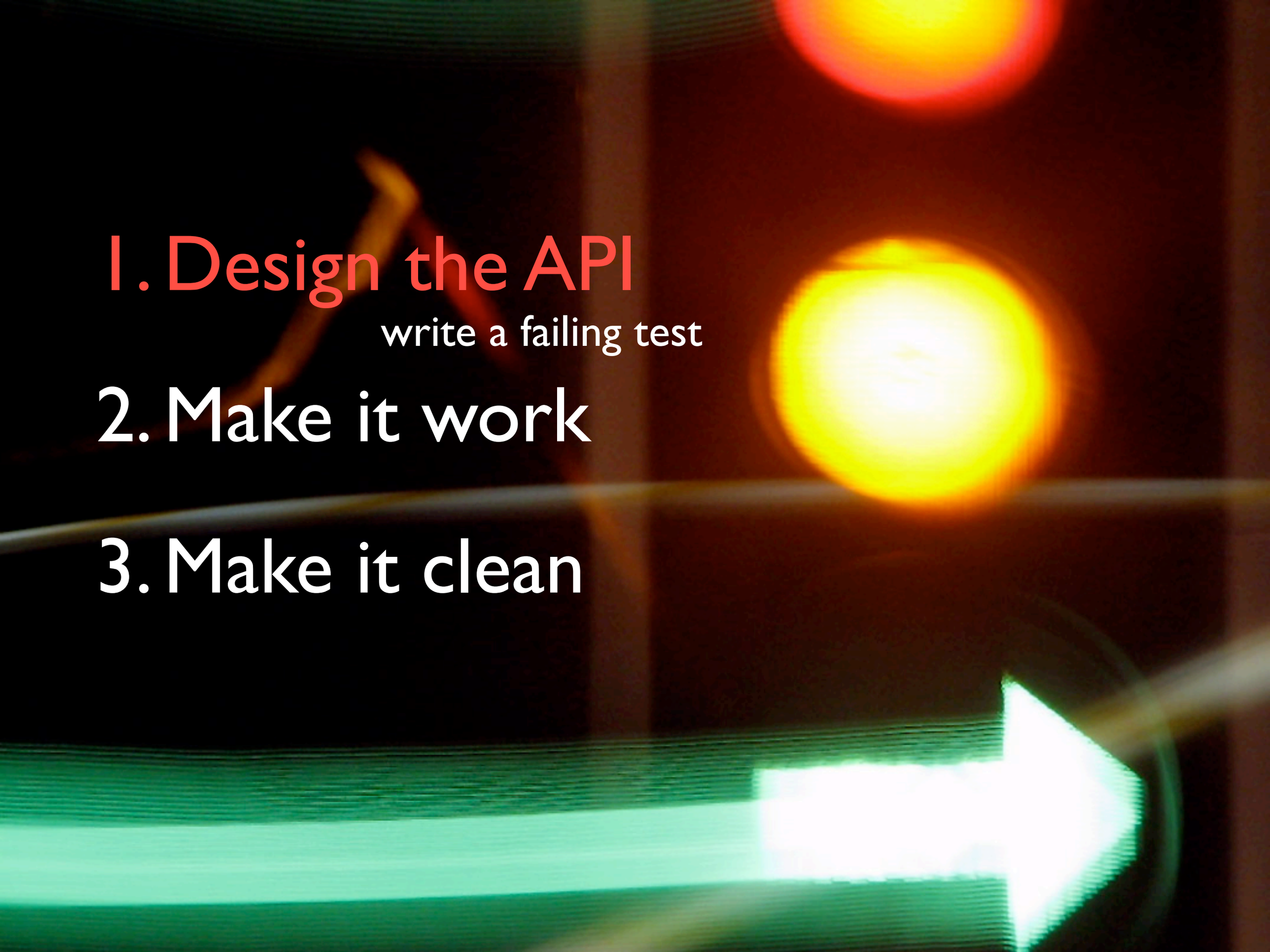
