In FRONT of the Pyramid

Testing for React and React Native

Oussama Ghalbzouri

Who am I ?

- Solution Architect at BIL
- Using React and React Native
- Giving talks at Mobile Apps Luxembourg Meetup and BIL Dev Community

Why this talk ?

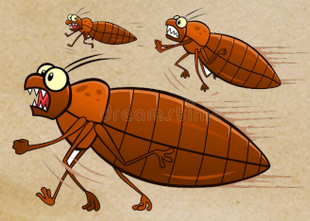
- Testing is important and mandatory in some projects
- But testing for Front-End development can be hard
- Understanding different layers of testing in React and React Native is difficult
- React Native is a framework that requires specific approach for UI/E2E testing
- Should we use TDD in Front-End development?

Why this talk ?

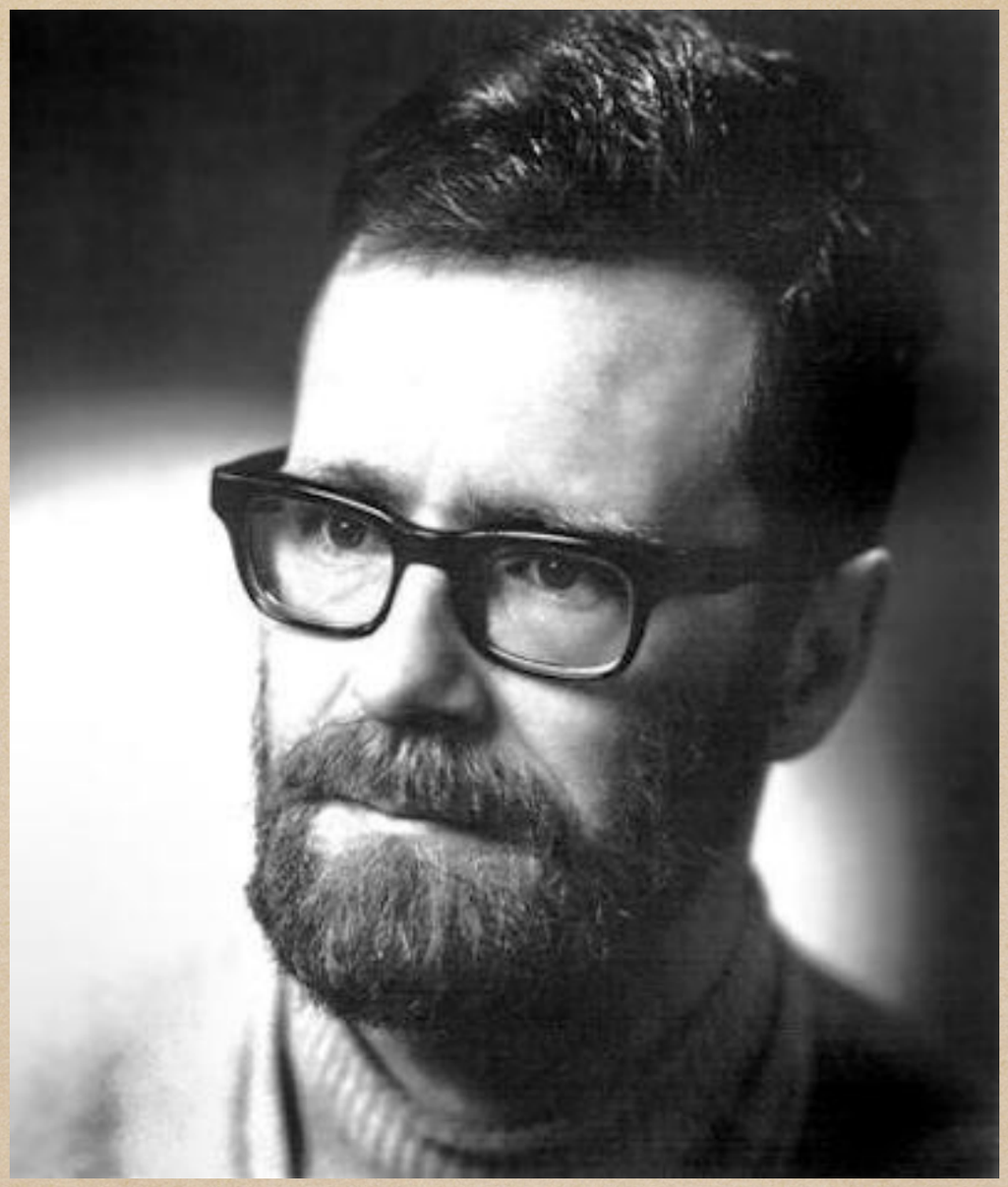
Prevent, Find, Fix

BUGS

Bugs don't like testing



Devs don't like testing ?



