Critique Your Own Code

Geoffrey Grosenbach
peepcode.com

PeepCode
screencasts

Nature Shots
Playing in the P
Tulum Vacation
Friends
Gymnastics
Biking the Islands
Sun Valley
A Day at the Orchard
Rome
Trip to Tikal

apple.com

```
describe “Money”, ->

  beforeEach ->
    @m = new Money ‘$15.99’

  it “extracts dollar and cents”, ->
    (expect @m.cents).toEqual 1599

  it “formats as string”, ->
    (expect @m.toString()).toEqual ‘$15.99’
```

```
describe "Money", ->

  beforeEach ->
    @m = new Money '$15.99'

  it "extracts dollar and cents", ->
    (expect @m.cents).toEqual 1599

  it "formats as string", ->
    (expect @m.toString()).toEqual '$15.99'
```

Bugs

Readability

Checklist

Obey the Rules
55
Velominati

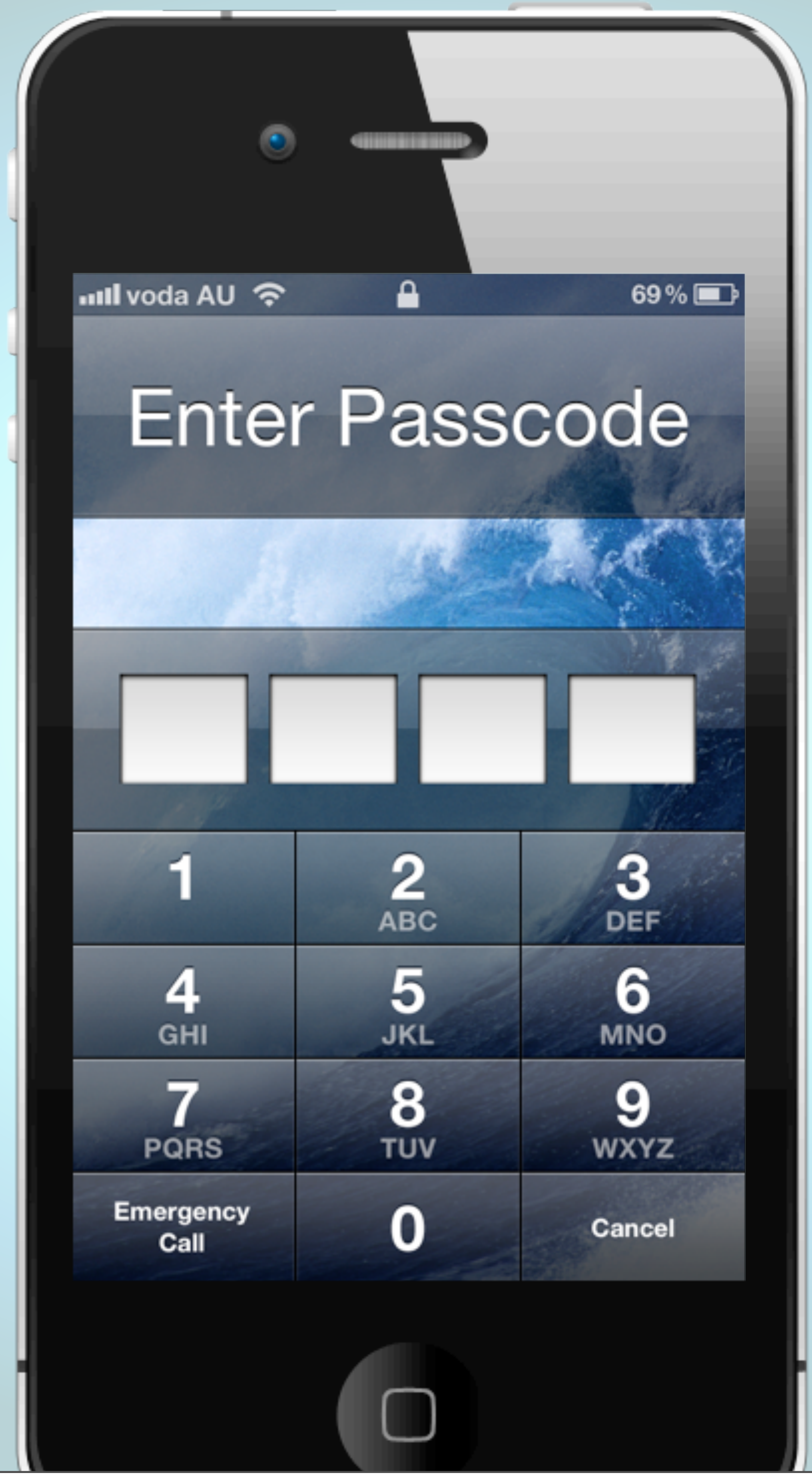
Bugs

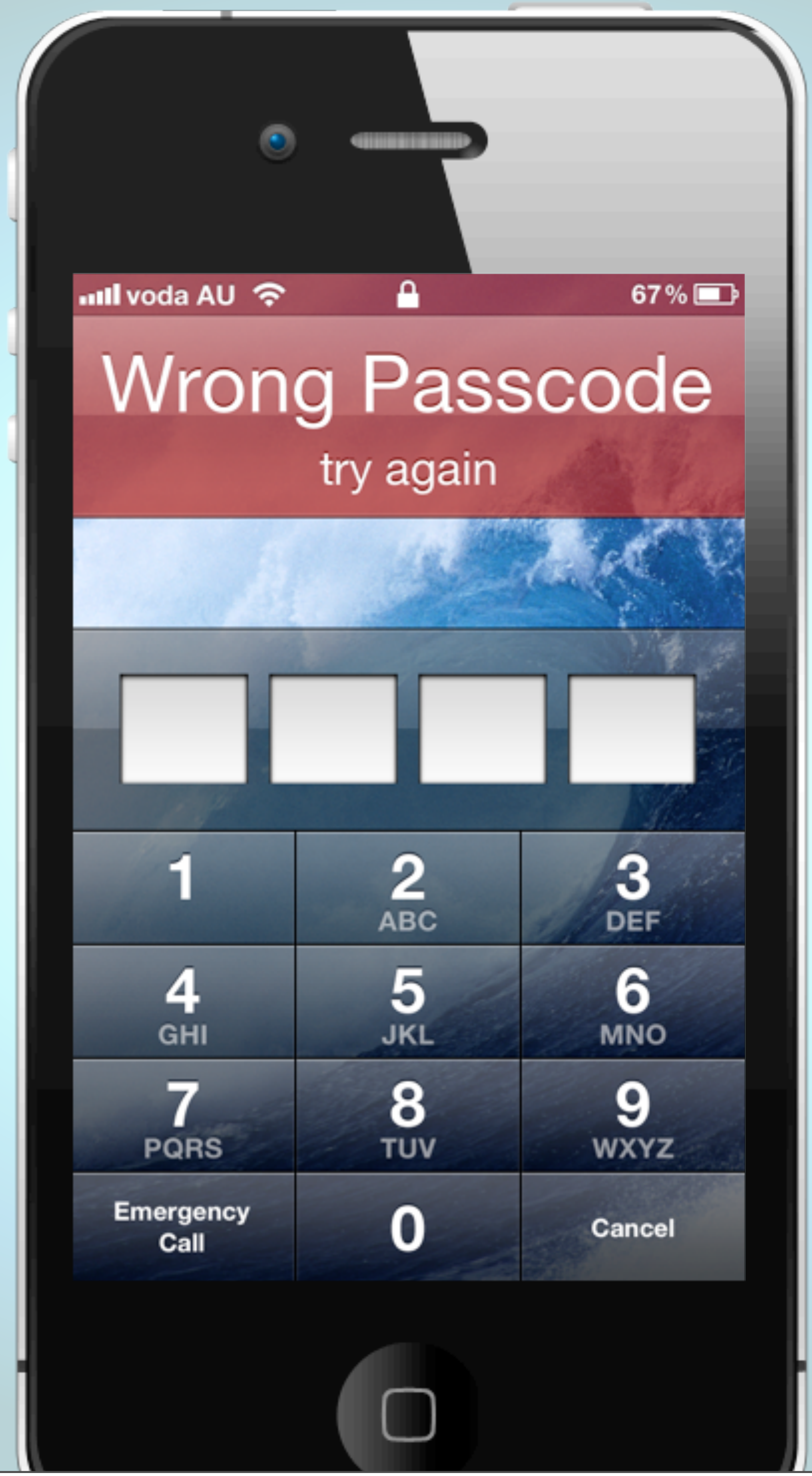
Unexpected things will happen.

#poopin

voda AU
69 %
15:20
Wednesday, November 30
slide to unlock

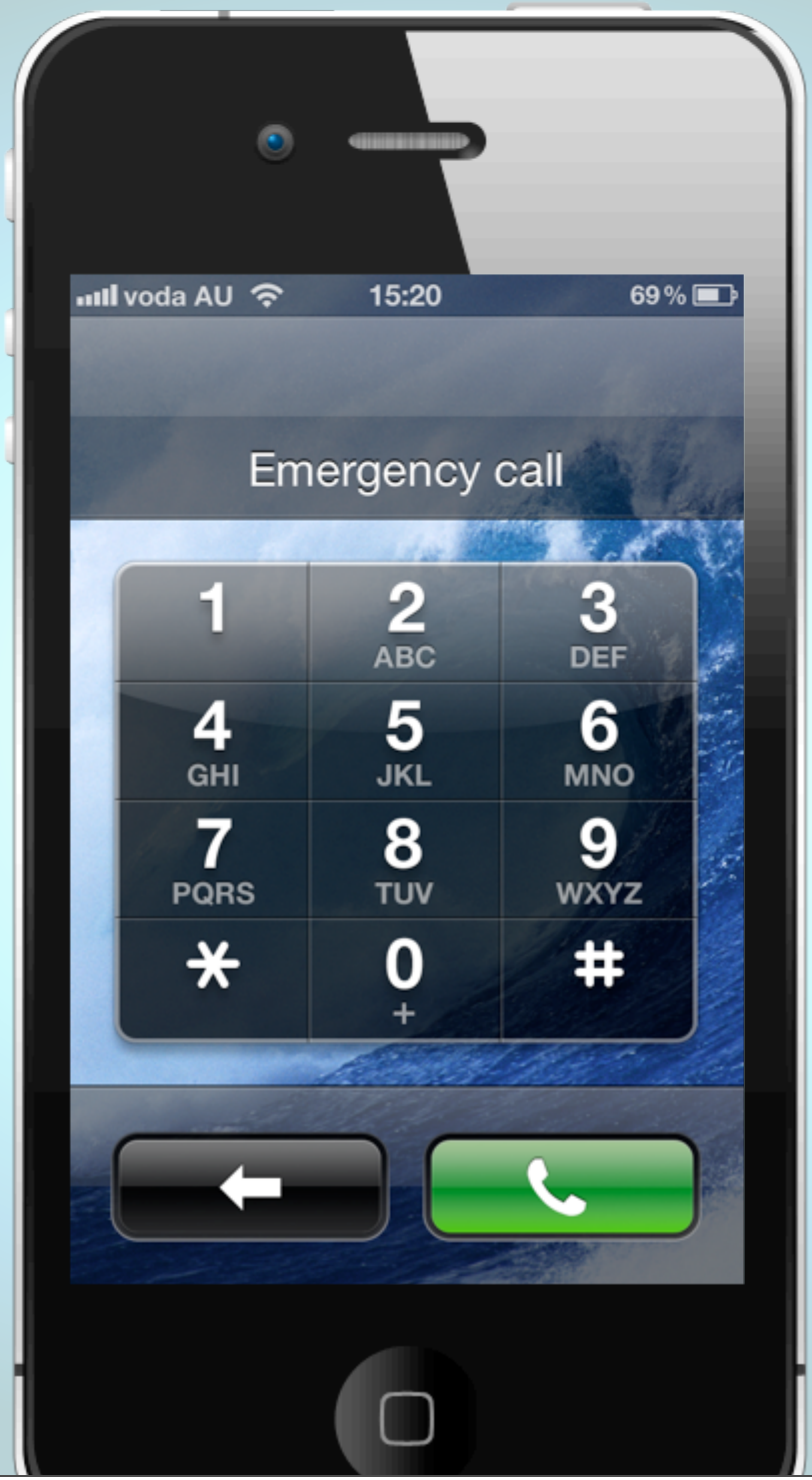
voda AU
69 %
15:20
Wednesday, November 30