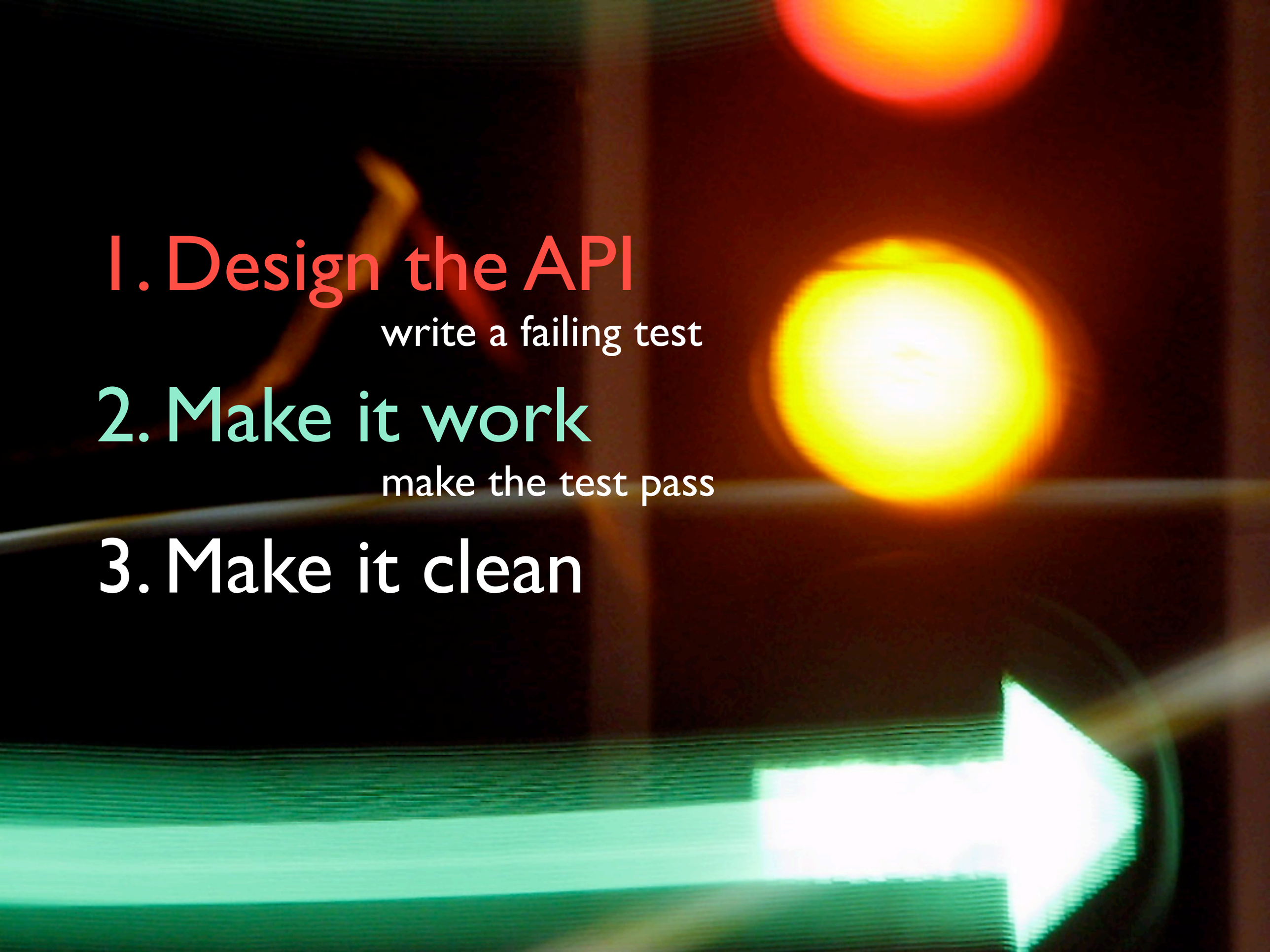
