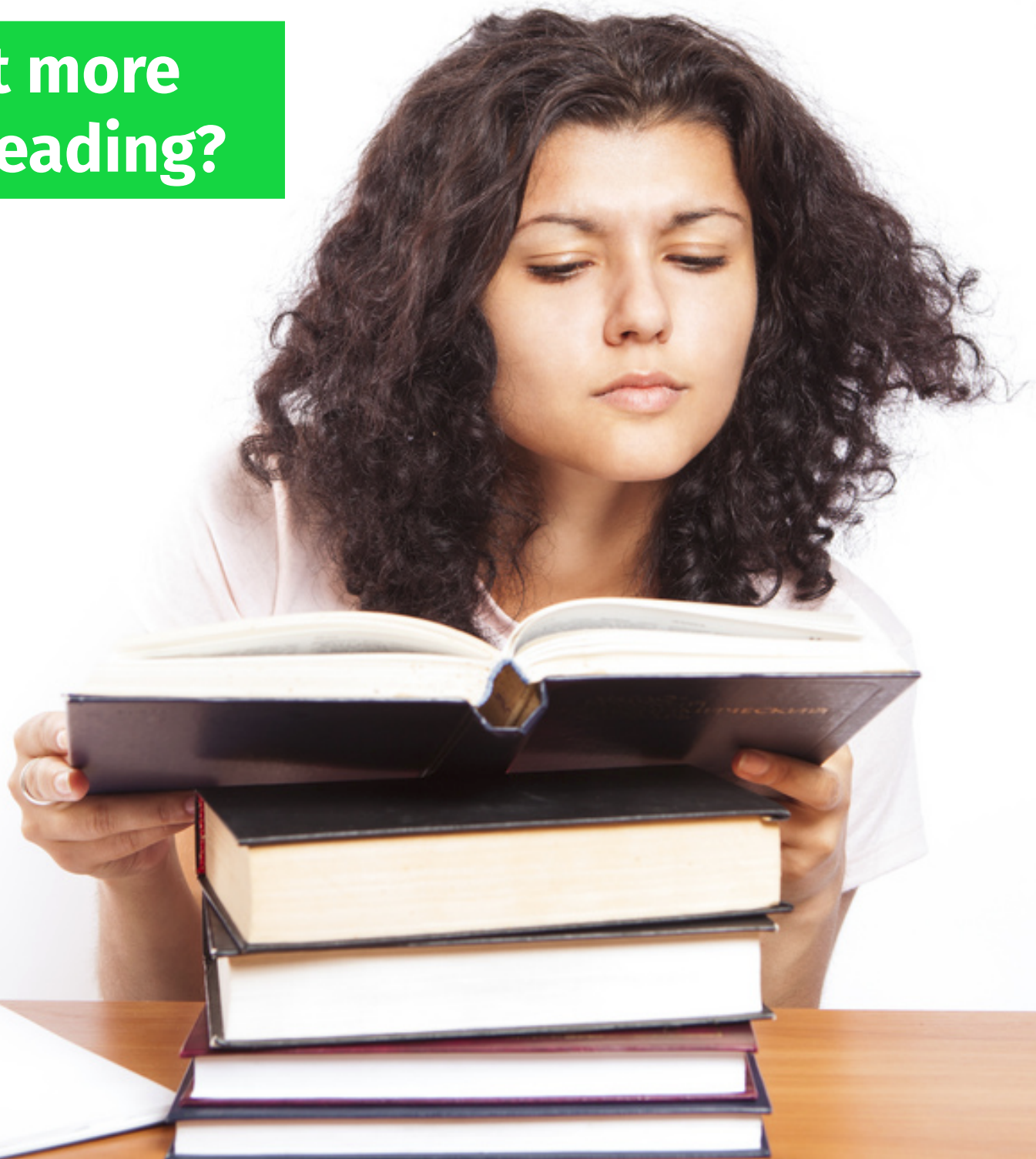




**Isn't it more  
about reading?**



Written **once** -  
read **many** times

(...) when you program, you have to **think about** how someone **will read your code**, not just how a computer will interpret it.

Kent Beck

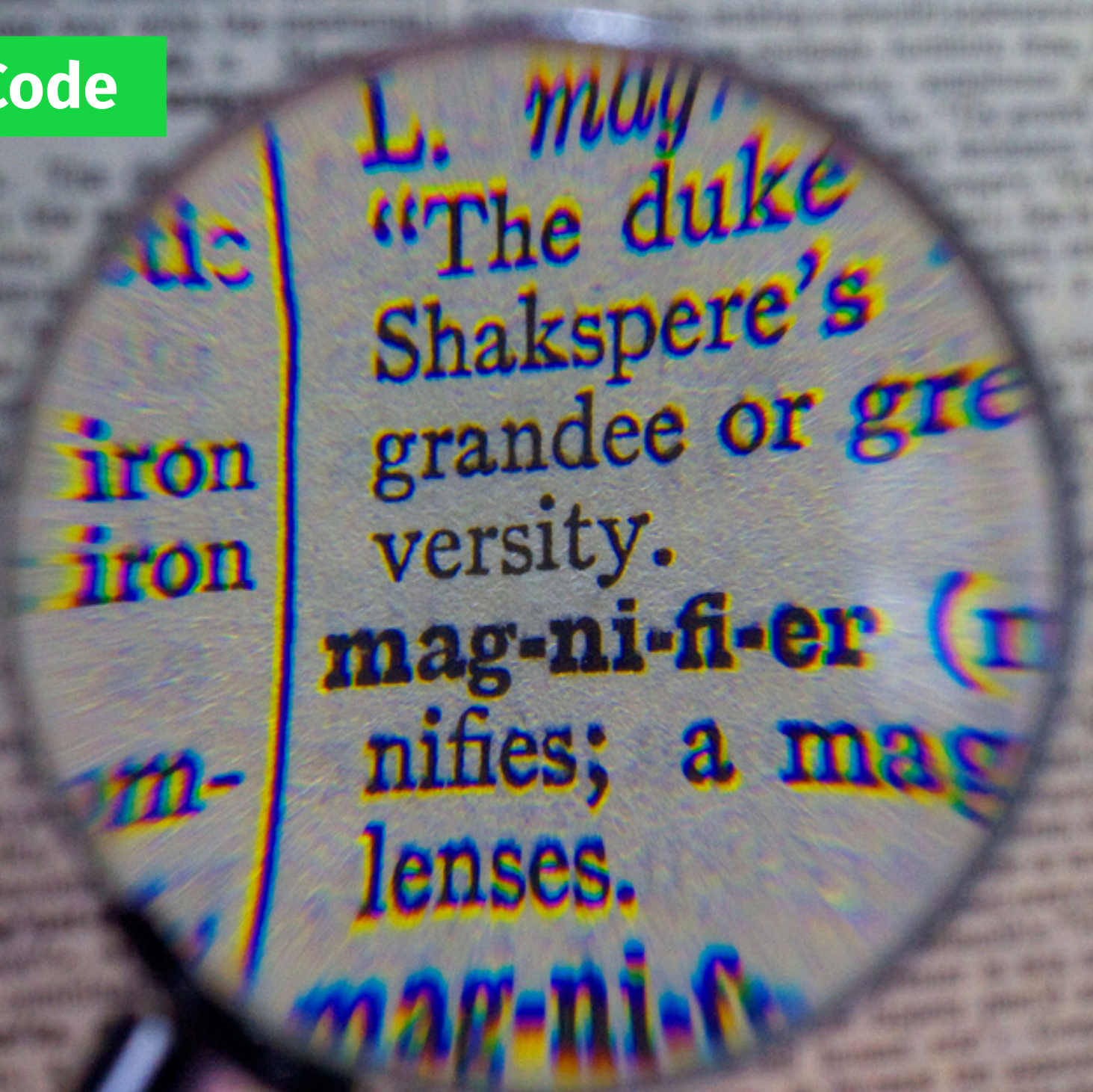
*Any fool can write code that a computer can understand. Good programmers write code that humans can understand.*

Martin Fowler

**Not about architecture**



# Methods & Code



**magnifier**

magnifies; a magnifying glass.

