Stop Testing. Start Storytelling.

Mike Schutte
Detroit ~~Craftsman~~ (Artisans? Roommates?) Guild
May 15, 2018

Time	Topic
1	About Me
2	Goals
7	Testing Paradigms
20	Storytelling
10	Demo
5	Wrap-up

About Me

Detroit

Quikly (Rails, React, Node.js, Go)

environmental science -> sociology -> software development
University of Denver -> Indiana University -> Turing -> Quikly

runner, musician, reader, writer

Goals

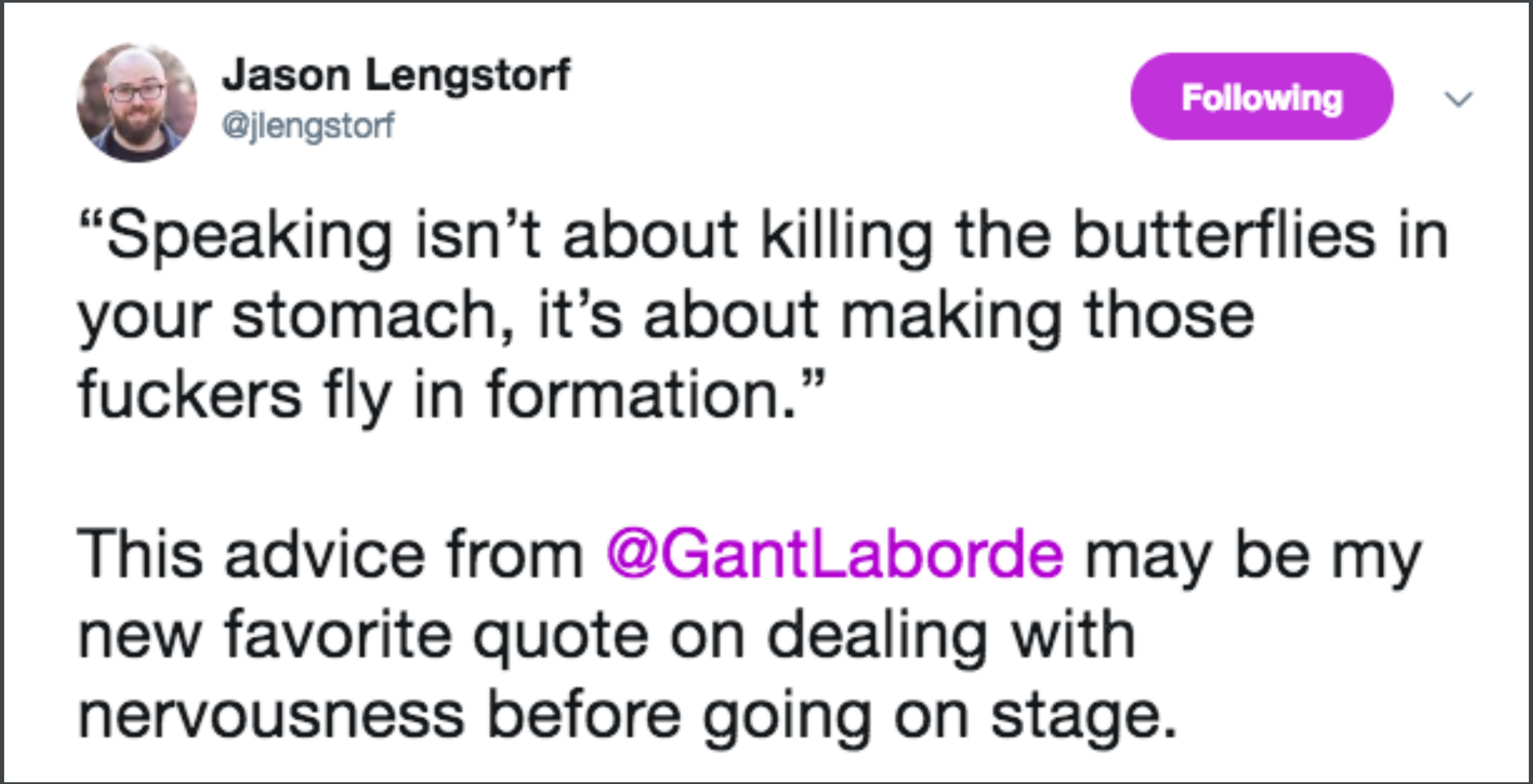
Goals

perspective

Goals

perspective

confidence


Jason Lengstorf
@jlengstorf

“Speaking isn’t about killing the butterflies in your stomach, it’s about making those fuckers fly in formation.”

This advice from @GantLaborde may be my new favorite quote on dealing with nervousness before going on stage.

```
describe("Stop Testing, Start Storytelling", () => {
  before(() => {
    tmikeschu.prepareForTalk()
    tmikeschu.getButterfliesInFormation()
    meetup.garnerInterest()
    audience.map(showUp)