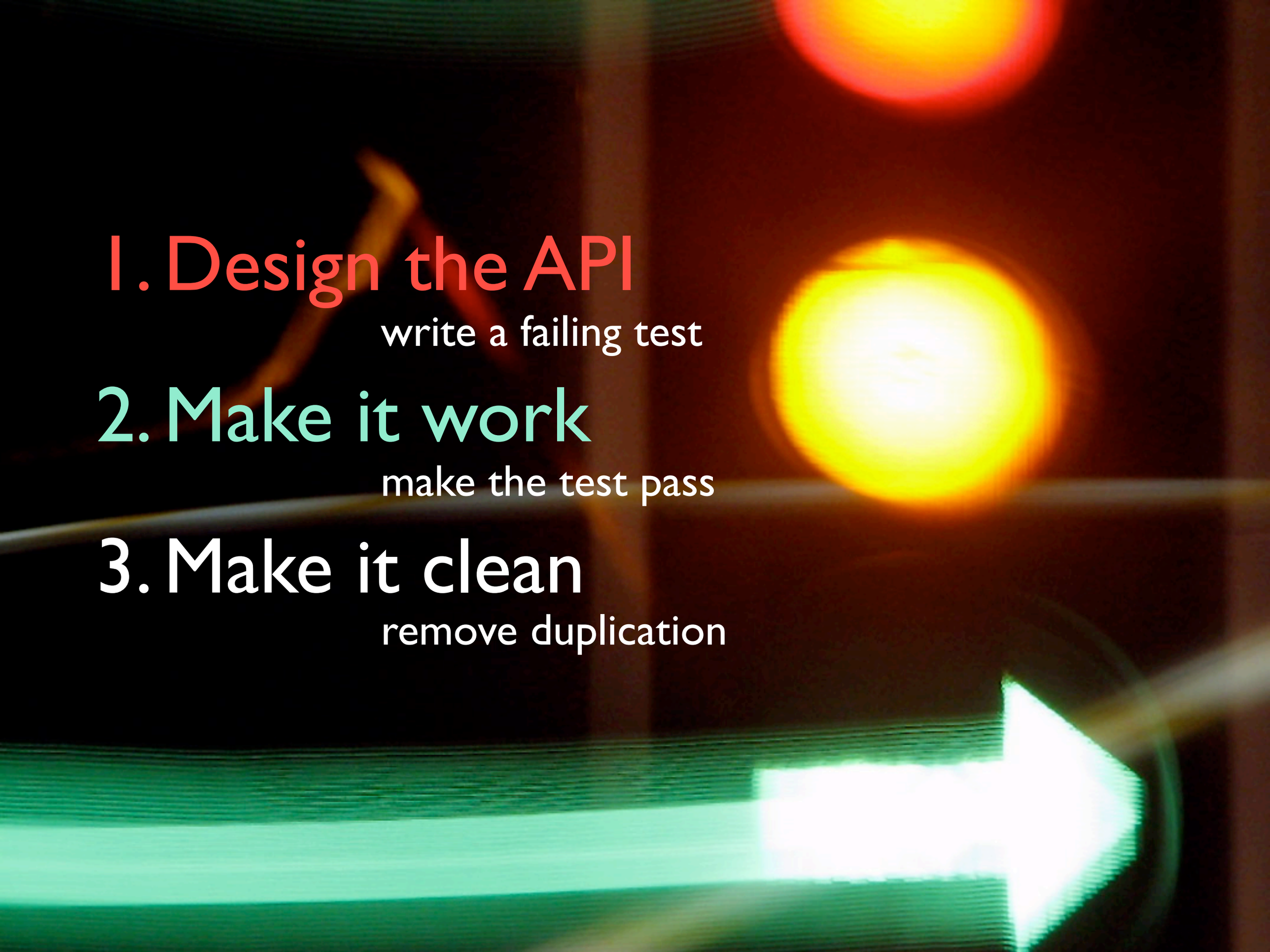