# Nurturing a code base



**Extra effort**



**Save time!**



A white plastic chair is positioned against a brick wall. The chair's seat is supported by a stack of approximately six bricks. A computer keyboard is placed on the chair's seat. To the right of the chair, there is a pile of debris on the ground, including a crumpled white plastic bag and some leaves. An orange rectangular box with white text is overlaid on the right side of the image.

**Your code base?**



**It's about joy!**



# Code, Comments, Concepts, Comprehension – Conclusion?

Tobias Pfeiffer  
@PragTob  
pragtob.info

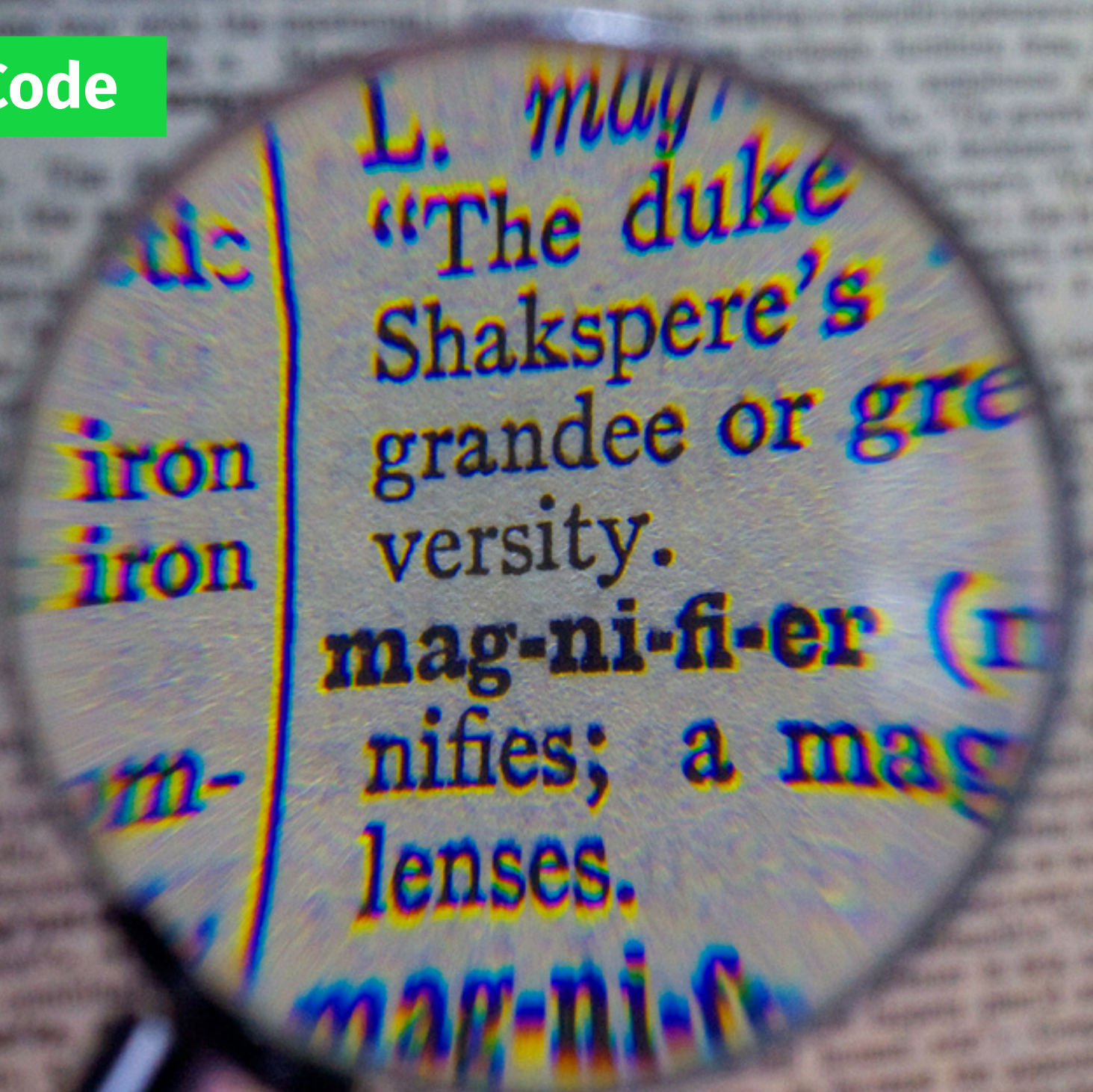


**L I E F E R Y**

**Crazy?**



# Methods & Code

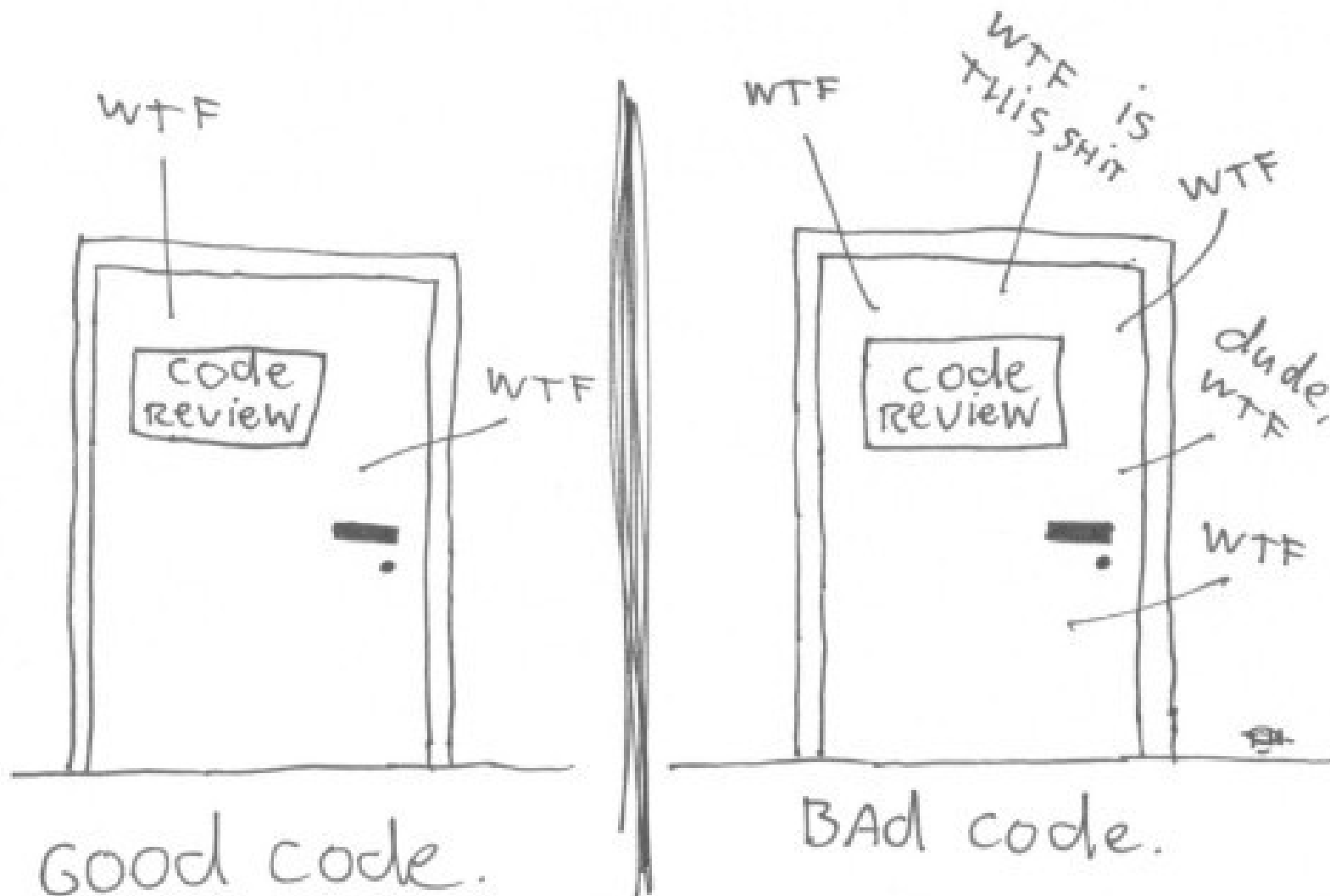


L. May  
"The duke  
Shakspeare's  
grandee or gre  
versity.

**magnifier** (T)  
nifies; a mag  
lenses.

magnifier

The ONLY VALID MEASUREMENT  
OF Code QUALITY: WTFs/MINUTE



**K**eeP

**I**t

**S**imple

**S**tupid

Are comments a **code smell**?

Comments are an **excuse** of the code that it could not be **clearer**.

Outdated comments are the **worst**

**WHY**

not the

**WHAT**

## WHY comment

```
def paint_control(event)
  # some painting code
rescue => e
  # Really important to rescue here.
  Failures that escape this method
  # cause odd-ball hangs with no
  backtraces. See #559 for an example.
  #
  puts "SWALLOWED PAINT EXCEPTION ON
#{@obj} - go take care of it: " + e.to_s
  puts 'Unfortunately we have to swallow
it because it causes odd failures :('
end
```

**Comments** are the smell that **tries**  
to make other **smells seem ok**

# do one thing

• • •

• • •

• • •

• • •

# do another thing

• • •

• • •

• • •

• • •

# do something more

• • •

• • •

# do one thing

• • •

• • •

• • •

• • •

# do another thing

• • •

• • •

• • •

• • •

# do something more

• • •

• • •

**# do one thing**

Cocepts

• • •

• • •

• • •

• • •

**# do another thing**

• • •

• • •

• • •

• • •

**# do something more**

• • •

• • •

**Method too long**

# Short Methods

**<= 8 LOC**

# Extract Methods

## Concepts

```
do_one_thing  
do_another_thing  
do_something_more
```

```
//Create a new session from a session data received by the API
```

```
func (session *Session) Create() error {
```

```
    //Check we have an app token
```

```
    if len(session.AppVersion.App.Key) <= 0 {
```

```
        return &Error{Code: AppNotFound, Message: "No app provided"}
```

```
    }
```

```
    //Check if the app is a valid app
```

```
    var app App
```

```
    val := Db.Where(session.AppVersion.App).First(&app)
```

```
    if val.Error != nil {
```

```
        return &Error{Code: AppNotFound, Message: "App not found"}
```

```
    }
```

```
    //Check if the team has reached the limit or not
```

```
    team := app.GetTeam()
```

```
    if team.IsWithinLimits() == false {
```

```
        return &Error{Code: LogLimitReached, Message: "Daily log limit reached"}
```

```
    }
```

```
    //Set the real app
```

```
    session.AppVersion.App = app
```

```
    session.AppVersion.AppId = app.Id
```

```
    session.AppId = app.Id
```

```
    //Set operating system in app if not informed
```

