# The Story of The Three Little Pigs

# Once upon a time...

```ruby
module ThreeLittlePigs
  module Chapters
    RSpec.describe OnceUponATime do
      let(:story) { Story.beginning }

      before { OnceUponATime.tell(story) }

      specify "there were three little pigs" do
        expect(story.first_pig).to be_a(Pig)
        expect(story.second_pig).to be_a(Pig)
        expect(story.third_pig).to be_a(Pig)
      end
    end
  end
end
```

```ruby
RSpec.describe PigsLiveWithTheirMother do
  let(:story) do
    Story.until_chapter(PigsLiveWithTheirMother)
  end
  let(:mother_pig) { story.mother_pig }
  # …

  before { PigsLiveWithTheirMother.tell(story) }

  specify "the little pigs had a mother" do
    expect(mother_pig).to be_a(Pig)
    expect(mother_pig.children).to \
      eq([first_pig, second_pig, third_pig])
  end
end
```

```ruby
RSpec.describe PigsLiveWithTheirMother do
  let(:story) do
    Story.until_chapter(PigsLiveWithTheirMother)
  end
  let(:mother_pig) { story.mother_pig }
  # …

  before { PigsLiveWithTheirMother.tell(story) }

  specify "the little pigs had a mother"
  specify "the pigs all lived together" do
    expect(mother_pig.house.occupants).to eq([
      mother_pig, first_pig,
      second_pig, third_pig
    ])
  end
end
```

```ruby
RSpec.describe PigsLeaveTheHouse do
  let(:story) { … }
  let(:mother_pig) { story.mother_pig }

  before { PigsLeaveTheHouse.tell(story) }

  specify "the mother pig did not"\
          "have enough to keep her children" do
    expect(mother_pig.wealth).to \
      eq(Wealth.level(:not_enough))
  end




end
```

```ruby
RSpec.describe PigsLeaveTheHouse do
  let(:story) { … }
  let(:mother_pig) { story.mother_pig }

  before { PigsLeaveTheHouse.tell(story) }

  specify "the mother pig did not"\
          "have enough to keep her children" do
    expect(mother_pig.wealth).to \
      eq(Wealth.level(:not_enough))
  end

  specify "the mother sent her children away" do
    expect(mother_pig.house.occupants).to \
      eq([mother_pig])
  end
end
```

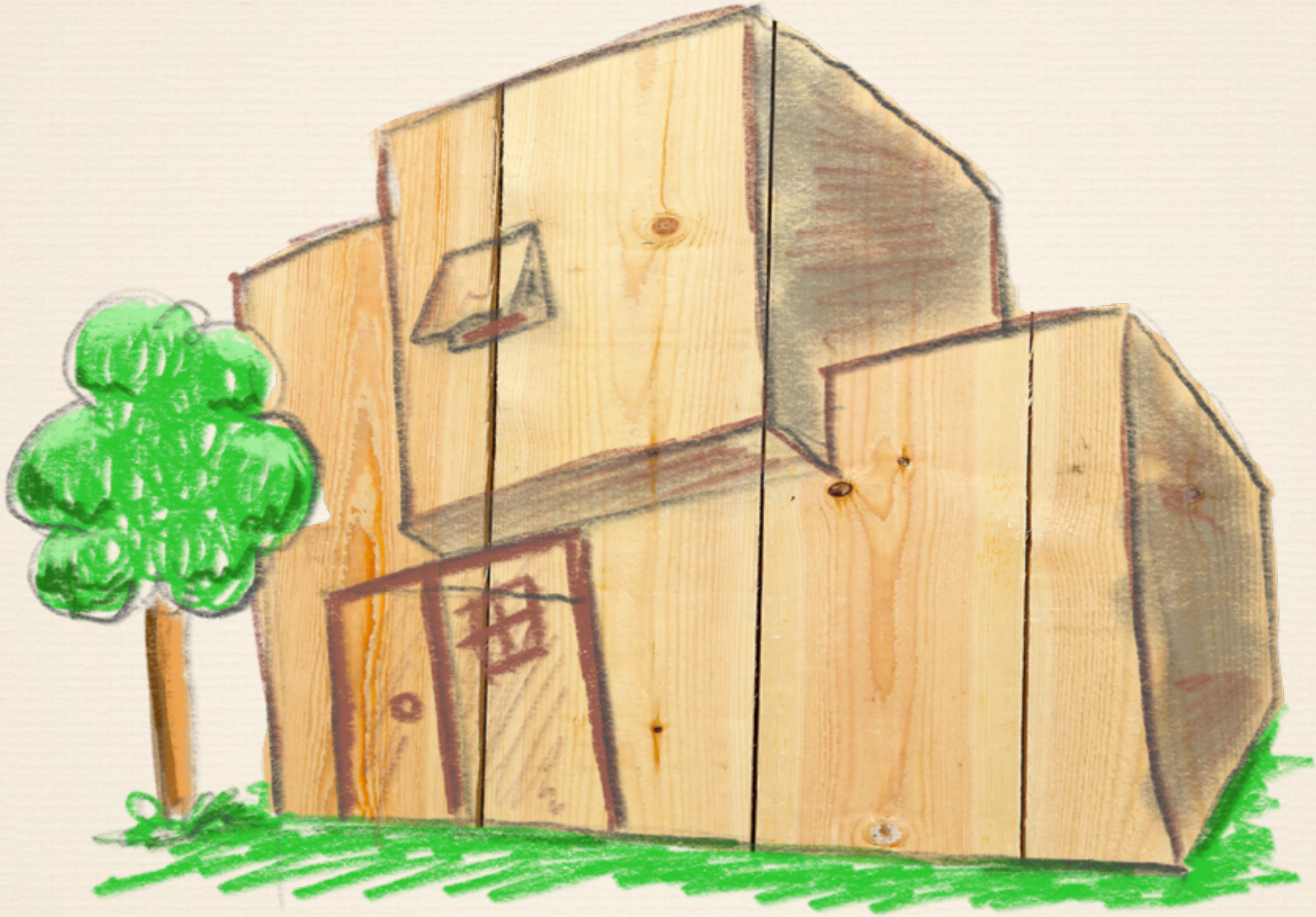
```ruby
RSpec.describe FirstPigMeetsStrawMan do
  let(:story) do
    Story.until_chapter(FirstPigMeetsStrawMan)
  end
  let(:straw_man) { story.straw_man }

  before { FirstPigMeetsStrawMan.tell(story) }

  specify "the first little pig met a"\
          "man carrying a bundle of straw" do
    expect(straw_man).to be_a(Man)
    expect(straw_man.inventory).to \