image: horrgakx @ flickr


1. Design the API
2. Make it work
3. Make it clean


1. Design the API
write a failing test
2. Make it work
3. Make it clean

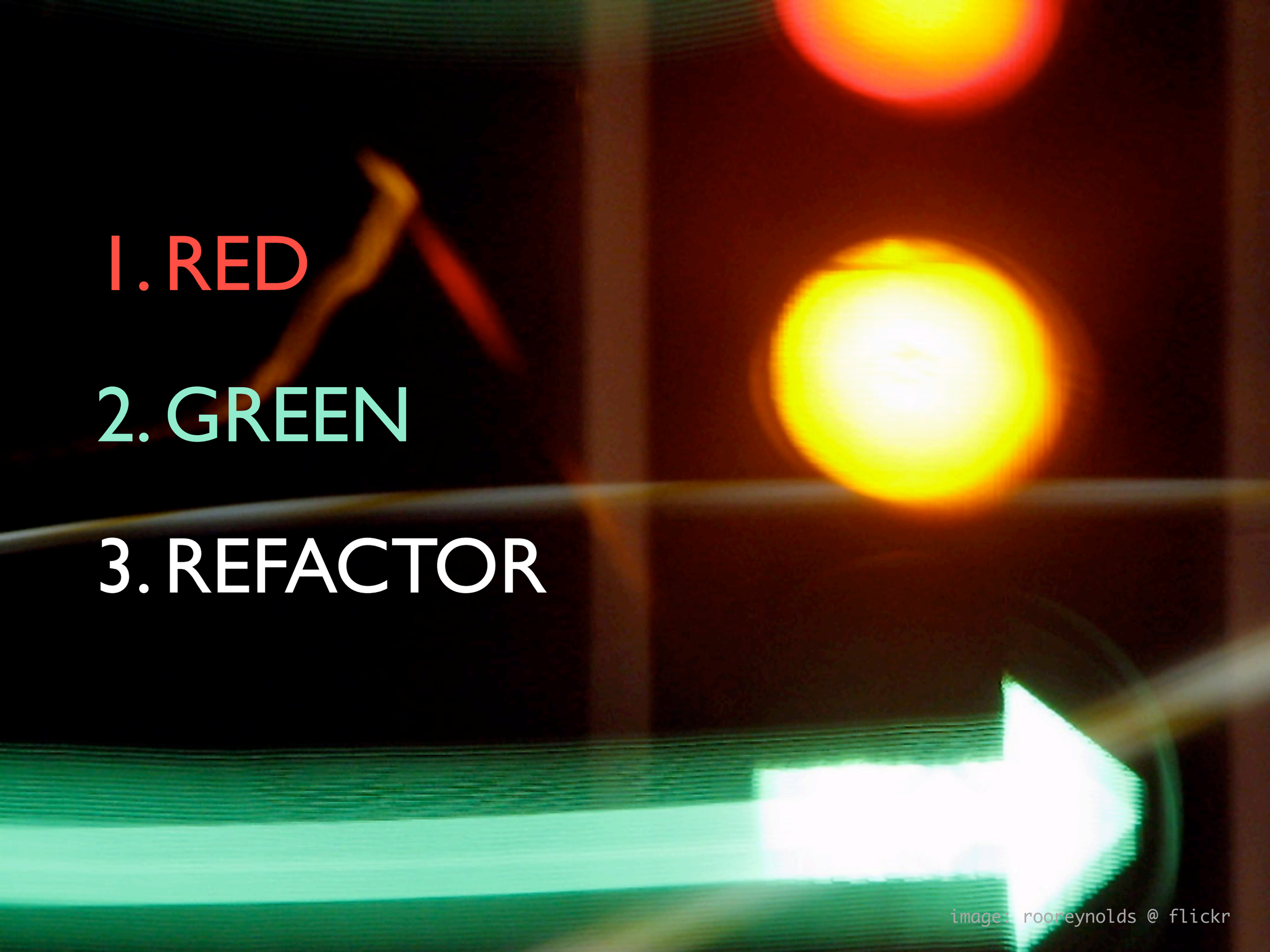

1. Design the API
write a failing test
2. Make it work
make the test pass
3. Make it clean


1. Design the API
write a failing test
2. Make it work
make the test pass
3. Make it clean
remove duplication

1. RED

2. GREEN

3. REFACTOR



```ruby
class FabricatedTest < Test::Unit::TestCase

  def setup
    File.touch('tempfile')
  end

  def teardown
    FileUtils.rm('tempfile')
  end

  def test_should_know_the_number_of_items
    bag = Bag.new(3)
    assert_equal 3, bag.item_count
  end

end
```

anatomy

```
class FabricatedTest < Test::Unit::TestCase
  def setup
    File.touch('tempfile')
  end

  def teardown
    FileUtils.rm('tempfile')
  end

  def test_should_know_the_number_of_items
    bag = Bag.new(3)
    assert_equal 3, bag.item_count
  end

end
```

Test case

anatomy

```ruby
class FabricatedTest < Test::Unit::TestCase

  def setup
    File.touch('tempfile')
  end

  def teardown
    FileUtils.rm('tempfile')
  end

  def test_should_know_the_number_of_items
    bag = Bag.new(3)
    assert_equal 3, bag.item_count
  end

end
```

Test case

Test setup
(run every test)

anatomy

```ruby
class FabricatedTest < Test::Unit::TestCase

  def setup
    File.touch('tempfile')
  end

  def teardown
    FileUtils.rm('tempfile')
  end

  def test_should_know_the_number_of_items
    bag = Bag.new(3)
    assert_equal 3, bag.item_count
  end

end
```

Test case

Test setup
(run every test)

Test teardown
(run every test)

anatomy

```ruby
class FabricatedTest < Test::Unit::TestCase

  def setup
    File.touch('tempfile')
  end

  def teardown
    FileUtils.rm('tempfile')
  end