Edsger Dijkstra

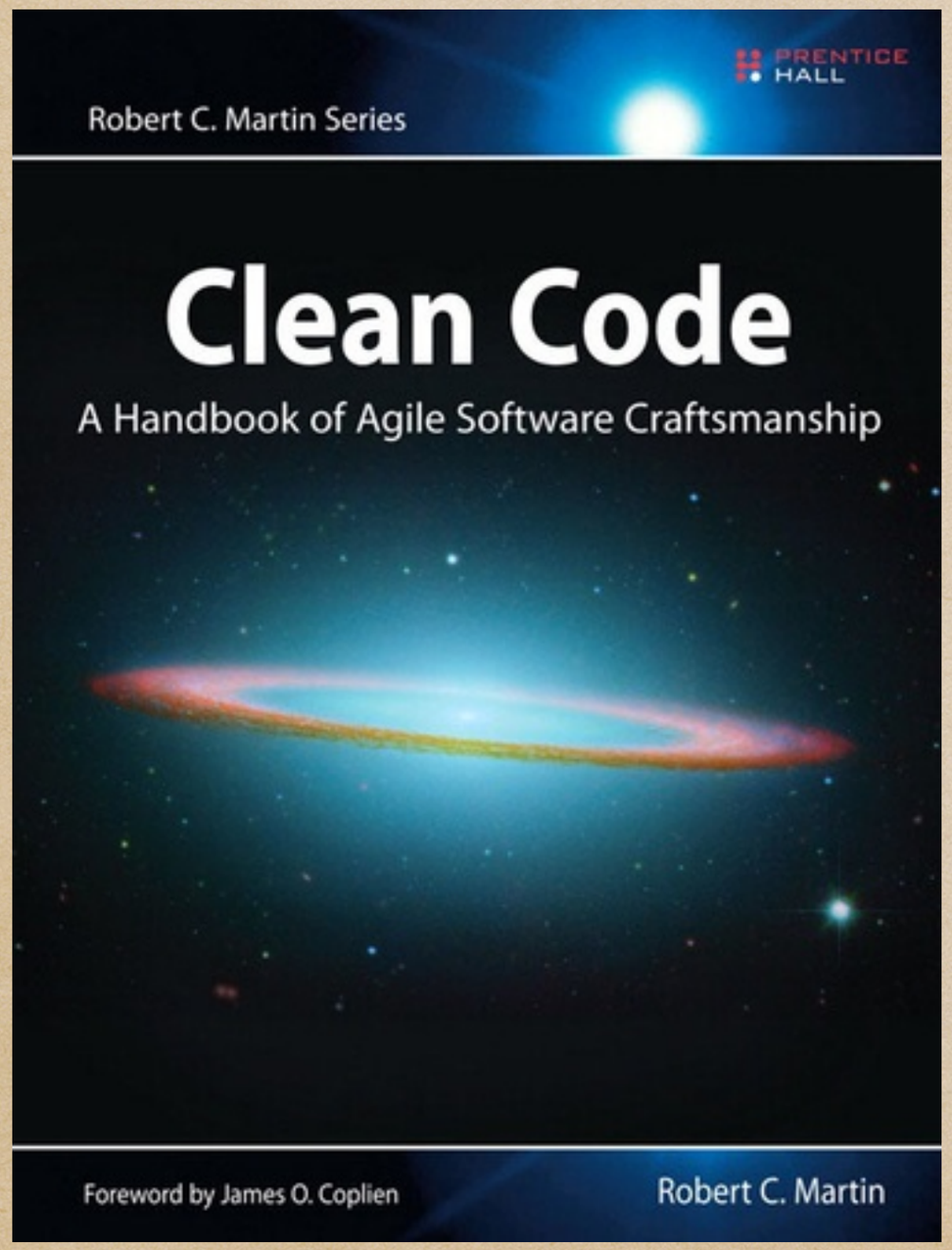
« Programming in Basic
causes brain damage. »