```
    if app.OperatingSystem != OPERATING_SYSTEM_ANDROID && session.SdkType == "android" {
```

```
        app.OperatingSystem = OPERATING_SYSTEM_ANDROID
```

```
        Db.Save(&app)
```

```
    }
```

People disagree



**tessi** 9:50 AM

note to myself: comments seem to be a good trigger to 🗨️ tobi



*"The more complex the code, the more comments it should have."*

You should **make the code less complex** **not add more comments.**

Tiago Teixeira

```
# context, outlet, times, time per  
step, state, data  
def pattern(c, o, t, l, s, d)  
    # ...  
end
```

**Incomprehensible** names

```
# context, outlet, times, time per  
step, state, data  
def pattern(c, o, t, l, s, d)  
    # ...  
end
```

```
# context, outlet, times, time per  
step, state, data  
def pattern(c, o, t, l, s, d)  
  # ...  
end
```

# Explanatory names

Naming is **hard**

# One Language



```
def pattern(context, outlet, time,  
            time_per_step, state,  
            data)  
    # ...  
end
```

Argument **order dependency**

Try to keep it to **2 parameters**

**Example**



```
# allowed to drink?  
if customer.age >= 18  
  say 'Okay'  
  drink = prepare_drink requested_drink  
  say 'here you go'  
  hand_drink_over drink, customer  
else  
  say 'I am sorry you are not legally  
      allowed rather to drink here'  
  say "Would you rather have a  
      #{['cola', 'mate'].sample}?"  
end
```

```
# allowed to drink?  
if customer.age >= 18  
  say 'Okay'  
  drink = prepare_drink requested_drink  
  say 'here you go'  
  hand_drink_over drink, customer  
else  
  say 'I am sorry you are not legally  
    allowed rather to drink here'  
  say "Would you rather have a  
    #{['cola', 'mate'].sample}?"  
end
```

## Magic „Numbers“

```
# allowed to drink?  
if customer.age >= 18  
  say 'Okay'  
  drink = prepare_drink requested_drink  
  say 'here you go'  
  hand_drink_over drink, customer  
else  
  say 'I am sorry you are not legally  
    allowed rather to drink here'  
  say "Would you rather have a  
    #{ ['cola', 'mate'].sample }?"  
end
```

```
NON_ALCOHOLIC_DRINKS = ['cola', 'mate']  
MIN_DRINKING_AGE     = 18
```

```
# allowed to drink?  
if customer.age >= MIN_DRINKING_AGE  
  say 'Okay'  
  drink = prepare_drink requested_drink  
  say 'here you go'  
  hand_drink_over drink, customer  
else  
  say 'I am sorry you are not legally  
    allowed rather to drink here'  
  say "Would you rather have a  
    {NON_ALCOHOLIC_DRINKS.sample}?"  
end
```

```
# allowed to drink?  
if customer.age >= MIN_DRINKING_AGE  
  say 'Okay'  
  drink = prepare_drink requested_drink  
  say 'here you go'  
  hand_drink_over drink, customer  
else  
  say 'I am sorry you are not legally  
    allowed rather to drink here'  
  say "Would you rather have a  
    {NON_ALCOHOLIC_DRINKS.sample}?"  
end
```

```
# allowed to drink?  
if customer.age >= MIN_DRINKING_AGE  
  say 'Okay'  
  drink = prepare_drink requested_drink  
  say 'here you go'  
  hand_drink_over drink, customer  
else  
  say 'I am sorry you are not legally  
    allowed rather to drink here'  
  say "Would you rather have a  
    #{NON_ALCOHOLIC_DRINKS.sample}?"  
end
```