  def test_should_know_the_number_of_items
    bag = Bag.new(3)
    assert_equal 3, bag.item_count
  end

end
```

Test case

Test setup
(run every test)

Test teardown
(run every test)

Test

anatomy

```ruby
class FabricatedTest < Test::Unit::TestCase

  def setup
    File.touch('tempfile')
  end

  def teardown
    FileUtils.rm('tempfile')
  end

  def test_should_know_the_number_of_items
    bag = Bag.new(3)
    assert_equal 3, bag.item_count
  end

end
```

Test case

Test setup
(run every test)

Test teardown
(run every test)

Test

Assertion

anatomy

image: mediageek @ flickr

Pitfalls

```ruby
class User
  attr_accessor :name
end

class UserTest < Test::Unit::TestCase

  def test_should_be_able_to_set_name
    user = User.new
    user.name = 'Patrick'
    assert_equal 'Patrick', user.name
  end

end
```

testing language features

```ruby
class UserTest < Test::Unit::TestCase

  def test_valid
    user = User.new

    assert !user.valid?
    assert user.errors.on(:name)
    assert user.errors.on(:username)
  end

end
```

multiple assertions

```ruby
class Line

  def initialize(content)
    @content = content
  end

  private
  def extract(pattern)
    @content.match(pattern)[1]
  end

end
```

```ruby
class LineTest < Test::Unit::TestCase

  def test_extract_should_return_captured_value
    line = Line.new('This is content')
    assert_equal 'is', line.send(:extract, /(is)/)
  end

end
```

testing private methods

```ruby
class UserTest < Test::Unit::TestCase

  def setup
    @content = '{"user_name":"reagent","user_id":"12345"}'
  end

  def test_should_be_able_to_extract_the_username
    user = User.new(@content)
    assert_equal 'reagent', user.username
  end

  def test_should_be_able_to_extract_the_user_id
    user = User.new(@content)
    assert_equal '12345', user.id
  end

end
```