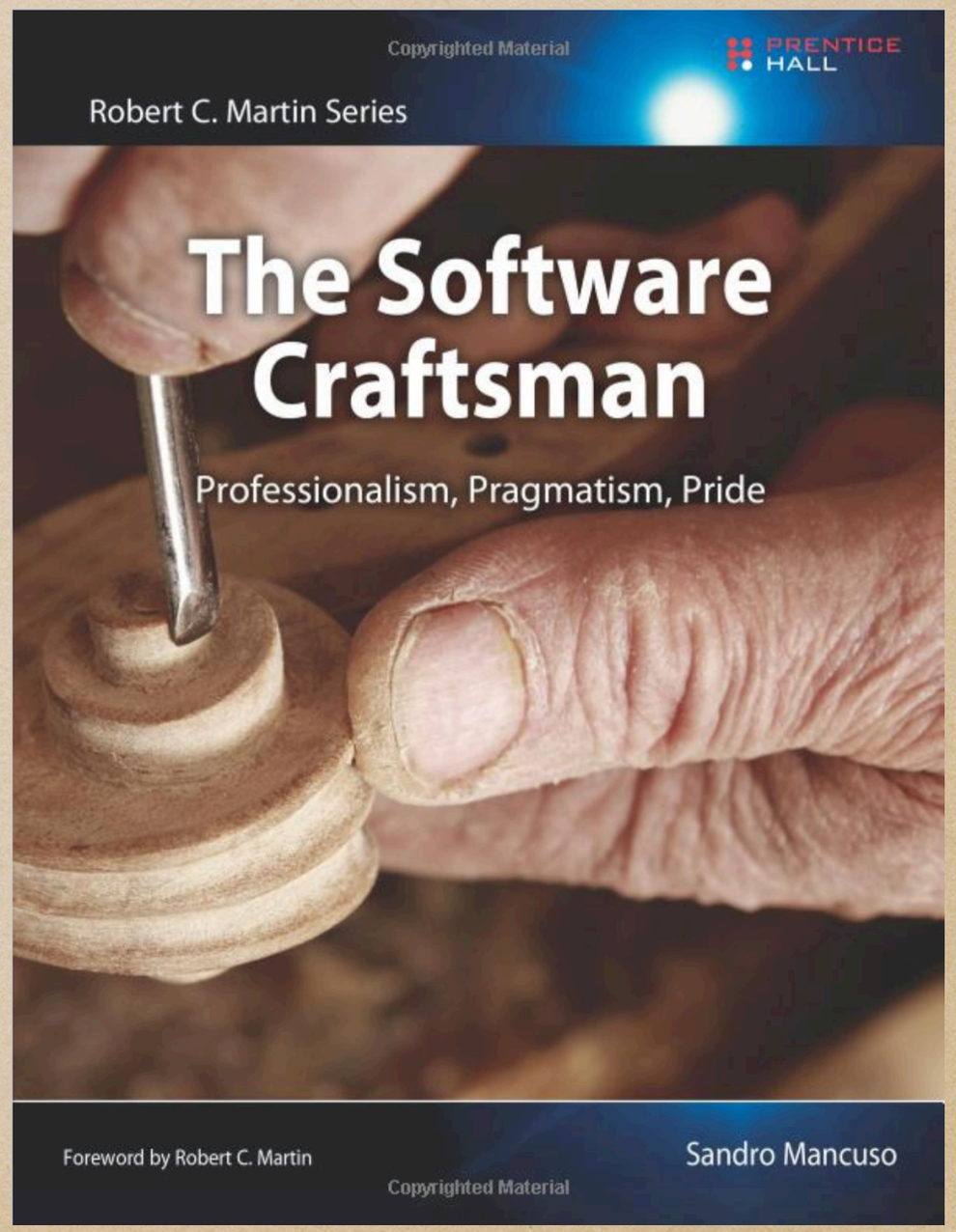
Edsger Dijkstra

« If debugging is the process of
removing software bugs,
then programming must be
the process of putting them in. »

Edsger Dijkstra

Software Craftsmanship ❤️ Testing

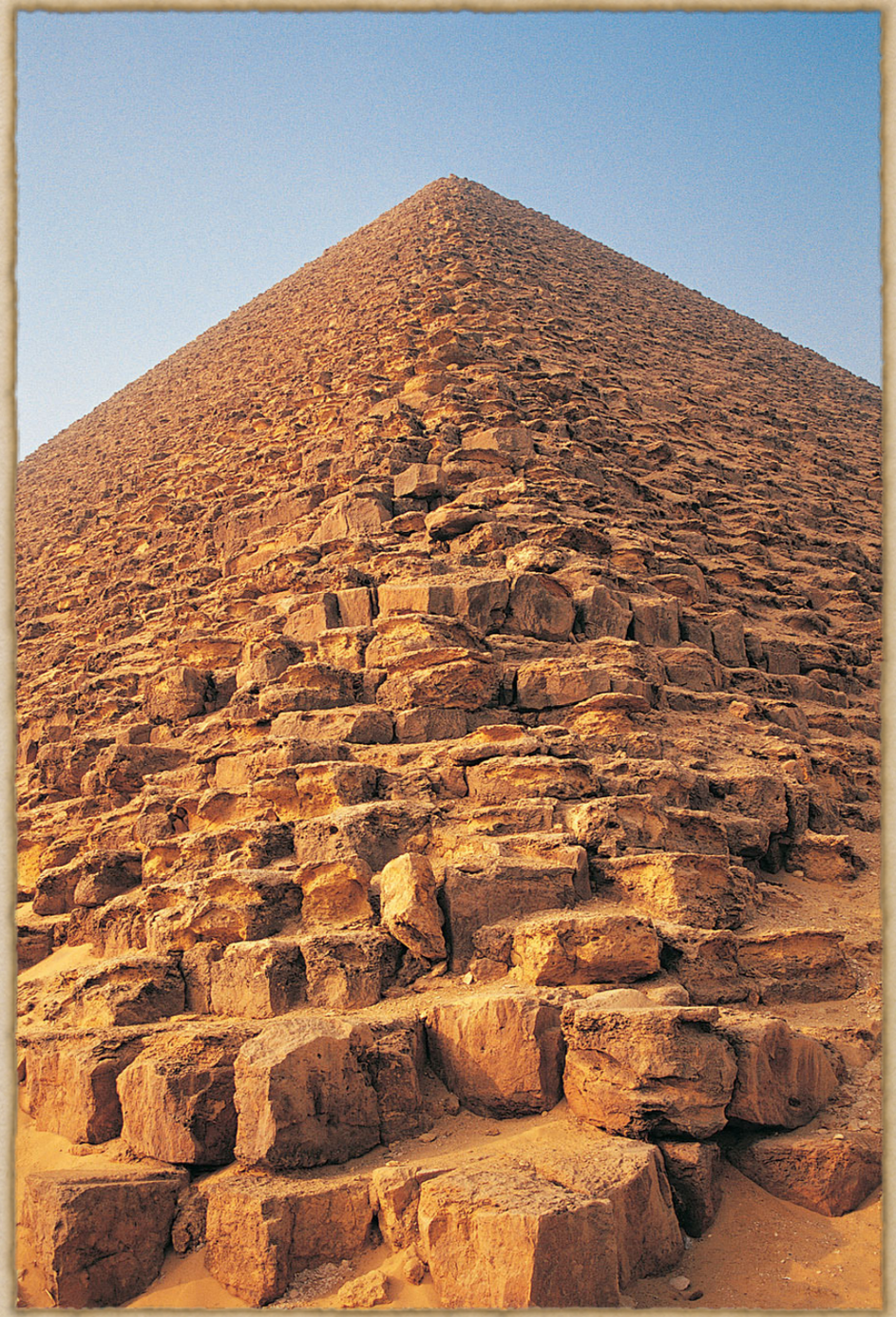




Content

- The Test Pyramid
- The Foundations (Unit Testing / Component Testing)
- The Link (Integration Testing)
- The Eye (UI Testing)
- The Legend (TDD)