  })

  after(() => {
    audience.map(goHome).map(thinkDifferently)
  })

  it("gives people perspective on and confidence in testing", () => {
    const actual = audience
      .map(stopTestingStartStorytelling)
      .flat_map(takeAways).uniq
    const expected = ["perspective", "confidence"]
    expect(actual).toEqual(expected)
  })
})
```

```ruby
describe "testing paradigms" do
  it "covers different testing paradigms" do
    actual   = stop_testing_start_storytelling.testing_paradigms
    expected = ["test last", "test first", "test driven", "behavior driven"]
    expect(actual).to eq(expected)
  end
end
```

@tmikeschu

Testing Paradigms

bonus word: *para-dig-MAT-ic (adj)*

- a framework containing the basic assumptions, ways of thinking, and methodolog[ies] [for discovering new ideas] that are commonly accepted by members of a ~~scientific~~ community.[1]

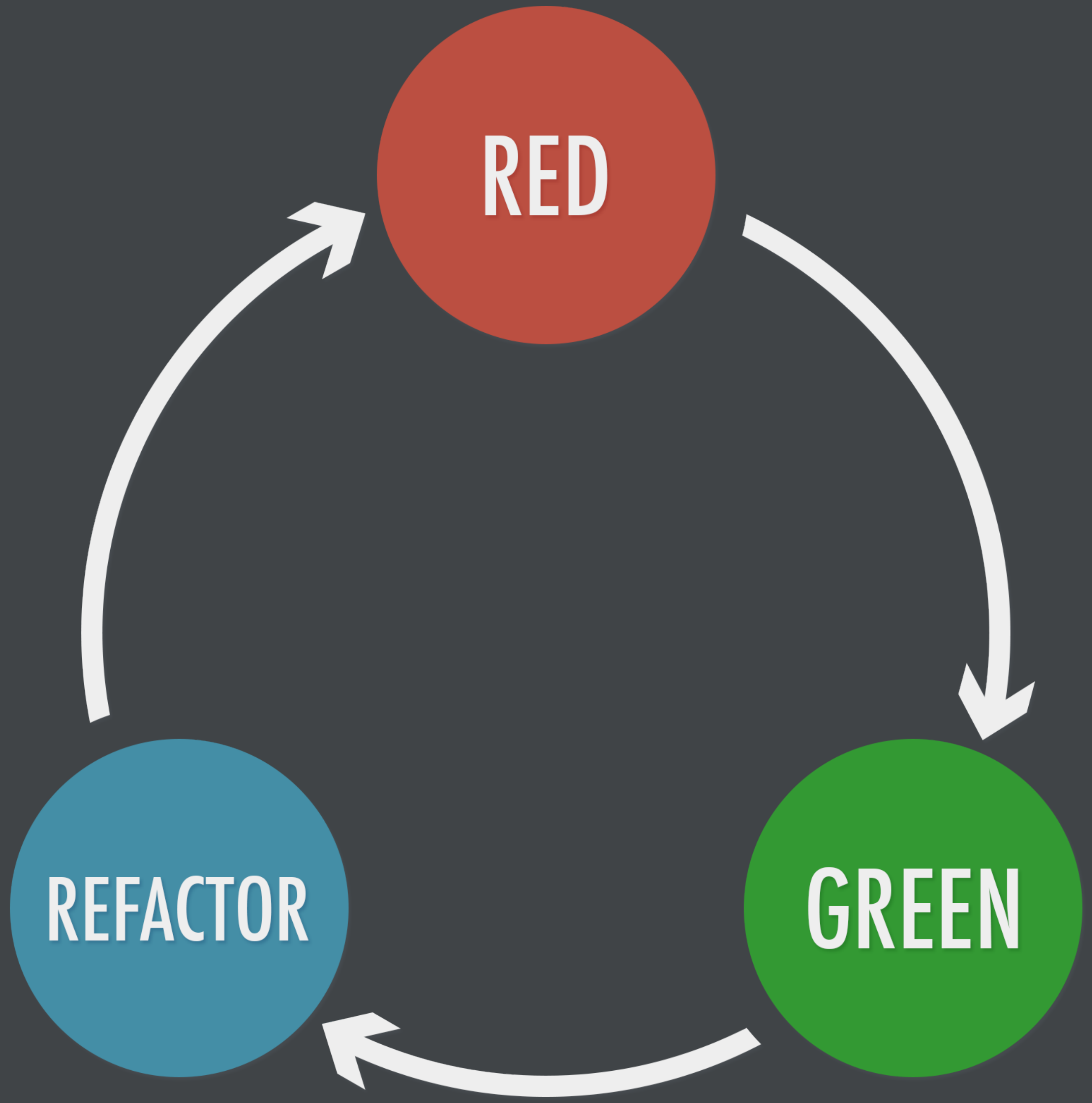
[1] http://www.dictionary.com/browse/paradigm