# Query method

**Intention revealing method**

```
# ...
```

```
text.color = red
```

```
# ...
```

```
# ...
```

```
text.color = red
```

```
# ...
```

# ...

**highlight**( *text* )

# ...

```
def highlight(text)
  text.color = red
end
```

```
def highlight(text)
  text.color      = red
  text.underline  = true
  update_highlights
end
```

```
# ...  
text.color      = red  
text.underline = true  
update_highlights  
# ...
```

```
# ...
```

```
highlight(text)
```

```
# ...
```

```
# allowed to drink?  
if customer.age >= MIN_DRINKING_AGE  
  say 'Okay'  
  drink = prepare_drink requested_drink  
  say 'here you go'  
  hand_drink_over drink, customer  
else  
  say 'I am sorry you are not legally  
    allowed rather to drink here'  
  say "Would you rather have a  
    {NON_ALCOHOLIC_DRINKS.sample}?"  
end
```

```
# allowed to drink?  
if customer.age >= MIN_DRINKING_AGE  
  say 'Okay'  
  drink = prepare_drink requested_drink  
  say 'here you go'  
  hand_drink_over drink, customer  
else  
  say 'I am sorry you are not legally  
    allowed rather to drink here'  
  say "Would you rather have a  
    #{NON_ALCOHOLIC_DRINKS.sample}?"  
end
```

```
# allowed to drink?  
if customer.age >= MIN_DRINKING_AGE  
  say 'Okay'  
  drink = prepare_drink requested_drink  
  say 'here you go'  
  hand_drink_over drink, customer  
else  
  say 'I am sorry you are not legally  
    allowed rather to drink here'  
  say "Would you rather have a  
    {NON_ALCOHOLIC_DRINKS.sample}?"  
end
```

```
# allowed to drink?  
if customer.age >= MIN_DRINKING_AGE  
  say 'Okay'  
  drink = prepare_drink requested_drink  
  say 'here you go'  
  hand_drink_over drink, customer  
else  
  say 'I am sorry you are not legally  
    allowed rather to drink here'  
  say "Would you rather have a  
    {NON_ALCOHOLIC_DRINKS.sample}?"  
end
```

```
if allowed_to_drink_alcohol?(customer)
  serve_drink requested_drink, customer
else
  propose_non_alcoholic_drink
end
```

„If you have a *good name* for a  
method you *don't need to look at*  
*the body.*“

Martin Fowler

*„The easiest code to **understand** is  
the code you **don't have to read** at  
all.“*

Tom Stuart (Berlin)

```
def serve_alcoholic_drink(customer, requested_drink)
  if allowed_to_drink_alcohol?(customer)
    serve_drink requested_drink, customer
  else
    propose_non_alcoholic_drink
  end
end
```

**Method Order**

```
# more public methods
```

```
private
```

```
def allowed_to_drink_alcohol?(customer)
end
```

```
def serve_drink(requested_drink, customer)
end
```

```
def propose_non_alcoholic_drink
end
```

```
def serve_alcoholic_drink(customer, requested_drink)
  if allowed_to_drink_alcohol?(customer)
    serve_drink requested_drink, customer
  else
    propose_non_alcoholic_drink
  end
end
```



```
# more public methods
```

```
private
```

```
def allowed_to_drink_alcohol?(customer)
end
```

```
def serve_drink(requested_drink, customer)
end
```

```
def propose_non_alcoholic_drink
end
```



```
def serve_alcoholic_drink(customer, requested_drink)
  if allowed_to_drink_alcohol?(customer)
    serve_drink requested_drink, customer
  else
    propose_non_alcoholic_drink
  end
end
```

**Method Order**

```
defp allowed_to_drink_alcohol?(customer)
end
```

```
defp serve_drink(requested_drink, customer)
end
```

```
defp propose_non_alcoholic_drink
end
```

```
# more public methods
```

```
def serve_alcoholic_drink(customer, requested_drink)
  if allowed_to_drink_alcohol?(customer)
    serve_drink requested_drink, customer
  else
    propose_non_alcoholic_drink
  end
end
```

```
defp allowed_to_drink_alcohol?(customer)
end
```

```
defp serve_drink(requested_drink, customer)
end
```

```
defp propose_non_alcoholic_drink
end
```

```
# more public methods
```



```
prepare_drink requested_drink
price = requested_drink.price
check = Check.new
check.add_price price
say 'That will be ' + check.total
```

```
prepare_drink requested_drink  
price = requested_drink.price  
check = Check.new  
check.add_price price  
say 'That will be ' + check.total
```

```
prepare drink requested drink
price = requested_drink.price
check = Check.new
check.add_price price
say 'That will be ' + check.total
```

Same **level of abstraction** in a  
method

prepare\_drink requested\_drink  
prepare\_check requested\_drink

# Code Formatting

```
@left      | | = 0  
@top       | | = 0  
@width     | | = 1.0  
@height    | | = 0
```

## Code Formatting

```
double character: 'something weird',  
      stateMask: CTRL | modifier,  
      KeyCode:   character.downcase.ord
```

# 80 character width limit

```

25 | def initialize(blk)
26 |   @block = blk
27 | end
28 |
29 | # implemented by subclasses
30 | def key_pressed(event)
31 | end
32 |
33 | def key_released(event)
34 | end
35 |
36 | private
37 |
38 | def handle_key_event(event)
39 |   modifiers = modifier_keys(event)
40 |   character = character_key(event)
41 |   key_string = modifiers + character
42 |   key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
43 |   eval_block key_string unless character.empty?
44 | end
45 |
46 | def eval_block(key_string)
47 |   @block.call key_string
48 | end
49 |
50 | def modifier_keys(event)
51 |   modifier_keys = ''
52 |   modifier_keys += 'control_' if control?(event)
53 |   modifier_keys += 'shift_' if shift?(event) && special_key?(event)
54 |   modifier_keys += 'alt_' if alt?(event)
55 |   modifier_keys
56 | end
57 |
58 | def alt?(event)
59 |   is_this_modifier_key?(event, ::SwT::SWT::ALT)
60 | end
61 |
62 | # NOTE: state_mask and key_code error for me so the java version is used
63 | def is_this_modifier_key?(event, key)
64 |   (event.stateMask & key) == key
65 | end
66 |
67 | def shift?(event)
68 |   is_this_modifier_key?(event, ::SwT::SWT::SHIFT)
69 | end
70 |
71 | def control?(event)
72 |   is_this_modifier_key?(event, ::SwT::SWT::CTRL)
73 | end