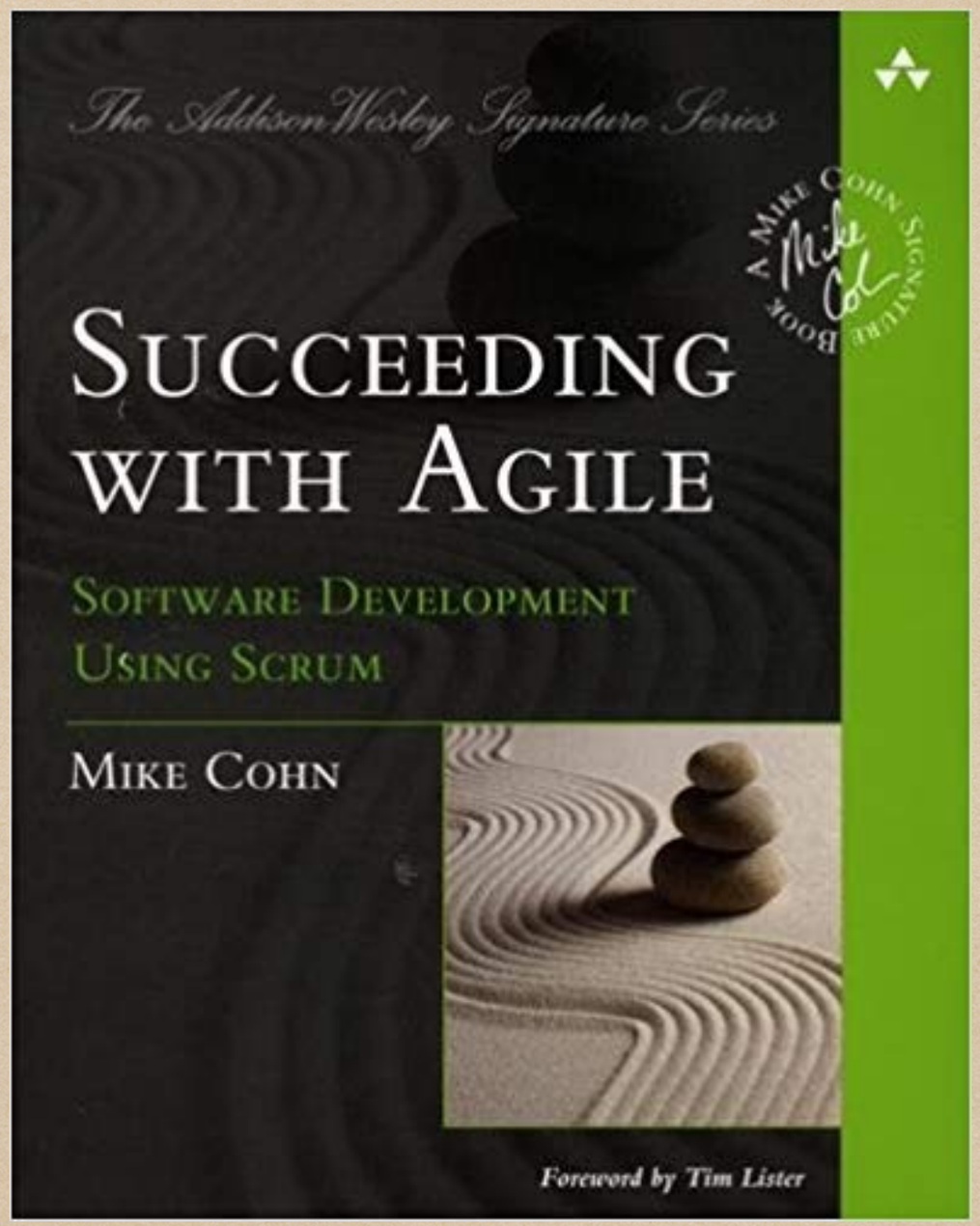
The Test Pyramid



« The "Test Pyramid" is a metaphor that tells us to group software tests into buckets of different granularity. »

Ham Vocke

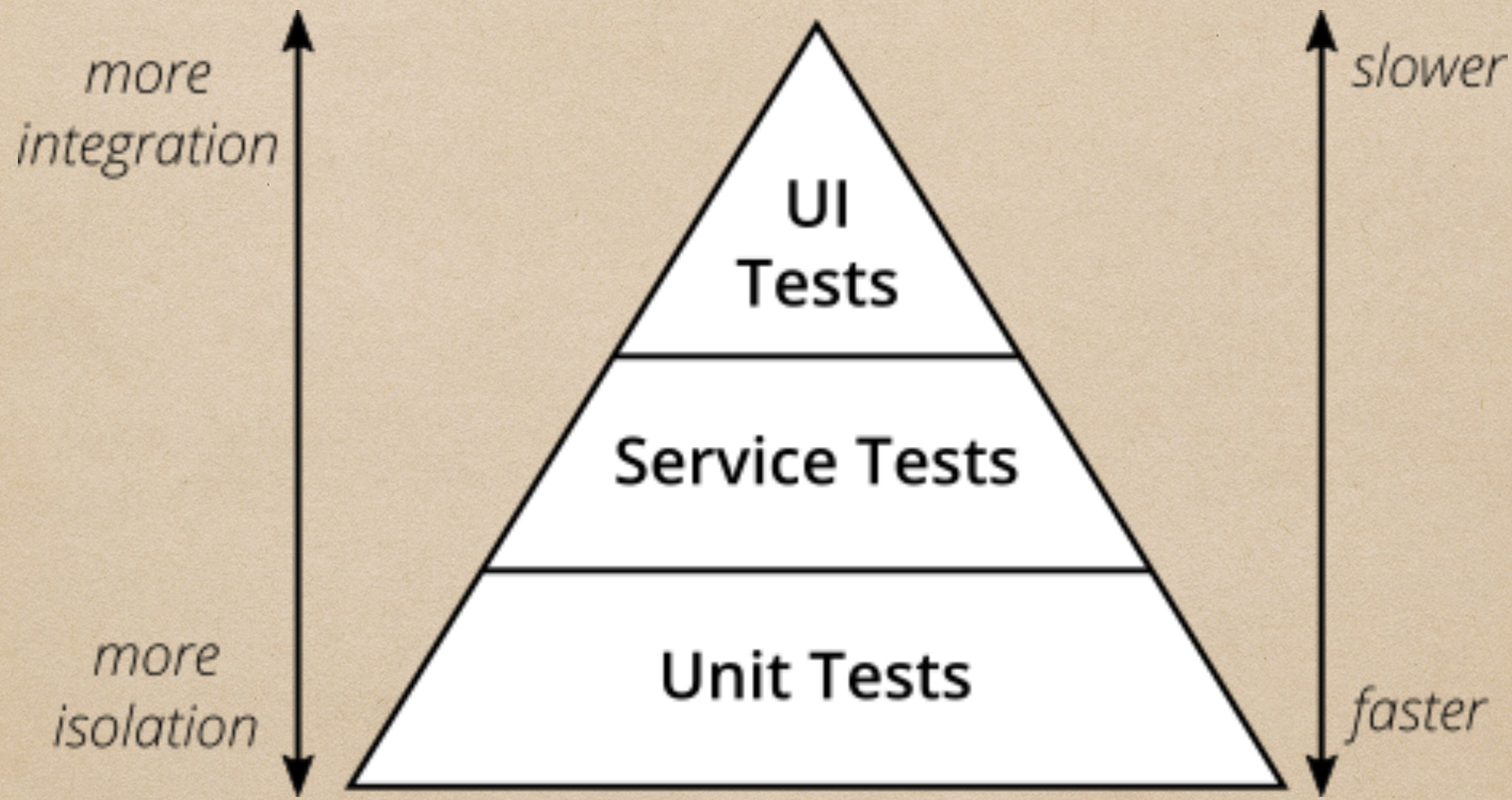
https://martinfowler.com/articles/practical-test-pyramid.html