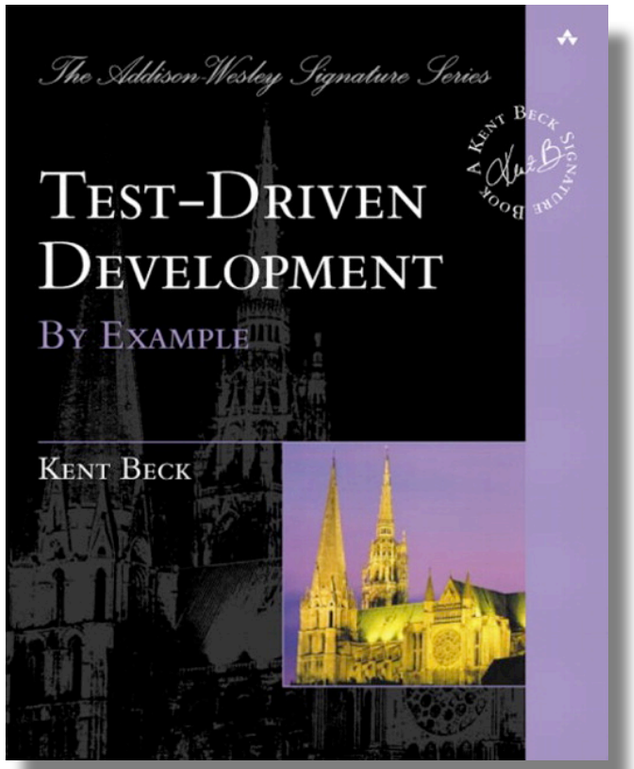
lack of context

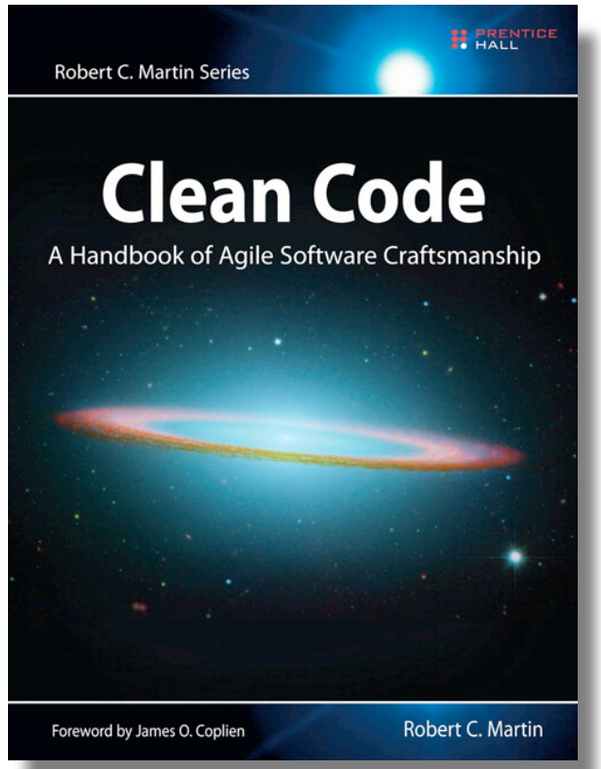
```ruby
class User
  attr_writer :name

  def name
    @name.strip
  end
end

class UserTest < Test::Unit::TestCase

  def test_should_remove_whitespace_from_name
    user = User.new
    user.name = ' Patrick '

    assert_equal 'Patrick', user.name
  end

end
```