voda AU
69 %
Enter Passcode
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
Emergency Call
0
Cancel



Wrong Passcode

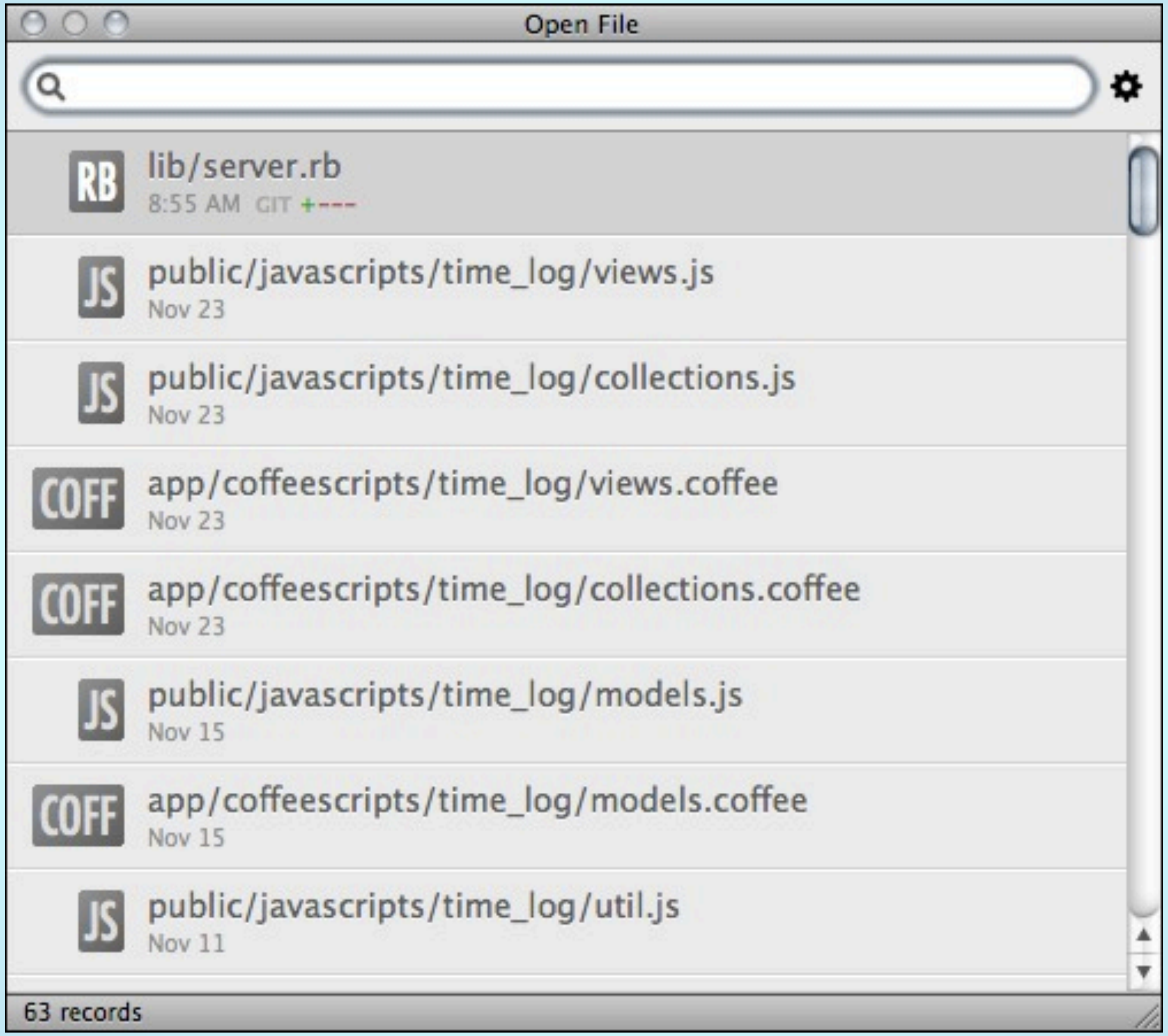
try again



voda AU
15:20
69 %
Emergency call
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
*
0 +
#