The Addison Wesley Signature Series
A Mike Cohn Signature Book
Succeeding with Agile
Software Development Using Scrum
Mike Cohn
Foreword by Tim Lister

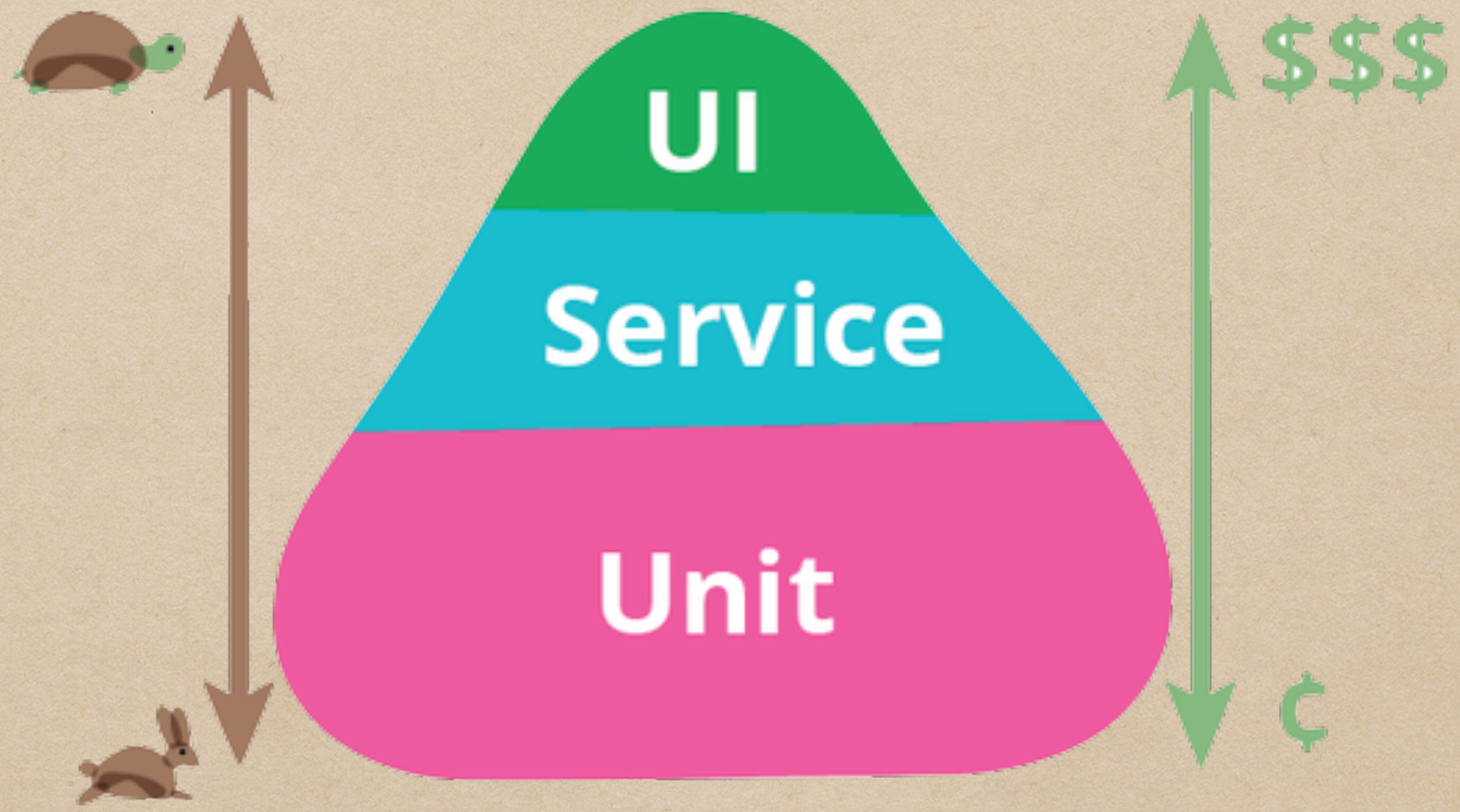
"Write tests with different granularity."