```

```

70 | it ':alt_/' do
71 |   test_alt_character_press '/'
72 | end
73 |
74 | it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
75 |   block.should_receive(:call).with(':alt_/')
76 |   event = double character: '÷'.ord,
77 |                 stateMask: ALT,
78 |                 keyCode: '\/'.ord
79 |   subject.key_pressed(event)
80 | end
81 | end
82 |
83 | describe 'works with the ctrl key pressed such as' do
84 |   def test_ctrl_character_press(character, modifier = 0)
85 |     result_char = ('control_' + character).to_sym
86 |     block.should_receive(:call).with(result_char)
87 |     event = double character: 'something weird like \x00',
88 |                   stateMask: CTRL | modifier,
89 |                   keyCode: character.downcase.ord
90 |     subject.key_pressed(event)
91 |   end
92 |
93 |   it ':ctrl_a' do
94 |     test_ctrl_character_press 'a'
95 |   end
96 |
97 |   it 'ctrl_z' do
98 |     test_ctrl_character_press 'z'
99 |   end
100 |
101 | describe 'and if we add the shift key' do
102 |   it ':ctrl_A' do
103 |     test_ctrl_character_press 'A', SHIFT
104 |   end
105 |
106 |   it ':ctrl_Z' do
107 |     test_ctrl_character_press 'Z', SHIFT
108 |   end
109 | end
110 |
111 | describe 'works with shift combined with alt yielding capital letters' do
112 |   def test_alt_shift_character_press(character)
113 |     test_alt_character_press(character, SHIFT)
114 |   end
115 |
116 |   it ':alt_A' do
117 |     test_alt_shift_character_press 'A'
118 |   end
119 |
120 |   it ':alt_Z' do
121 |     test_alt_shift_character_press 'Z'

```

# 80 character width limit

```

26 | @block = blk
27 | end
28 |
29 | # implemented by subclasses
30 | def key_pressed(event)
31 | end
32 |
33 | def key_released(event)
34 | end
35 |
36 | private
37 |
38 | def handle_key_event(event)
39 |   modifiers = modifier_keys(event)
40 |   character = character_key(event)
41 |   key_string = modifiers + character
42 |   key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
43 |   eval_block key_string unless character.empty?
44 | end
45 |
46 | def eval_block(key_string)
47 |   @block.call key_string
48 | end
49 |
50 | def modifier_keys(event)
51 |   modifier_keys = ''
52 |   modifier_keys += 'control_' if control?(event)
53 |   modifier_keys += 'shift_' if shift?(event) && special_key?(event)
54 |   modifier_keys += 'alt_' if alt?(event)
55 |   modifier_keys
56 | end
57 |
58 | def alt?(event)
59 |   is_this_modifier_key?(event, ::SwT::SWT::ALT)
60 | end
61 |
62 | # NOTE: state_mask and key_code error for me so the java version is used
63 | def is_this_modifier_key?(event, key)
64 |   (event.stateMask & key) == key
65 | end
66 |
67 | def shift?(event)
68 |   is_this_modifier_key?(event, ::SwT::SWT::SHIFT)
69 | end
70 |
71 | def control?(event)
72 |   is_this_modifier_key?(event, ::SwT::SWT::CTRL)
73 | end
74 |

```

```

70 | it ':alt/' do
71 |   test_alt_character_press '/'
72 | end
73 |
74 | it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
75 |   block.should_receive(:call).with(:'alt_/')
76 |   event = double character: '÷'.ord,
77 |                 stateMask: ALT,
78 |                 keyCode: '/' .ord
79 |   subject.key_pressed(event)
80 | end
81 | end
82 |
83 | describe 'works with the ctrl key pressed such as' do
84 |   def test_ctrl_character_press(character, modifier = 0)
85 |     result_char = ('control_' + character).to_sym
86 |     block.should_receive(:call).with(result_char)
87 |     event = double character: 'something weird like \x00',
88 |                   stateMask: CTRL | modifier,
89 |                   keyCode: character.downcase.ord
90 |     subject.key_pressed(event)
91 |   end
92 |
93 | it ':ctrl_a' do
94 |   test_ctrl_character_press 'a'
95 | end
96 |
97 | it 'ctrl_z' do
98 |   test_ctrl_character_press 'z'
99 | end
100 |
101 | describe 'and if we add the shift key' do
102 |   it ':ctrl_A' do
103 |     test_ctrl_character_press 'A', SHIFT
104 |   end
105 |
106 |   it ':ctrl_Z' do
107 |     test_ctrl_character_press 'Z', SHIFT
108 |   end
109 | end
110 | end
111 |
112 | describe 'works with shift combined with alt yielding capital letters' do
113 |   def test_alt_shift_character_press(character)
114 |     test_alt_character_press(character, SHIFT)
115 |   end
116 |
117 |   it ':alt_A' do
118 |     test_alt_shift_character_press 'A'
119 |   end
120 |
121 |   it ':alt_Z' do
122 |     test_alt_shift_character_press 'Z'

```

# 80 character width limit

```

25 | def initialize(blk)
26 |   @block = blk
27 | end
28 |
29 | # implemented by subclasses
30 | def key_pressed(event)
31 | end
32 |
33 | def key_released(event)
34 | end
35 |
36 | private
37 |
38 | def handle_key_event(event)
39 |   modifiers = modifier_keys(event)
40 |   character = character_key(event)
41 |   key_string = modifiers + character
42 |   key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
43 |   eval_block key_string unless character.empty?
44 | end
45 |
46 | def eval_block(key_string)
47 |   @block.call key_string
48 | end
49 |
50 | def modifier_keys(event)
51 |   modifier_keys = ''
52 |   modifier_keys += 'control_' if control?(event)
53 |   modifier_keys += 'shift_' if shift?(event) && special_key?(event)
54 |   modifier_keys += 'alt_' if alt?(event)
55 |   modifier_keys
56 | end
57 |
58 | def alt?(event)
59 |   is_this_modifier_key?(event, ::Swt::SWT::ALT)
60 | end
61 |
62 | # NOTE: state_mask and key_code error for me so the java version is used
63 | def is_this_modifier_key?(event, key)
64 |   (event.stateMask & key) == key
65 | end
66 |
67 | def shift?(event)
68 |   is_this_modifier_key?(event, ::Swt::SWT::SHIFT)
69 | end
70 |
71 | def control?(event)
72 |   is_this_modifier_key?(event, ::Swt::SWT::CTRL)
73 | end
74 |

```

```

70 | it ':alt/' do
71 |   test_alt_character_press '/'
72 | end
73 |
74 | it 'works with what Macs seem to produce for opt + / (should be alt_/' do
75 |   block.should_receive(:call).with(':alt_/'
76 |     event = double character: '÷'.ord,
77 |                   stateMask: ALT,
78 |                   keyCode: '/'
79 |   subject.key_pressed(event)
80 | end
81 | end
82 |
83 | describe 'works with the ctrl key pressed such as' do
84 |   def test_ctrl_character_press(character, modifier = 0)
85 |     result_char = ('control_' + character).to_sym
86 |     block.should_receive(:call).with(result_char)
87 |     event = double character: 'something weird like \x00',
88 |                   stateMask: CTRL | modifier,
89 |                   keyCode: character.downcase.ord
90 |     subject.key_pressed(event)
91 |   end
92 |
93 | it ':ctrl_a' do
94 |   test_ctrl_character_press 'a'
95 | end
96 |
97 | it 'ctrl_z' do
98 |   test_ctrl_character_press 'z'
99 | end
100 |
101 | describe 'and if we add the shift key' do
102 |   it ':ctrl_A' do
103 |     test_ctrl_character_press 'A', SHIFT
104 |   end
105 |
106 |   it ':ctrl_Z' do
107 |     test_ctrl_character_press 'Z', SHIFT
108 |   end
109 | end
110 | end
111 |
112 | describe 'works with shift combined with alt yielding capital letters' do
113 |   def test_alt_shift_character_press(character)
114 |     test_alt_character_press(character, SHIFT)
115 |   end
116 |
117 |   it ':alt_A' do
118 |     test_alt_shift_character_press 'A'
119 |   end
120 | end
121 |
122 |   it ':alt_Z' do
123 |     test_alt_shift_character_press 'Z'