Opposite *v.* Unexpected

peepcode.com/products/peepopen



RB lib/server.rb
8:55 AM GIT +---

JS public/javascripts/time_log/views.js
Nov 23

JS public/javascripts/time_log/collections.js
Nov 23

COFF app/coffeescripts/time_log/views.coffee
Nov 23

COFF app/coffeescripts/time_log/collections.coffee
Nov 23

JS public/javascripts/time_log/models.js
Nov 15

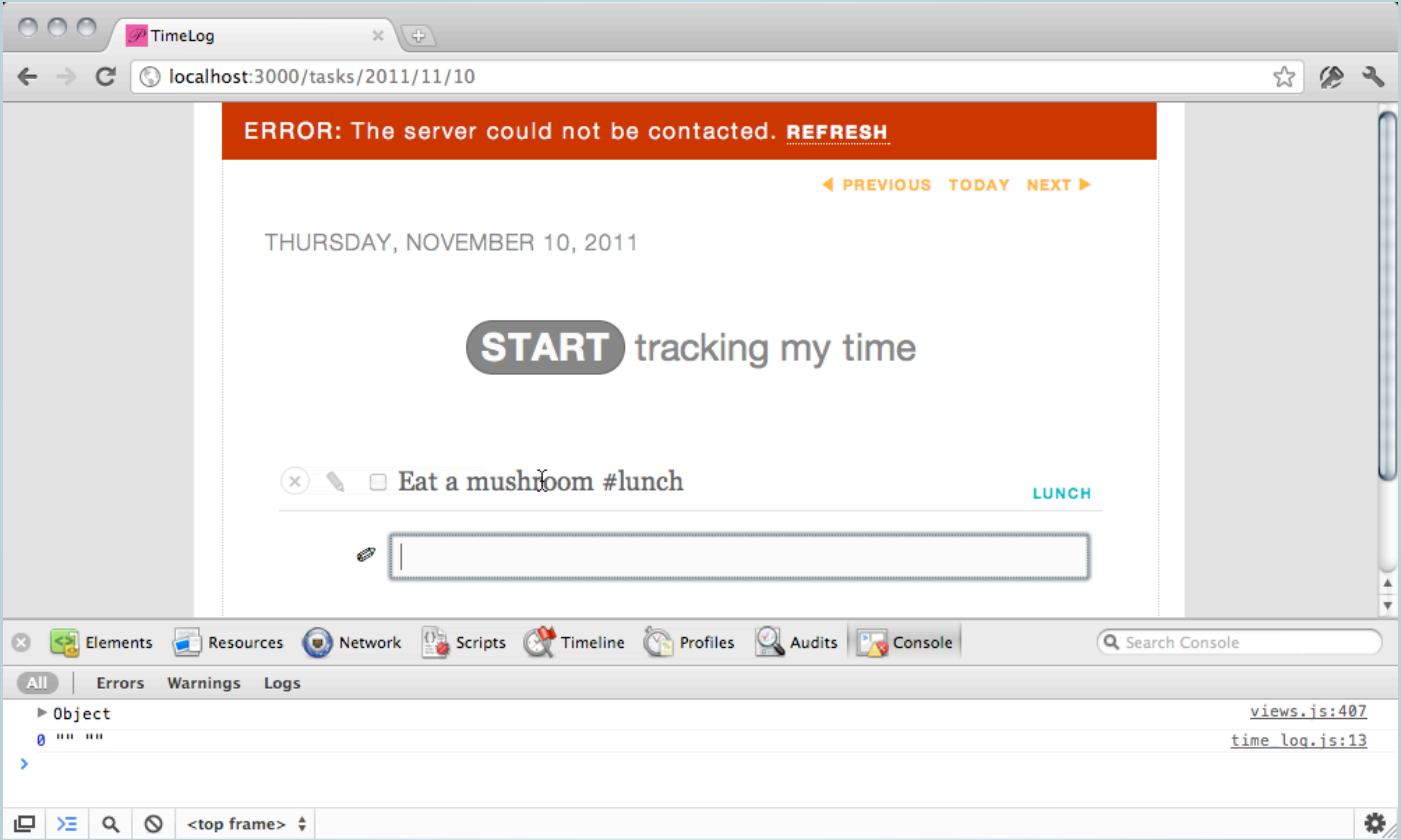
COFF app/coffeescripts/time_log/models.coffee
Nov 15

JS public/javascripts/time_log/util.js
Nov 11

63 records

peepcode.com/products/peepopen

Pass *v.* Fail *v.* Maybe?



ERROR: The server could not be contacted. REFRESH

THURSDAY, NOVEMBER 10, 2011

START tracking my time

□ Eat a mushroom #lunch LUNCH

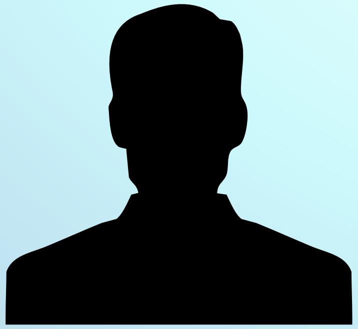
```
▶ Object                                                    views.js:407
0 "" ""                                                     time_log.js:13
>
```