      eq(Bundle.of(:straw))
  end
end
```

```ruby
RSpec.describe FirstPigReceivesStraw do
  let(:story) do
    Story.until_chapter(FirstPigReceivesStraw)
  end
  let(:straw_man) { story.straw_man }
  let(:first_pig) { story.first_pig }

  before { FirstPigReceivesStraw.tell(story) }


  specify "the man gives the first"\
          "little pig a bundle of straw" do
    expect(first_pig.inventory).to \
      eq(Bundle.of(:straw))
    expect(straw_man.inventory).to be_empty
  end
end
```

```ruby
RSpec.describe FirstPigBuildsStrawHouse do
  let(:story) { … }
  let(:first_pig) { story.first_pig }

  before { FirstPigBuildsStrawHouse.tell(story) }

  specify "the first pig"\
          "built his house of straw" do
    expect(first_pig.house).to be_a(House)
    expect(first_pig.house.building_material).to
      eq(:straw)
    expect(first_pig.house.occupants).to \
      eq([first_pig])
    expect(first_pig.inventory).to be_empty
  end
end
```

```ruby
RSpec.describe SecondPigMeetsStickMan do
  let(:story) do
    Story.until_chapter(SecondPigMeetsStickMan)
  end
  let(:stick_man) { story.stick_man }

  before { SecondPigMeetsStickMan.tell(story) }

  specify "the second little pig met"\
          "a man carrying a bundle of sticks" do
    expect(stick_man).to be_a(Man)
    expect(stick_man.inventory).to \
      eq(Bundle.of(:sticks))
  end
end
```

```ruby
RSpec.describe SecondPigReceivesSticks do
  let(:story) do
    Story.until_chapter(SecondPigReceivesSticks)
  end
  let(:stick_man) { story.stick_man }
  let(:second_pig) { story.second_pig }

  before { SecondPigReceivesSticks.tell(story) }


  specify "the man gives the second"\
          "little pig a bundle of sticks" do
    expect(second_pig.inventory).to \
      eq(Bundle.of(:sticks))
    expect(stick_man.inventory).to be_empty
  end
end
```



```ruby
RSpec.describe SecondPigBuildsStickHouse do
  let(:story) { … }
  let(:second_pig) { story.second_pig }

 before { SecondPigBuildsStickHouse.tell(story) }

  specify "the second pig"\
          "built his house of sticks" do
    expect(second_pig.house).to be_a(House)
    expect(second_pig.house.building_material).to
      eq(:sticks)
    expect(second_pig.house.occupants).to \
      eq([second_pig])
    expect(second_pig.inventory).to be_empty
  end
end
```

```ruby
RSpec.describe ThirdPigMeetsBrickMan do
  let(:story) do
    Story.until_chapter(ThirdPigMeetsBrickMan)
  end
  let(:brick_man) { story.brick_man }