```

# 80 character width limit

```

25  def initialize(blk)
26    @block = blk
27  end
28
29  # implemented by subclasses
30  def key_pressed(event)
31  end
32
33  def key_released(event)
34  end
35
36  private
37
38  def handle_key_event(event)
39    modifiers = modifier_keys(event)
40    character = character_key(event)
41    key_string = modifiers + character
42    key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
43    eval_block key_string unless character.empty?
44  end
45
46  def eval_block(key_string)
47    @block.call key_string
48  end
49
50  def modifier_keys(event)
51    modifier_keys = ''
52    modifier_keys += 'control_' if control?(event)
53    modifier_keys += 'shift_' if shift?(event) && special_key?(event)
54    modifier_keys += 'alt_' if alt?(event)
55  end
56  end
57
58  def alt?(event)
59    is_this_modifier_key?(event, ::SwT::SWT::ALT)
60  end
61
62  # NOTE: state_mask and key_code error for me so the java version is used
63  def is_this_modifier_key?(event, key)
64    (event.stateMask & key) == key
65  end
66  end
67
68  def shift?(event)
69    is_this_modifier_key?(event, ::SwT::SWT::SHIFT)
70  end
71
72  def control?(event)
73    is_this_modifier_key?(event, ::SwT::SWT::CTRL)
74  end

```

```

70  it ':alt/' do
71    test_alt_character_press '/'
72  end
73
74  it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
75
76    event = double character: '÷'.ord,
77                  stateMask: ALT,
78                  keyCode: '/' .ord
79    subject.key_pressed(event)
80  end
81  end
82
83  describe 'works with the ctrl key pressed such as' do
84    def test_ctrl_character_press(character, modifier = 0)
85      result_char = ('control_' + character).to_sym
86      block.should_receive(:call).with(result_char)
87      event = double character: 'something weird like \x00',
88                  stateMask: CTRL | modifier,
89                  keyCode: character.downcase.ord
90      subject.key_pressed(event)
91    end
92  end
93
94  it ':ctrl_a' do
95    test_ctrl_character_press 'a'
96  end
97
98  it 'ctrl_z' do
99    test_ctrl_character_press 'z'
100  end
101
102  describe 'and if we add the shift key' do
103    it ':ctrl_A' do
104      test_ctrl_character_press 'A', SHIFT
105    end
106
107    it ':ctrl_Z' do
108      test_ctrl_character_press 'Z', SHIFT
109    end
110  end
111
112  describe 'works with shift combined with alt yielding capital letters' do
113    def test_alt_shift_character_press(character)
114      test_alt_character_press(character, SHIFT)
115    end
116  end
117
118  it ':alt_A' do
119    test_alt_shift_character_press 'A'
120  end
121
122  it ':alt_Z' do
123    test_alt_shift_character_press 'Z'

```

# 80 character width limit

```

25 | def initialize(blk)
26 |   @block = blk
27 | end
28 |
29 | # implemented by subclasses
30 | def key_pressed(event)
31 | end
32 |
33 | def key_released(event)
34 | end
35 |
36 | private
37 |
38 | def handle_key_event(event)
39 |   modifiers = modifier_keys(event)
40 |   character = character_key(event)
41 |   key_string = modifiers + character
42 |   key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
43 |   eval_block key_string unless character.empty?
44 | end
45 |
46 | def eval_block(key_string)
47 |   @block.call key_string
48 | end
49 |
50 | def modifier_keys(event)
51 |   modifier_keys = ''
52 |   modifier_keys += 'control_' if control?(event)
53 |   modifier_keys += 'shift_' if shift?(event) && special_key?(event)
54 |   modifier_keys += 'alt_' if alt?(event)
55 |   modifier_keys
56 | end
57 |
58 | def alt?(event)
59 |   is_this_modifier_key?(event, ::SwT::SWT::ALT)
60 | end
61 |
62 | # NOTE: state_mask and key_code error for me so the java version is used
63 | def is_this_modifier_key?(event, key)
64 |   (event.stateMask & key) == key
65 | end
66 |
67 | def shift?(event)
68 |   is_this_modifier_key?(event, ::SwT::SWT::SHIFT)
69 | end
70 |
71 | def control?(event)
72 |   is_this_modifier_key?(event, ::SwT::SWT::CTRL)
73 | end
74 |

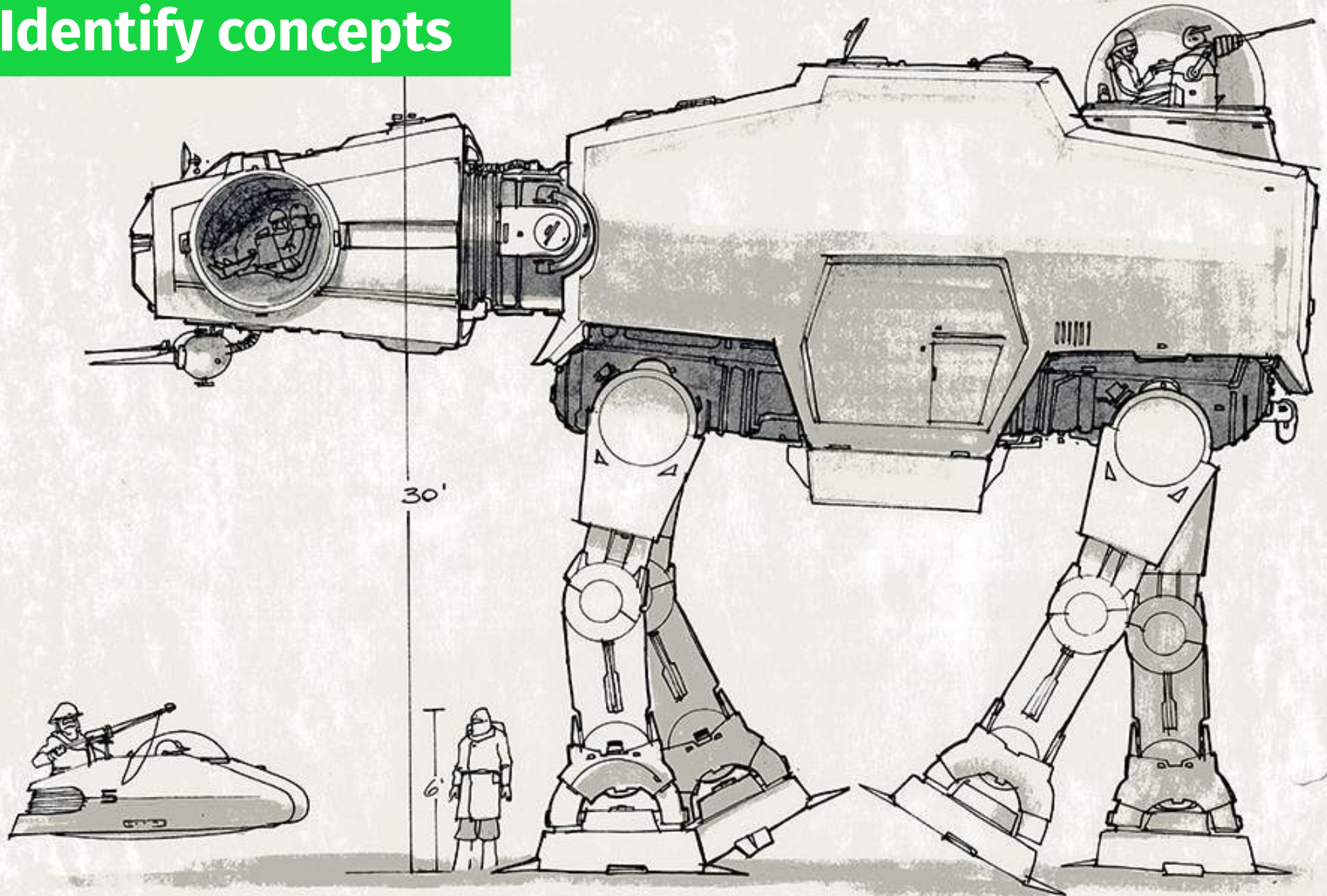
```

```

70 | it ':alt_/' do
71 |   test_alt_character_press '/'
72 | end
73 |
74 | it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
75 |   block.should_receive(:call).with(':alt_/')
76 |   event = double character: '÷'.ord,
77 |                 stateMask: ALT,
78 |                 keyCode: '\/'.ord
79 |   subject.key_pressed(event)
80 | end
81 | end
82 |
83 | describe 'works with the ctrl key pressed such as' do
84 |   def test_ctrl_character_press(character, modifier = 0)
85 |     result_char = ('control_' + character).to_sym
86 |     block.should_receive(:call).with(result_char)
87 |     event = double character: 'something weird like \x00',
88 |                   stateMask: CTRL | modifier,
89 |                   keyCode: character.downcase.ord
90 |     subject.key_pressed(event)
91 |   end
92 |
93 |   it ':ctrl_a' do
94 |     test_ctrl_character_press 'a'
95 |   end
96 |
97 |   it 'ctrl_z' do
98 |     test_ctrl_character_press 'z'
99 |   end
100 |
101 |   describe 'and if we add the shift key' do
102 |     it ':ctrl_A' do
103 |       test_ctrl_character_press 'A', SHIFT
104 |     end
105 |
106 |     it ':ctrl_Z' do
107 |       test_ctrl_character_press 'Z', SHIFT
108 |     end
109 |   end
110 | end
111 |
112 | describe 'works with shift combined with alt yielding capital letters' do
113 |   def test_alt_shift_character_press(character)
114 |     test_alt_character_press(character, SHIFT)
115 |   end
116 |
117 |   it ':alt_A' do
118 |     test_alt_shift_character_press 'A'
119 |   end
120 |
121 |   it ':alt_Z' do
122 |     test_alt_shift_character_press 'Z'