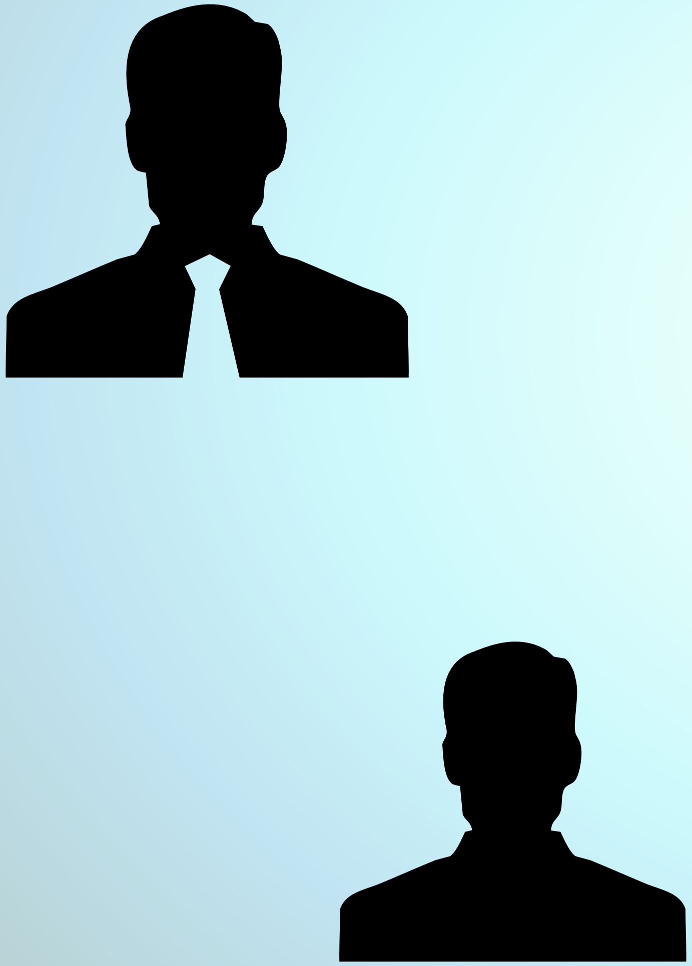
Automate the unexpected: fuzz testing, heckle.