  before { ThirdPigMeetsBrickMan.tell(story) }

  specify "the third little pig met"\
          "a man carrying a load of bricks" do
    expect(brick_man).to be_a(Man)
    expect(brick_man.inventory).to \
      eq(Load.of(:bricks))
  end
end
```

```ruby
RSpec.describe ThirdPigReceivesBricks do
  let(:story) do
    Story.until_chapter(ThirdPigReceivesBricks)
  end
  let(:brick_man) { story.brick_man }
  let(:third_pig) { story.third_pig }

  before { ThirdPigReceivesBricks.tell(story) }


  specify "the man gives the third"\
          "little pig a load of bricks" do
    expect(third_pig.inventory).to \
      eq(Load.of(:bricks))
    expect(brick_man.inventory).to be_empty
  end
end
```

```ruby
RSpec.describe ThirdPigBuildsBrickHouse do
  let(:story) { … }
  let(:third_pig) { story.third_pig }

  before { ThirdPigBuildsBrickHouse.tell(story) }

  specify "the third pig"\
          "built his house of bricks" do
    expect(third_pig.house).to be_a(House)
    expect(third_pig.house.building_material).to
      eq(:bricks)
    expect(third_pig.house.occupants).to \
      eq([third_pig])
    expect(third_pig.inventory).to be_empty
  end
end
```

```ruby
RSpec.describe EnterTheWolf do
  let(:story) do
    Story.until_chapter(EnterTheWolf)
  end
  let(:wolf) { story.wolf }

  before { EnterTheWolf.tell(story) }

  specify "a wolf came along" do
    expect(wolf).to be_a(Wolf)
  end
end
```

```ruby
RSpec.describe WolfAttacksStrawHouse do
  let(:story) { … }
  # …
  let(:straw_house) { story.first_pig.house }

  before do
    allow(wolf).to \
      receive(:huff).with(at: straw_house).
        and_call_original
    allow(wolf).to \
      receive(:puff).with(at: straw_house).
        and_call_original
    WolfAttacksStrawHouse.tell(story)
  end
end
```

```ruby
RSpec.describe WolfAttacksStrawHouse do
  let(:story) { … }
  # …
  let(:straw_house) { story.first_pig.house }

  before do
    allow(wolf).to \
      receive(:huff).with(at: straw_house).
        and_call_original
    allow(wolf).to \
      receive(:puff).with(at: straw_house).
        and_call_original
    WolfAttacksStrawHouse.tell(story)
  end
end
```

```ruby
RSpec.describe WolfAttacksStrawHouse do
  let(:story) { … }
  # …
  let(:straw_house) { story.first_pig.house }

  before do
    allow(wolf).to \
      receive(:huff).with(at: straw_house).
        and_call_original
    allow(wolf).to \
      receive(:puff).with(at: straw_house).