```

# Identify concepts



Johnston 0135 1/79

FOR SCALE ONLY

WONDERFULI

**D**on ' t  
**R**epeat  
**Y**ourself

**Don't**  
**Repeat**  
**Yourself**  
**(In Concepts)**

# Virtues



**Simplicity**

The **least powerful** construct

**Explicit** vs **Implicit**

# Immutable Data

```
person = Person.new(attributes)  
do_something(person)  
insert_in_db(person)
```

# Immutable Data

```
person = Person.new(attributes)
person = do_something(person)
insert_in_db(person)
```

# **Transformation** of data

**Reads like a book**

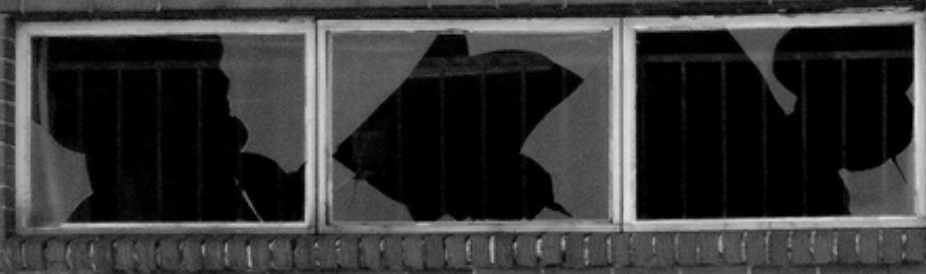


# Nurturing a code base



Code bases **deteriorate**

**No broken windows!**



IN  
KID  
KID





6032



**REFACTOR**



**ALL THE THINGS**

**YOU CAN'T**





**Magical time?**

# The boyscout rule



# Opportunistic Refactoring

# Confident Tests

**80%** Code Coverage

**20% never executed**



Jason Clark commented on an outdated diff on Sep 29, 2014

Hide outdated diff

spec/shoes/app\_spec.rb

```

@@ -389,6 +389,20 @@
 389   389       expect(subject.instance_variable_get(:@_additional_context_)).to be_r
 390   390       end
 391   391       end
+ 392   +
+ 393   + describe '.new_dsl_method (for'
+ 394   +   before :each do
+ 395   +     Shoes::App.new_dsl_method
+ 396   +     # noop
+ 397   +   end
+ 398   + end
+ 399   +
+ 400   + [Shoes::App, Shoes::URL, Shoes::Widget].each do |klass|

```

# Code Review Culture

Jason Clark added a note on Sep 29, 2014 Owner

If we add another class to `METHOD_SUBSCRIBER`, we won't be running the tests on them unless someone remembers to update this. Is it too self-referential for this to use `[Shoes::App] + METHOD_SUBSCRIBERS` as the basis for the `each`?

PragTob added a note on Sep 29, 2014 Owner

It most likely is not - thanks for the input!  
  
Might as well stay like this if I add the subscription mechanism, though (because then we don't have the explicit list anywhere anymore).

Add a line note



PragTob added some commits on Oct 1, 2014

- Mechanism to subscribe classes/modules to DSL methods ... f2d7fc7
- Moved DELEGATE\_METHODS specs down to form a logical unit ✓ 878109a
- Open class forward references no longer needed ✓ ffae2fd



Jason Clark commented on an outdated diff on Sep 29, 2014

Hide outdated diff

spec/shoes/app\_spec.rb

```

-389,6 +389,20 @@
389 389     expect(subject.instance_variable_get(:@_additional_context_)).to be_r
390 390     end
391 391     end

```

Always the code, never the person

```

396 +     # noop
397 +     end
398 +     end
399 +
400 +     [Shoes::App, Shoes::URL, Shoes::Widget].each do |klazz|

```

Jason Clark added a note on Sep 29, 2014 Owner

If we add another class to `METHOD_SUBSCRIBER`, we won't be running the tests on them unless someone remembers to update this. Is it too self-referential for this to use `[Shoes::App] + METHOD_SUBSCRIBERS` as the basis for the `each`?

PragTob added a note on Sep 29, 2014 Owner

It most likely is not - thanks for the input!  
Might as well stay like this if I add the subscription mechanism, though (because then we don't have the explicit list anywhere anymore).