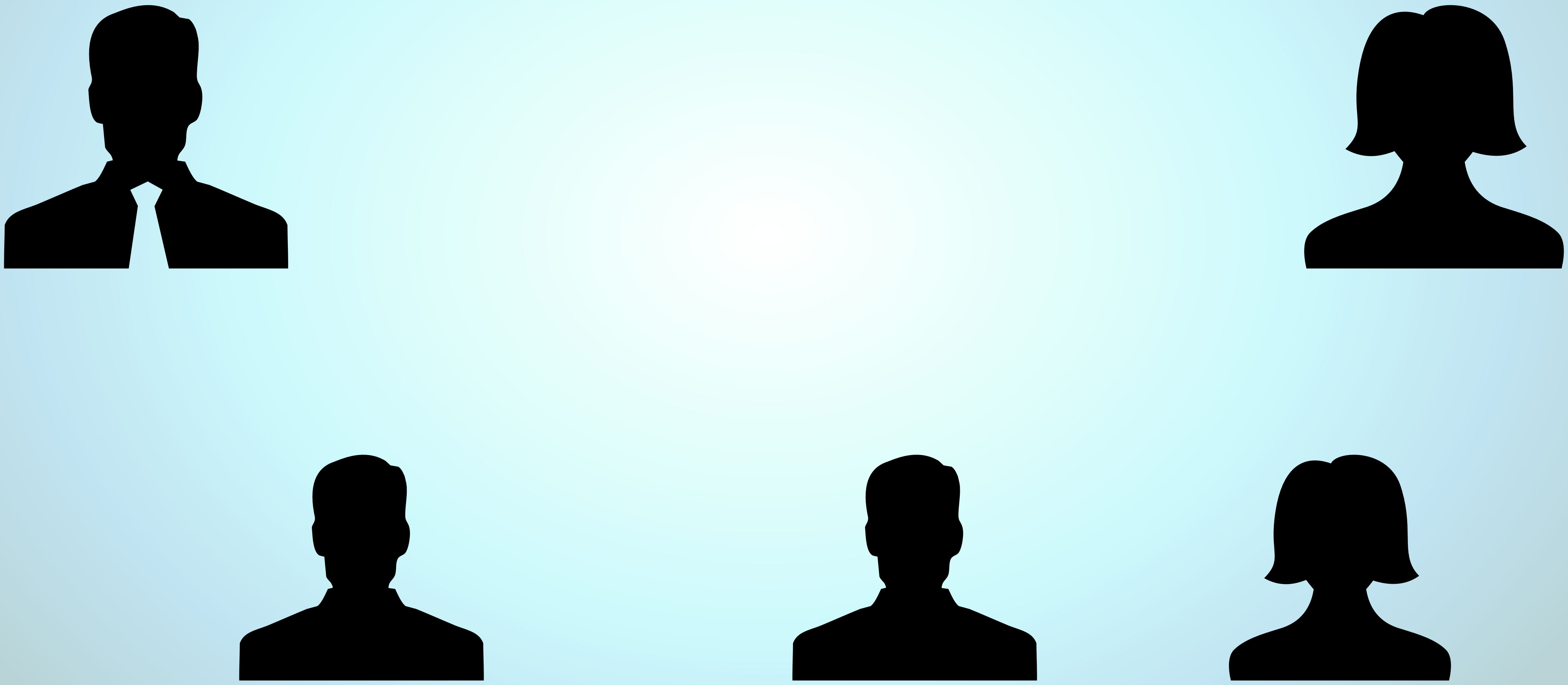
Readability

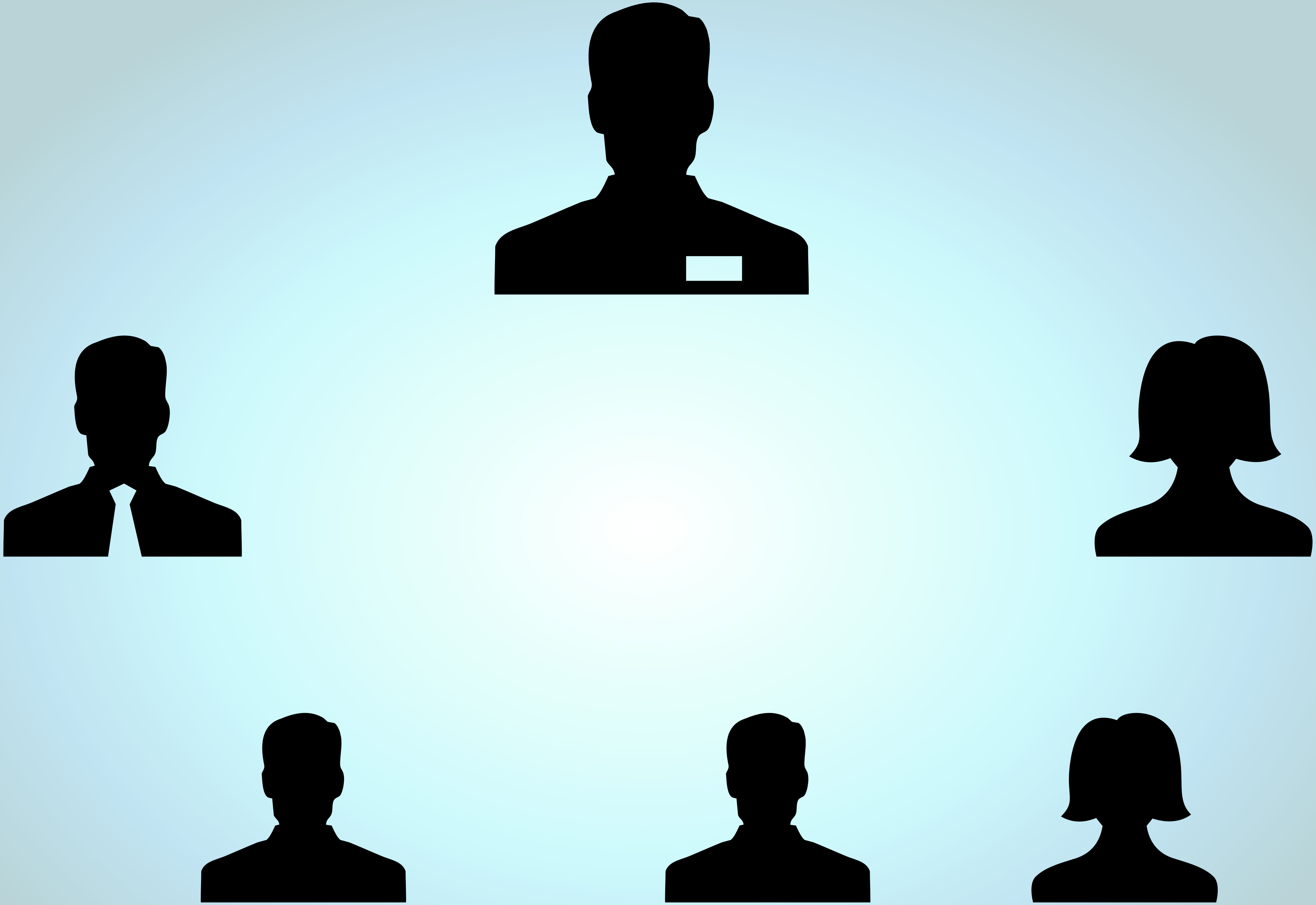
It’s easier to edit than to create.

But if you try hard enough, you can create something that can’t be edited.









Current
Me

Current Me

Future Me

```
loop do
  begin # ^c handler
    get_to_green
    if tainted? and not options[:no_full_after_failed] then
      rerun_all_tests
    else
      hook :all_good
    end
    wait_for_changes
  rescue Interrupt
    break if wants_to_quit
    reset
  end
end
hook :quit
```

autotest.rb

```
loop do
  begin # ^c handler
    get_to_green
    if tainted? and not options[:no_full_after_failed] then
      rerun_all_tests
    else
      hook :all_good
    end
    wait_for_changes
  rescue Interrupt
    break if wants_to_quit
    reset
  end
end
hook :quit
```

autotest.rb

```
loop do
  begin # ^c handler
    get_to_green
    if tainted? and not options[:no_full_after_failed] then
      rerun_all_tests
    else
      hook :all_good
    end
    wait_for_changes
  rescue Interrupt
    break if wants_to_quit
    reset
  end
end
hook :quit
```

autotest.rb

```
loop do
  begin # ^c handler
    get_to_green
    if tainted? and not options[:no_full_after_failed] then
      rerun_all_tests
    else
      hook :all_good
    end
    wait_for_changes
  rescue Interrupt
    break if wants_to_quit
    reset
  end
end
hook :quit
```

autotest.rb

```
loop do
  begin # ^c handler
    get_to_green
    if tainted? and not options[:no_full_after_failed] then
      rerun_all_tests
    else
      hook :all_good
    end
    wait_for_changes
  rescue Interrupt
    break if wants_to_quit
    reset
  end
end
hook :quit
```