        and_call_original
    WolfAttacksStrawHouse.tell(story)
  end
end
```

```ruby
RSpec.describe WolfAttacksStrawHouse do
  let(:story) { … }
  # …
  let(:straw_house) { story.first_pig.house }

  before



end
```

```ruby
RSpec.describe WolfAttacksStrawHouse do
  let(:story) { … }
  # …
  let(:straw_house) { story.first_pig.house }

  before { … }

  specify "the wolf huffed and puffed"\
          "and blew the straw house down" do
    expect(wolf).to \
      have_received(:huff).with(at: straw_house)
    expect(wolf).to \
      have_received(:puff).with(at: straw_house)
    expect(first_pig.house).to be nil
  end
end
```

```ruby
RSpec.describe WolfAttacksStrawHouse do
  let(:story) { … }
  # …
  let(:straw_house) { story.first_pig.house }

  before { … }

  specify "the wolf huffed and puffed"\
          "and blew the straw house down"

  specify "the first little pig ran"\
          "to his brother's house of sticks" do
    expect(second_pig.house.occupants).to \
      match_array([first_pig, second_pig])
  end
end
```

```ruby
RSpec.describe WolfAttacksStickHouse do
  let(:story) { … }
  # …
  let(:stick_house) { story.second_pig.house }

  before do
    allow(wolf).to \
      receive(:huff).with(at: stick_house).
        and_call_original
    allow(wolf).to \
      receive(:puff).with(at: stick_house).
        and_call_original
    WolfAttacksStickHouse.tell(story)
  end
end
```

```ruby
RSpec.describe WolfAttacksStickHouse do
  let(:story) { … }
  # …
  let(:stick_house) { story.second_pig.house }

  before do
    allow(wolf).to \
      receive(:huff).with(at: stick_house).
        and_call_original
    allow(wolf).to \
      receive(:puff).with(at: stick_house).
        and_call_original
    WolfAttacksStickHouse.tell(story)
  end
end
```

```ruby
RSpec.describe WolfAttacksStickHouse do
  let(:story) { … }
  # …
  let(:stick_house) { story.second_pig.house }

  before do
    allow(wolf).to \
      receive(:huff).with(at: stick_house).
        and_call_original
    allow(wolf).to \
      receive(:puff).with(at: stick_house).
        and_call_original