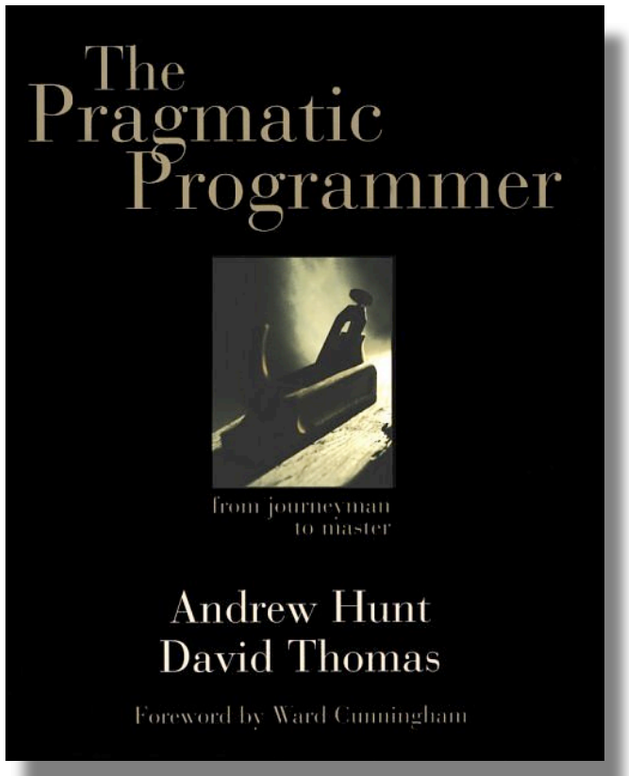
testing the happy path

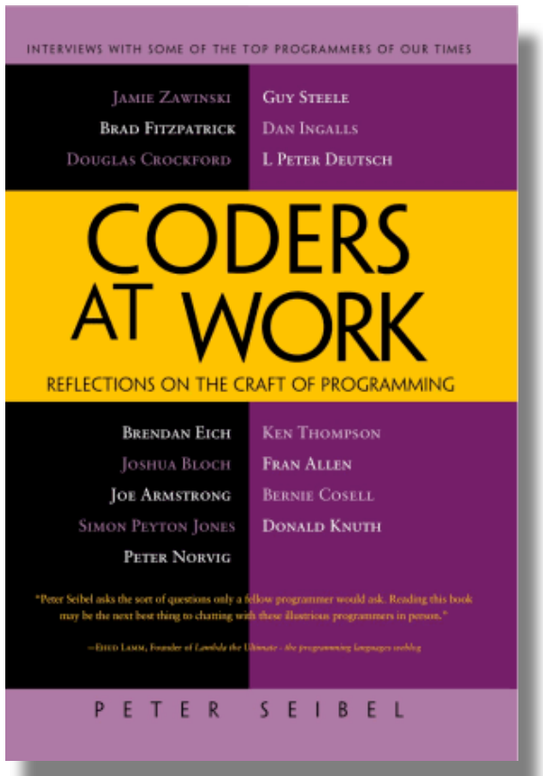
TDD is a tool

TDD is not a religion


The Addison-Wesley Signature Series
A KENT BECK SIGNATURE BOOK
TEST-DRIVEN DEVELOPMENT
BY EXAMPLE
KENT BECK


Robert C. Martin Series
PRENTICE HALL
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin


The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas
Foreword by Ward Cunningham


INTERVIEWS WITH SOME OF THE TOP PROGRAMMERS OF OUR TIMES
JAMIE ZAWINSKI
BRAD FITZPATRICK
DOUGLAS CROCKFORD
GUY STEELE
DAN INGALLS
L PETER DEUTSCH
CODERS AT WORK
REFLECTIONS ON THE CRAFT OF PROGRAMMING
BRENDAN EICH
JOSHUA BLOCH
JOE ARMSTRONG
SIMON PEYTON JONES
PETER NORVIG
KEN THOMPSON
FRAN ALLEN
BERNIE COSELL
DONALD KNUTH
PETER SEIBEL

Resources

- Slides / Code: http://github.com/reagent/talks

Contact

- patrick.reagan@viget.com
- http://twitter.com/reagent

Rate This Talk

- http://speakerrate.com/preagan

image: nomeacuerdo @ flickr