Add a line note



PragTob added some commits on Oct 1, 2014

- Mechanism to subscribe classes/modules to DSL methods ... f2d7fc7
- Moved DELEGATE\_METHODS specs down to form a logical unit ✓ 878109a
- Open class forward references no longer needed ✓ ffae2fd



vectorclark commented on an outdated diff on Sep 29, 2014

Hide outdated diff

spec/shoes/app\_spec.rb

```

-389 -389 expect(subject.instance_variable_get(:@_additional_context_)).to be_r
-390 -390 end
-391 -391 end

```

Always the code, never the person

```

+396 + # noop
+397 + end
+398 + end
+399 +
+400 + [Shoes::App, Shoes::URL, S

```

Say something positive



vectorclark added a note on Sep 29, 2014

Owner

If we add another class to `METHOD_SUBSCRIBER`, we won't be running the tests on them unless someone remembers to update this. Is it too self-referential for this to use `[Shoes::App] + METHOD_SUBSCRIBERS` as the basis for the `each`?



PragTob added a note on Sep 29, 2014

Owner

It most likely is not - thanks for the input!  
Might as well stay like this if I add the subscription mechanism, though (because then we don't have the explicit list anywhere anymore).

Add a line note



PragTob added some commits on Oct 1, 2014

- Mechanism to subscribe classes/modules to DSL methods ... f2d7fc7
- Moved DELEGATE\_METHODS specs down to form a logical unit ✓ 878109a
- Open class forward references no longer needed ✓ ffae2fd



vectorclark commented on an outdated diff on Sep 29, 2014

Hide outdated diff

spec/shoes/app\_spec.rb

```

-389,6 +389,20 @@
389 389     expect(subject.instance_variable_get(:@_additional_context_)).to be_r
390 390     end
391 391     end

```

Always the code, never the person

```

396 + # noop
397 +     end
398 +     end
399 +
400 + [Shoes::App, Shoes::URL, S

```

Say something positive

vectorclark added a note on Sep 29, 2014

If we add another class to METHOD\_SUBSCRIBER, w someone remembers to update this. Is it too self-r METHOD\_SUBSCRIBERS as the basis for the each ?

Use automated linters

PragTob added a note on Sep 29, 2014

Owner

It most likely is not - thanks for the input!

Might as well stay like this if I add the subscription mechanism, though (because then we don't have the explicit list anywhere anymore).

Add a line note

PragTob added some commits on Oct 1, 2014

- Mechanism to subscribe classes/modules to DSL methods ... f2d7fc7
- Moved DELEGATE\_METHODS specs down to form a logical unit ✓ 878109a
- Open class forward references no longer needed ✓ ffae2fd



Jason Clark commented on an outdated diff on Sep 29, 2014

Hide outdated diff

spec/shoes/app\_spec.rb

```

-389,6 +389,20 @@
389 389 expect(subject.instance_variable_get(:@_additional_context_)).to be_r
390 390 end
391 391 end

```

Always the code, never the person

```

396 + # noop
397 + end
398 + end
399 +
400 + [Shoes::App, Shoes::URL, S

```

Say something positive

Jason Clark added a note on Sep 29, 2014

If we add another class to METHOD\_SUBSCRIBER, will someone remember to update this. Is it too self-referential to use METHOD\_SUBSCRIBERS as the basis for the each?

Use automated linters

PragTob added a note on Sep 29, 2014

It most likely is not - thanks for the input!  
Might as well stay like this if I add the subscription mechanism, though (because then we don't have the explicit list anywhere anymore).

Everyone reviews

Add a line note



PragTob added some commits on Oct 1, 2014

- Mechanism to subscribe classes/modules to DSL methods ... f2d7fc7
- Moved DELEGATE\_METHODS specs down to form a logical unit ✓ 878109a
- Open class forward references no longer needed ✓ ffae2fd

```
spec/shoes/app_spec.rb

-389 -389 expect(subject.instance_variable_get(:@_additional_context_)).to be_r
-390 -390 end
-391 -391 end

+396 + # noop
+397 + end
+398 + end
+399 +
+400 + [Shoes::App, Shoes::URL, S
```

**Always the code, never the person**

**Say something positive**

**JasonClark** added a note on Sep 29, 2014

If we add another class to `METHOD_SUBSCRIBER`, will someone remember to update this. Is it too self-referential to use `METHOD_SUBSCRIBERS` as the basis for the `each`?

**Use automated linters**

**PragTob** added a note on Sep 29, 2014

It most likely is not - thanks for the input!

Might as well stay like this if I add the subscription mechanism, though (because then we don't have the explicit list anywhere anymore).

**Everyone reviews**

**Pairing Review**

Add a line note

- PragTob** added some commits on Oct 1, 2014
- Mechanism to subscribe classes/modules to DSL methods ... f2d7fc7
  - Moved DELEGATE\_METHODS specs down to form a logical unit ✓ 878109a
  - Open class forward references no longer needed ✓ ffae2fd

# Pair Programming





## The power of magic hats



# Reaping the benefits



# When to break the rules



*If you still like your code from **two**  
**years ago**,  
then you are not **learning** fast  
enough.*

Enjoy writing **readable code!**

Tobias Pfeiffer  
@PragTob  
pragtob.info



**L I E F E R Y**

RUG::B



# Ruby Implementations



elixir



LIEFERY

Open Source

Testing



Rails Girls Berlin

benchmarking



# Sources

- The Pragmatic Programmer
- Smalltalk Best Practice Patterns
- Clean Code
- Practical Object Oriented Design in Ruby

# Photo Credit

- <http://officeimg.vo.msecnd.net/en-us/images/MP900439313.jpg>
- <http://officeimg.vo.msecnd.net/en-us/images/MC900021328.wmf>
- [http://www.osnews.com/story/19266/WTFs\\_m](http://www.osnews.com/story/19266/WTFs_m)
- (CC BY-SA 2.0)
  - <http://www.flickr.com/photos/83633410@N07/7658272558/in/photostream/>
  - <http://www.flickr.com/photos/83633410@N07/7658165122/>
  - <https://www.flickr.com/photos/93425126@N00/313056379/>
- (CC BY-NC-ND 2.0)
  - <http://www.flickr.com/photos/andih/86577529/>
  - <http://www.flickr.com/photos/12584908@N08/3293117576/>
  - <http://www.flickr.com/photos/jasonparks/4525188865/>
  - <http://www.flickr.com/photos/20714221@N04/2293045156/>
  - <https://www.flickr.com/photos/eyewash/2603717864/>
  - [https://www.flickr.com/photos/stevie\\_gill/3950697539/](https://www.flickr.com/photos/stevie_gill/3950697539/)
  - <https://www.flickr.com/photos/randar/15787696685/>
- <http://www.flickr.com/photos/47833351@N02/5488791911/> (CC BY-ND 2.0)
- (CC BY 2.0)
  - [http://www.flickr.com/photos/barry\\_b/76055201/](http://www.flickr.com/photos/barry_b/76055201/)
  - <http://www.flickr.com/photos/25165196@N08/7725273678/>
  - <http://www.flickr.com/photos/29254399@N08/3187186308/>
  - <https://www.flickr.com/photos/garryknight/5650367750/>
  - <https://www.flickr.com/photos/alper/10742816123/>
- (CC BY-NC-SA 2.0)
  - <http://www.flickr.com/photos/dolescum/7380616658/>
  - <http://www.flickr.com/photos/antonkovalyov/5795281215/>
  - <http://www.flickr.com/photos/doug88888/2792209612/>
  - <https://www.flickr.com/photos/denverjjeffrey/4392418334/>
- (CC BY-NC 2.0)
  - <http://www.flickr.com/photos/37996583811@N01/5757983532/>
  - <http://www.flickr.com/photos/sevendead/5650065458/>
  - <https://www.flickr.com/photos/whitecatsg/3146092196/>