autotest.rb

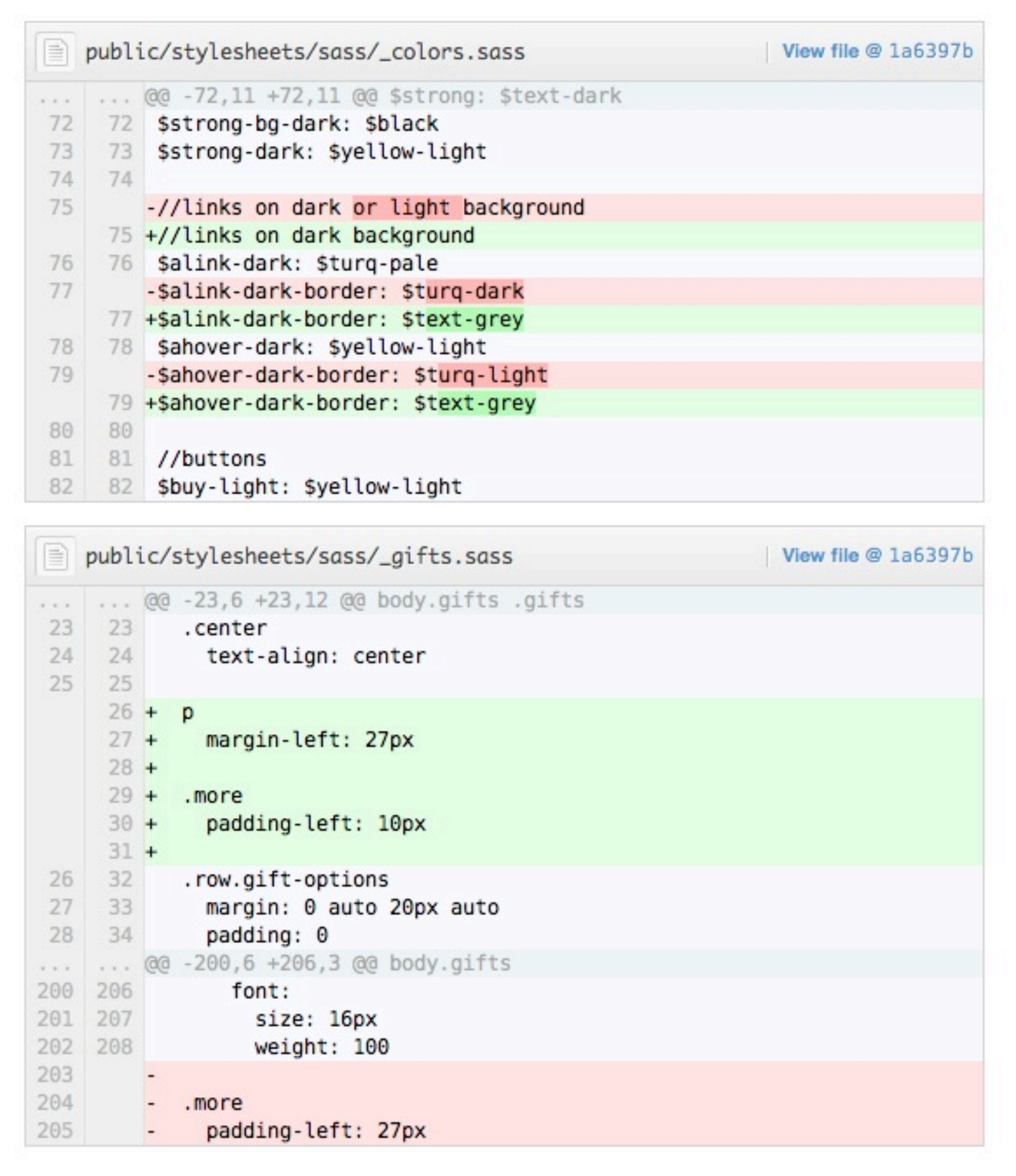
```
loop do
  begin # ^c handler
    get_to_green
    if tainted? and not options[:no_full_after_failed] then
      rerun_all_tests
    else
      hook :all_good
    end
    wait_for_changes
  rescue Interrupt
    break if wants_to_quit
    reset
  end
end
hook :quit
```

autotest.rb

```ruby
loop do
  begin # ^c handler
    get_to_green
    if tainted? and not options[:no_full_after_failed] then
      rerun_all_tests
    else
      hook :all_good
    end
    wait_for_changes
  rescue Interrupt
    break if wants_to_quit
    reset
  end
end
hook :quit
```

autotest.rb

The English language is a major hurdle to readability.