Test last

- The thing that's supposed to happen is the the thing that happens.
- ~> It's blue because it's blue 🤔.

Test last

@tmikeschu


RED
GREEN
REFACTOR

Test first

- Writing a test to fit an implementation

Test driven
Behavior driven

- Implementing code to fit a test
- *'Program to an interface, not an implementation'*

  - GOF

```ruby
describe "testing paradigms" do
  it "covers different testing paradigms" do
    actual   = stop_testing_start_storytelling.testing_paradigms
    expected = ["test last", "test first", "test driven", "behavior driven"]
    expect(actual).to eq(expected)
  end
end
```

Storytelling and Software

*Software is **always** means to an end*

- Function to Feature
- (HINT: the end is to help users.)

```go
func TestStoryTellingAndSoftware(t *testing.T) {
  var (
    actual   []string
    expected []string
  )

  actual = stopTestingStartStorytelling.StorytellingAndSoftware()
  expected = []string{"communication", "context", "abstraction", "encapsulation"}

  if actual != expected {
    t.Error("Expected talk to cover communication, context, abstraction, and encapsulation, got: ", actual)
  }
}
```

Storytelling and Software

- communication
- context
- abstraction
- encapsulation

Communication

Users communicate with software.

Let's make good listeners.

Context

- `:cause && :effect`
- More context = more understanding
- More understanding = more predictive power
- More predictive power = more confident decision-making
- More confident decision-making = more happiness!

Context

When we tell stories about our software, we write better software.

Abstraction

- Touching a stove
- Breathing
- Muscle memory
- Literally every language

Encapsulation

We encapsulate our code, why not our thoughts?

- Too many jobs makes you bad at all of them.

Encapsulation

Code constrains clarity and imagination.

- Tests can be your compass home

Encapsulation

Meta alert!