"The more high-level you get,
the fewer tests you should have "

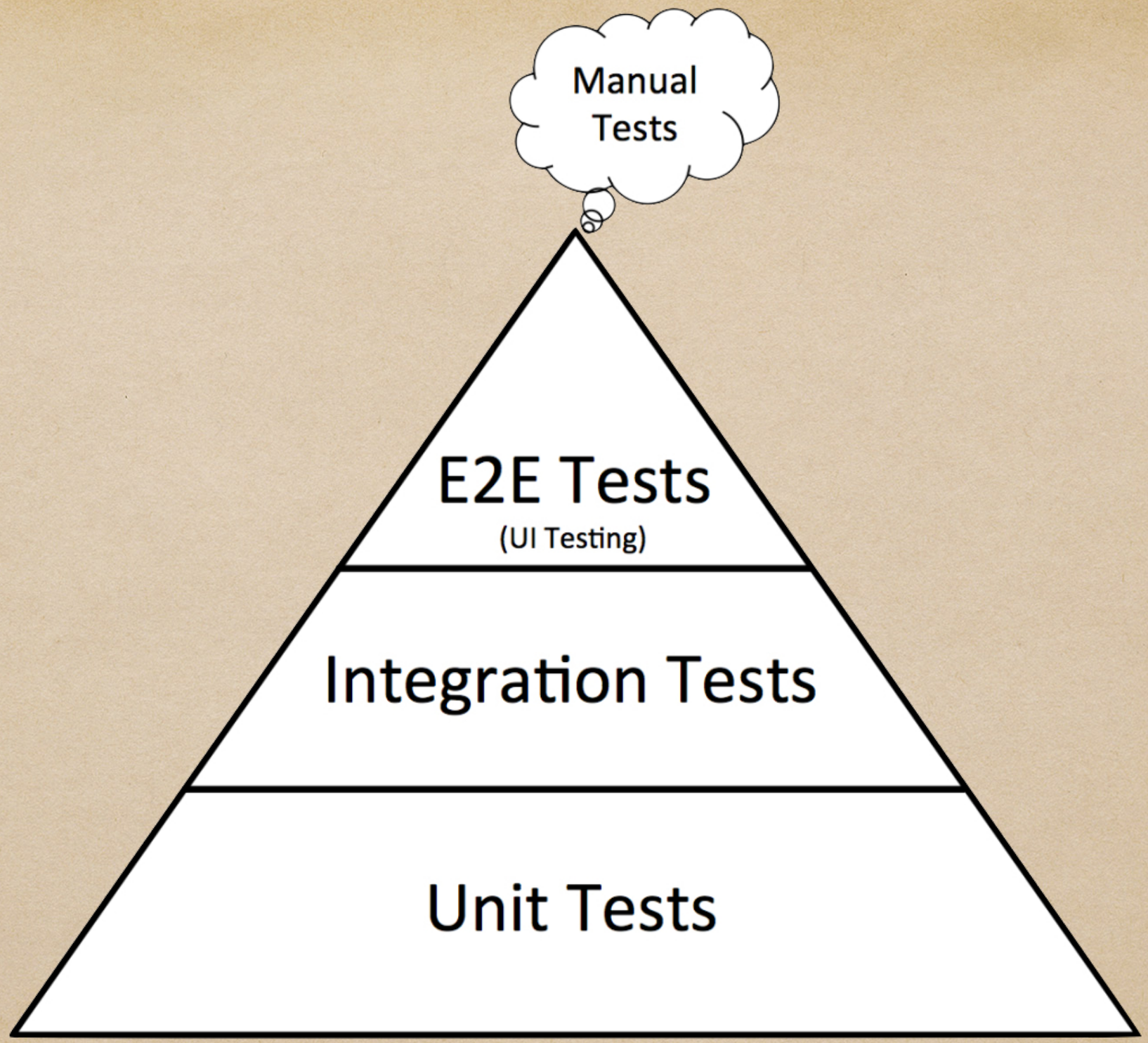
Mike Cohn


more
integration
UI
Tests
Service Tests
Unit Tests
more
isolation
slower
faster

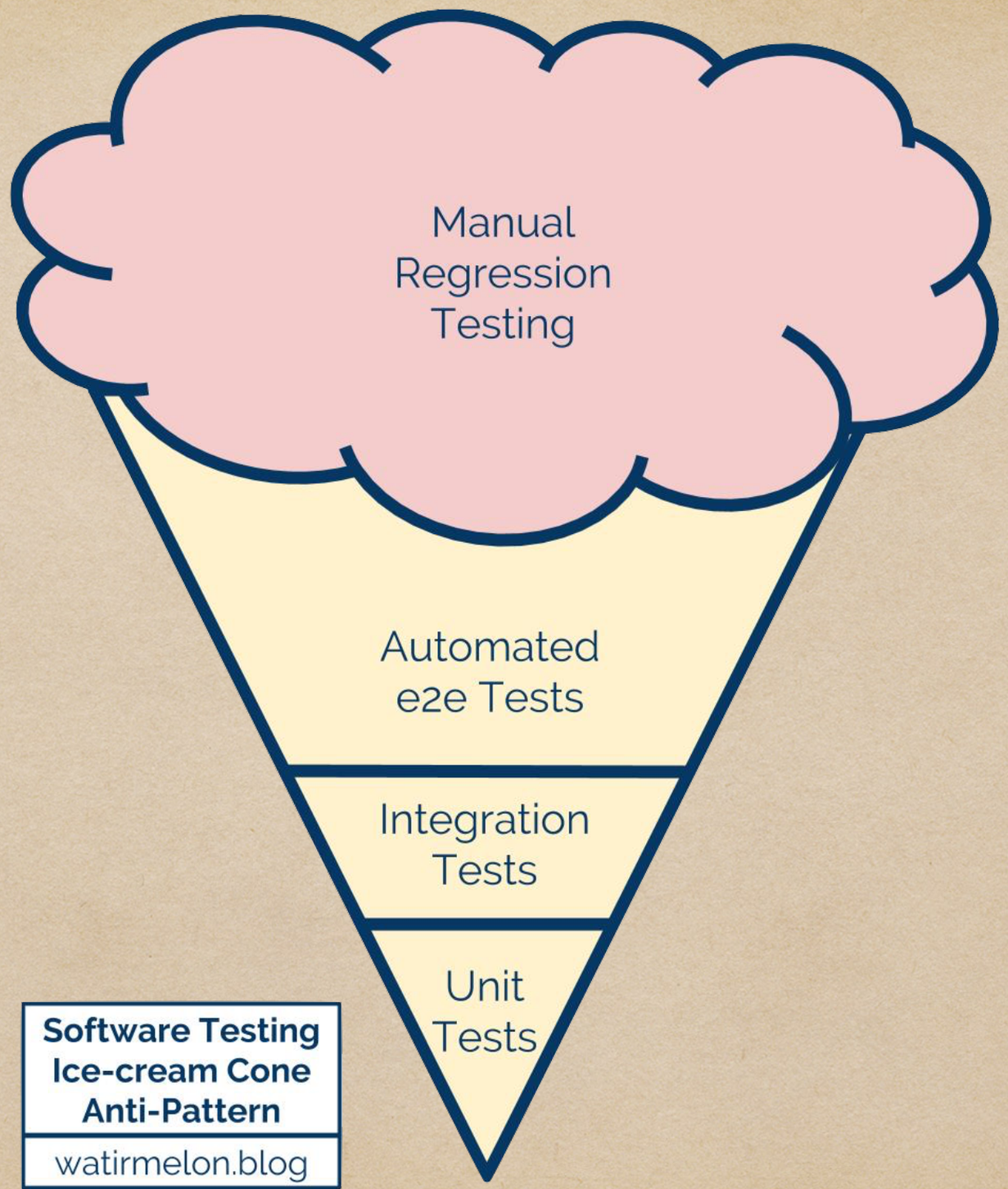
Mike Cohn



https://martinfowler.com/bliki/TestPyramid.html



https://dojo.ministryoftesting.com/dojo/lessons/the-mobile-test-pyramid



Software Testing Ice-cream Cone Anti-Pattern

watirmelon.blog

https://watirmelon.blog/testing-pyramids/

First Level

The Foundations

Before writing tests...

- Static Analysis: ESLint
- Type Checking: Flow
- Transpiled Languages :
  - TypeScript
  - ReasonML

What should I test ?

Uses cases

Bill Sempf
@sempf

QA Engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 999999999 beers. Orders a lizard. Orders -1 beers. Orders a sfdeljknesv.

7:56 PM · 23 sept. 2014 · Twitter Web Client

28,5 k Retweets **21,2 k** J'aime

Use cases for login screen?

- Happy Path: user submits valid credentials 😊
- Sad Path: user submits invalid credentials 😔
- Evil Path: user is hacking your UI with DevTools 😈
- Weird Path: user is using Internet Explorer 

Unit-Testing for Plain Old JavaScript

Know your test doubles

- Stubs
- Mocks
- Spies

Pure Functions

No side effects

$> code plain-js/

Unit-Test for React Component ?

Component Test Tips

- Have a clear test plan (ask for examples to QA/Product)
- Use Given / When / Then
- Test how the component should behave
  (simulate interactions)
- Test how the component should be rendered
  (conditional rendering, dynamic list)

$> code react/

What about Hooks ?

- Good for testing because its separate state and lifecycle logic from rendering logic