    WolfAttacksStickHouse.tell(story)
  end
end
```

```ruby
RSpec.describe WolfAttacksStickHouse do
  let(:story) { … }
  # …
  let(:stick_house) { story.second_pig.house }

  before



end
```

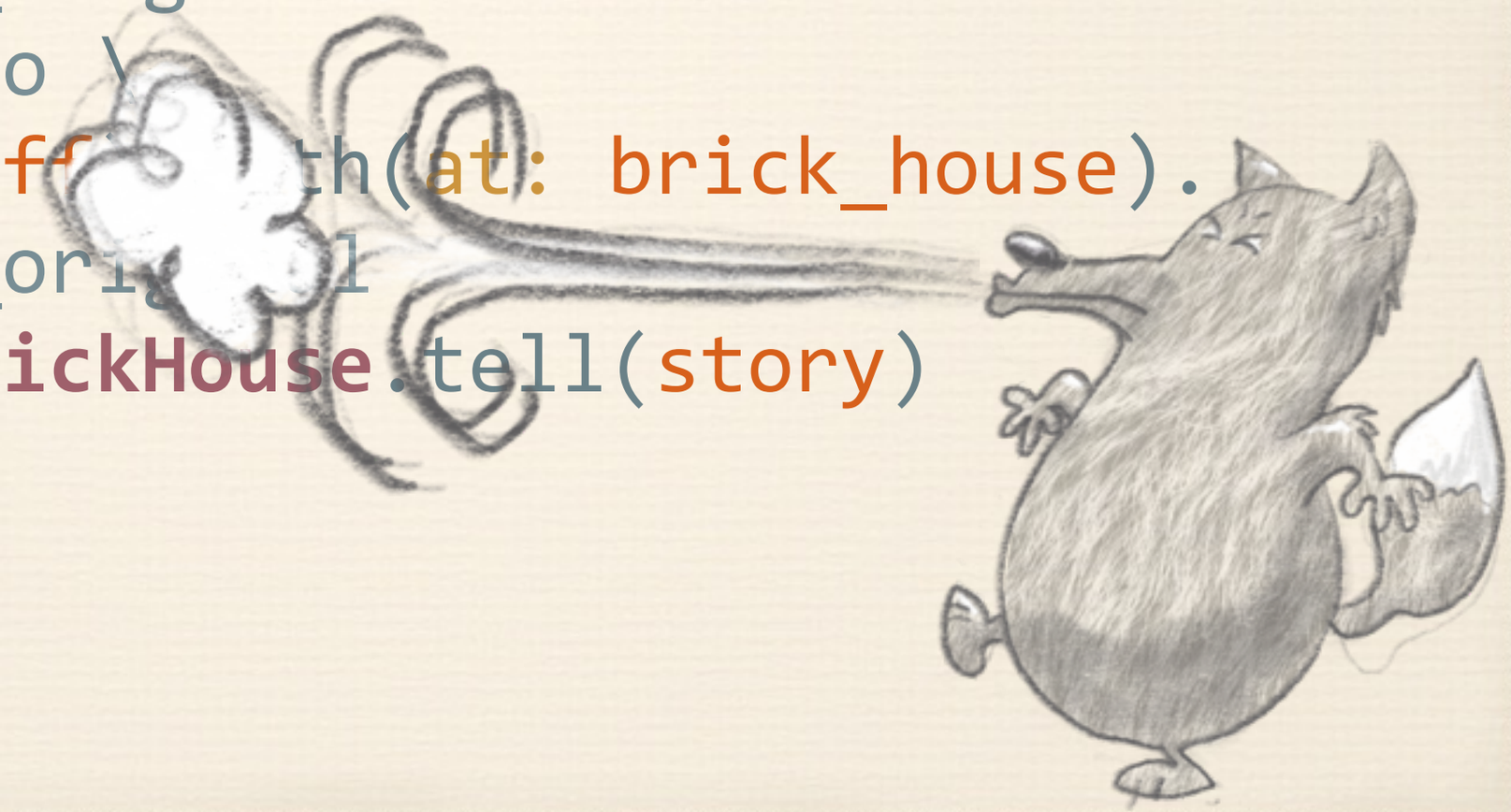
```ruby
RSpec.describe WolfAttacksStickHouse do
  let(:story) { … }
  # …
  let(:stick_house) { story.second_pig.house }

  before { … }

  specify "the wolf huffed and puffed"\
          "and blew the stick house down" do
    expect(wolf).to \
      have_received(:huff).with(at: stick_house)
    expect(wolf).to \
      have_received(:puff).with(at: stick_house)
    expect(second_pig.house).to be nil
  end
end
```

```ruby
RSpec.describe WolfAttacksStickHouse do
  let(:story) { … }
  # …
  let(:stick_house) { story.second_pig.house }

  before { … }

  specify "the wolf huffed and puffed"\
          "and blew the stick house down"

  specify "the little pigs ran to"\
          "their brother's house of bricks" do
    expect(third_pig.house.occupants).to \
      match_array(
        [first_pig, second_pig, third_pig]
      )
  end
end
```

```ruby
RSpec.describe WolfAttacksBrickHouse do
  let(:story) { … }
  # …
  let(:brick_house) { story.third_pig.house }

  before do
    allow(wolf).to \
      receive(:huff).with(at: brick_house).
        and_call_original
    allow(wolf).to \
      receive(:puff).with(at: brick_house).
        and_call_original
    WolfAttacksBrickHouse.tell(story)
  end
end
```

```ruby
RSpec.describe WolfAttacksBrickHouse do
  let(:story) { … }
  # …
  let(:brick_house) { story.third_pig.house }

  before do
    allow(wolf).to \
      receive(:huff).with(at: brick_house).
        and_call_original
    allow(wolf).to \
      receive(:puff).with(at: brick_house).
        and_call_original
    WolfAttacksBrickHouse.tell(story)
  end
end
```

```ruby
RSpec.describe WolfAttacksBrickHouse do
  let(:story) { … }
  # …
  let(:brick_house) { story.third_pig.house }

  before do
    allow(wolf).to \
      receive(:huff).with(at: brick_house).
        and_call_original
    allow(wolf).to \
      receive(:puff).with(at: brick_house).