- Programs solve problems
- Programming is a problem-solving process
- Maybe the principles that make better *programs* can also result in better *programming*?

```go
func TestStoryTellingAndSoftware(t *testing.T) {
  var (
    actual   []string
    expected []string
  )

  actual = stopTestingStartStorytelling.StorytellingAndSoftware()
  expected = []string{"communication", "context", "abstraction", "encapsulation"}

  if actual != expected {
    t.Error("Expected talk to cover communication, context, abstraction, and encapsulation, got: ", actual)
  }
}
```

Still learning the DSL?

```
/*
description: developers
assert:      should be mindful of context

actual:      perspective while implementing
expected:    developer
  actual should be the same as expected

actual:      perspective while testing
expected:    user
  actual should be the same as expected
*/
```

User <-----------------------> Programmer

Implementing

User <-------------------X> Programmer

Storytelling/testing

User <X-------------------> Programmer

```
/*
description: developers
assert:      should be mindful of context

actual:      perspective while implementing
expected:    developer
actual should be the same as expected

actual:      perspective while testing
expected:    user
actual should be the same as expected
*/
```

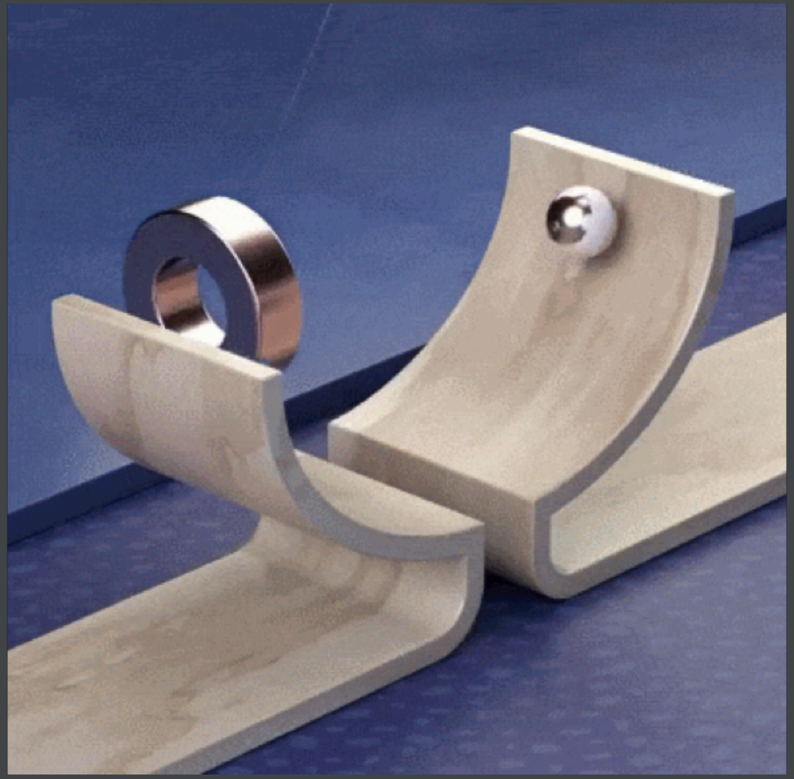
Crowdsource Demo!

- 3 Nouns
- 1 Verb
- 1 Datatype

Wrap-up

- Testing is a *paradigmatic* process, not a product.
- Communication: build good listeners
- Context: enable confident decision making
- Abstraction: you're good at it, it let's you do cool stuff.
- Encapsulation: apply the same discipline to your process as you do to your code.
- Storytelling keeps you closer to users

Action Plan



- Develop a pendulum workflow between storyteller and developer.

```
describe("Stop Testing, Start Storytelling", () => {
  before(() => {
    tmikeschu.prepareForTalk()
    tmikeschu.getButterfliesInFormation()
    meetup.garnerInterest()
    audience.map(showUp)
  })

  after(() => {
    audience.map(goHome).map(thinkDifferently)
  })

  it("gives people perspective on and confidence in testing", () => {
    const actual = audience
      .map(stopTestingStartStorytelling)
      .flat_map(takeAways).uniq
    const expected = ["perspective", "confidence"]
    expect(actual).toEqual(expected)
  })
})
```

Thank you!

@tmikeschu