public/stylesheets/sass/_colors.sass

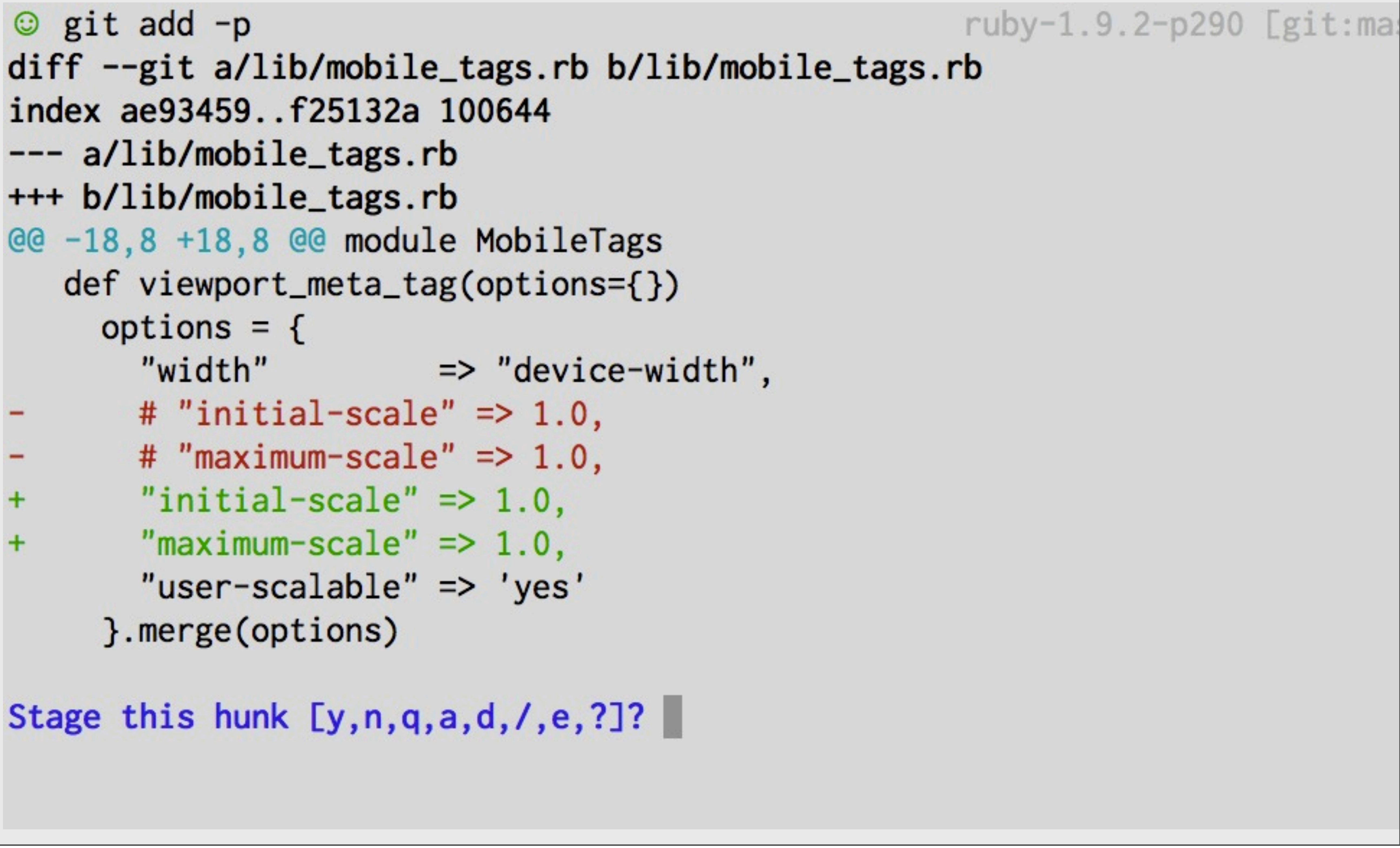
```diff
@@ -72,11 +72,11 @@ $strong: $text-dark
 $strong-bg-dark: $black
 $strong-dark: $yellow-light
 
-//links on dark or light background
+//links on dark background
 $alink-dark: $turq-pale
-$alink-dark-border: $turq-dark
+$alink-dark-border: $text-grey
 $ahover-dark: $yellow-light
-$ahover-dark-border: $turq-light
+$ahover-dark-border: $text-grey
 
 //buttons
 $buy-light: $yellow-light
```

public/stylesheets/sass/_gifts.sass

```diff
@@ -23,6 +23,12 @@ body.gifts .gifts
   .center
     text-align: center
 
+  p
+    margin-left: 27px
+
+  .more
+    padding-left: 10px
+
   .row.gift-options
     margin: 0 auto 20px auto
     padding: 0
@@ -200,6 +206,3 @@ body.gifts
       font:
         size: 16px
         weight: 100
-
-  .more
-    padding-left: 27px
```

```
git add -p
```

git add -p



```
☺ git add -p                                                  ruby-1.9.2-p290 [git:ma
diff --git a/lib/mobile_tags.rb b/lib/mobile_tags.rb
index ae93459..f25132a 100644
--- a/lib/mobile_tags.rb
+++ b/lib/mobile_tags.rb
@@ -18,8 +18,8 @@ module MobileTags
   def viewport_meta_tag(options={})
     options = {
       "width"         => "device-width",
-      # "initial-scale" => 1.0,
-      # "maximum-scale" => 1.0,
+      "initial-scale" => 1.0,
+      "maximum-scale" => 1.0,
       "user-scalable" => 'yes'
     }.merge(options)

Stage this hunk [y,n,q,a,d,/,e,?]? 
```

Sometimes hacks are necessary.

Checklist

- ☑ Do the tests pass?
- ☑ Is there any duplication?
- ☑ Does it express my intent?
- ☑ Are methods, variables, and classes named well?
- ☑ Does it use a minimum of classes and methods?

Rules of Simple Design, Kent Beck, *Clean Code*

☑ Are lines 80 characters wide at maximum?

- ☑ Are lines 80 characters wide at maximum?
- ☑ Does each method fit on one screen?

☑ Are lines 80 characters wide at maximum?

☑ Does each method fit on one screen?

☑ Is the code indented?

Use an automated source formatter

☑ Do method names describe the application?

☑ Do method names describe the application?

☑ Is punctuation and naming style consistent?

☑ Do I know of any bugs right now?

- ☑ Do I know of any bugs right now?
- ☑ Do I know of any hacks that I perpetrated?

- ☑ Do I know of any bugs right now?
- ☑ Do I know of any hacks that I perpetrated?
- ☑ Am I proud of this code?

☑ Could I use this code to teach someone else how to write programs in this language?

Get a code review

YOW

peepcode.com/c