        and_call_original
    WolfAttacksBrickHouse.tell(story)
  end
end
```



```ruby
RSpec.describe WolfAttacksBrickHouse do
  let(:story) { … }
  # …
  let(:brick_house) { story.third_pig.house }

  before do
    allow(wolf).to \
      receive(:huff).with(at: brick_house).
        and_call_original
    allow(wolf).to \
      receive(:puff).with(at: brick_house).
        and_call_original
    WolfAttacksBrickHouse.tell(story)
  end
end
```

```ruby
RSpec.describe WolfAttacksBrickHouse do
  let(:story) { … }
  # …
  let(:brick_house) { story.third_pig.house }

  before { … }
  specify "the wolf huffed and puffed but could"\
          "not blow the brick house down" do
    expect(wolf).to \
      have_received(:huff).
        with(at: brick_house).twice
    expect(wolf).to \
      have_received(:puff).
        with(at: brick_house).twice
    expect(third_pig.house).to be_truthy
  end
end
```

```ruby
RSpec.describe PigsCloseUpBrickHouse do
  let(:story) do
    Story.until_chapter(PigsCloseUpBrickHouse)
  end
  let(:brick_house) { story.third_pig.house }

  before { PigsCloseUpBrickHouse.tell(story) }

  specify "the door and windows were closed"\
          "but the chimney was open" do
    expect(brick_house.door).to be_closed
    expect(brick_house.windows).to be_closed
    expect(brick_house.chimney).to be_open
  end
end
```

```ruby
RSpec.describe PigsPreparePotOfWater do
  let(:pot) { story.pot }
  let(:water) { pot.water }
  let(:fireplace) { third_pig.house.fireplace }
  let(:boiling_point) { 100 } # °C

  before { PigsPreparePotOfWater.tell(story) }

  specify "the pigs hung a pot of water"\
          "on the fireplace and made a fire" do
    expect(pot.contents).to eq([water])
    expect(fireplace.hearth).to eq(pot)
    expect(fireplace).to be_lit
    expect(water.temperature).to \
      eq(boiling_point)
  end
end
```

```ruby
RSpec.describe WolfInvadesBrickHouse do
  let(:story) do
    Story.until_chapter(WolfInvadesBrickHouse)
  end
  let(:wolf) { story.wolf }
  let(:pot) { story.pot }

  before do
    allow(pot).to \
      receive(:<<).with(wolf).and_call_original
    allow(Story).to \
      receive(:kill).with(wolf).and_call_original
    WolfInvadesBrickHouse.tell(story)
  end
end
```

```ruby
RSpec.describe WolfInvadesBrickHouse do
  let(:story) do
    Story.until_chapter(WolfInvadesBrickHouse)
  end
  let(:wolf) { story.wolf }
  let(:pot) { story.pot }

  before { … }

  specify "the wolf slid into the"\
          "pot of water and was boiled up" do
    expect(pot).to have_received(:<<).with(wolf)
    expect(Story).to \
      have_received(:kill).with(wolf)
    expect(story.wolf).to be nil
  end
end
```

```
➜ [three_little_pigs (master)]$
```

```
➜ [three_little_pigs (master)]$ rspec
Run options: include {:focus=>true}

All examples were filtered out; ignoring {:focus=>true}
 25/25 |================ 100 ==================>| Time: 00:00:00

Finished in 0.03788 seconds (files took 0.31285 seconds to load)
25 examples, 0 failures
Coverage report generated for RSpec to /Users/paul/three_little_
pigs/coverage. 667 / 667 LOC (100.0%) covered.
➜ [three_little_pigs (master)]$
```

# And they lived

# happily

# ever after

# paulfioravanti / three_little_pigs

```ruby
class OnceUponATimeTest < Minitest::Test
  def setup
    story = Story.beginning
    OnceUponATime.tell(story)
    @first_pig = story.first_pig
    @second_pig = story.second_pig
    @third_pig = story.third_pig
  end

  def test_there_were_three_little_pigs
    assert_kind_of(Pig, @first_pig)
    assert_kind_of(Pig, @second_pig)
    assert_kind_of(Pig, @third_pig)
  end
end
```

@paulfioravanti

@leesheppard

# The End