- Use custom hooks to reuse logic across different components

- Test *your* custom hooks

$> code custom-hook/

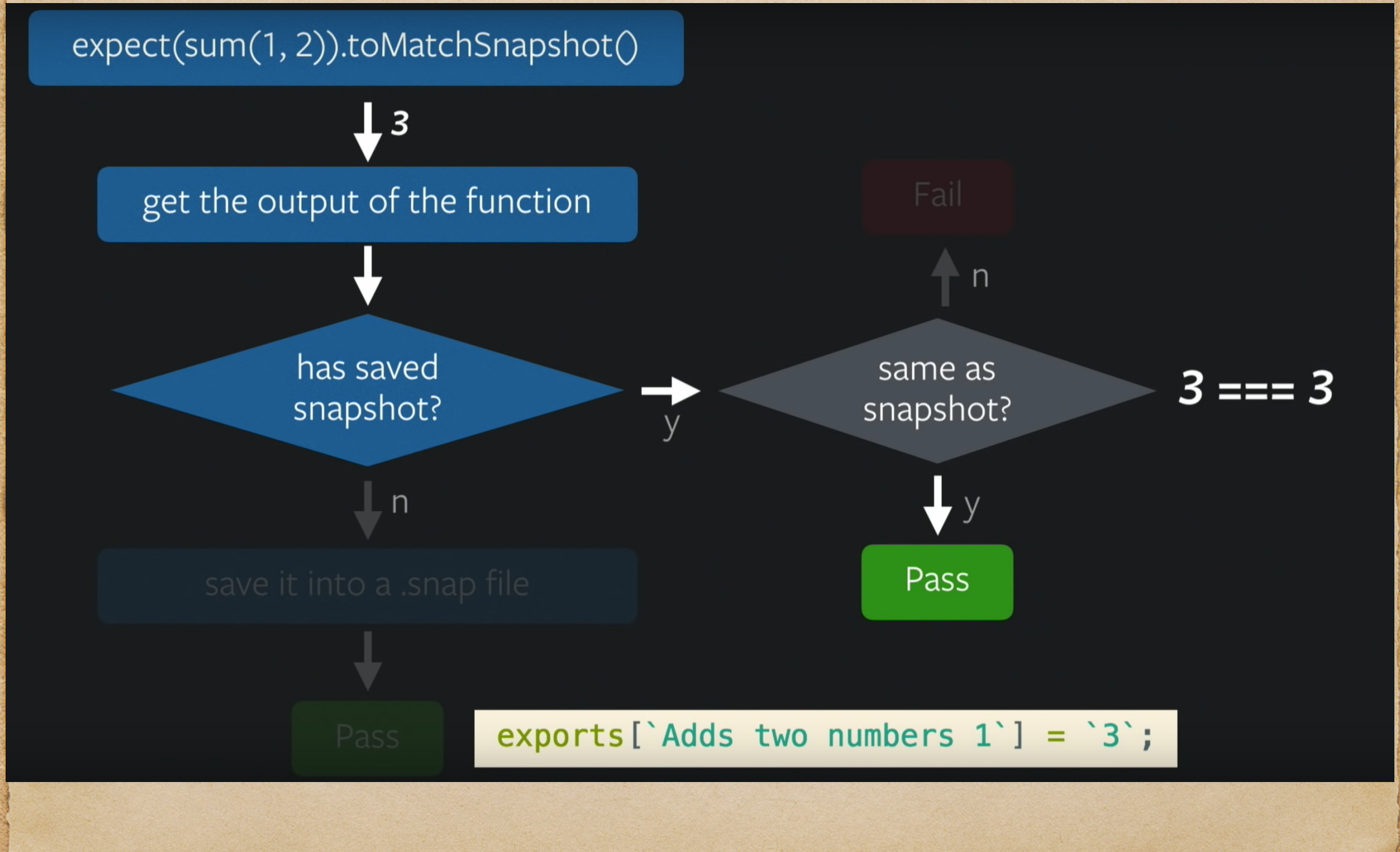
John Carmack
@ID_AA_Carmack

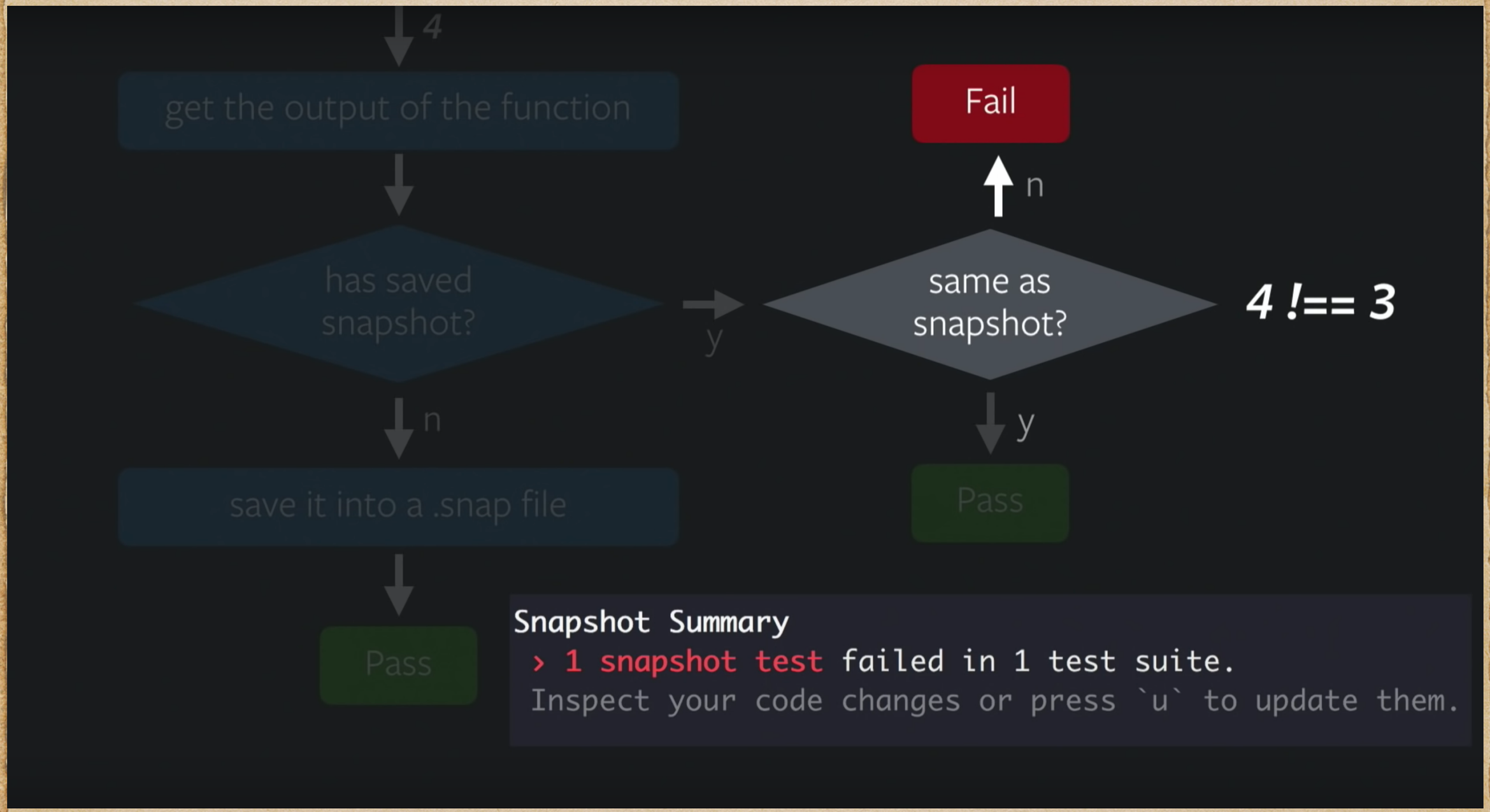
Sometimes, the elegant implementation is just a function. Not a method. Not a class. Not a framework. Just a function.

Traduire le Tweet
7:41 PM · 31 mars 2011 · Twitter Web Client

1,3 k Retweets **986** J'aime

Snapshot Testing


expect(sum(1, 2)).toMatchSnapshot()
3
get the output of the function
has saved snapshot?
y
same as snapshot?
3 === 3
Fail
n
n
y
save it into a .snap file
Pass
Pass
exports[`Adds two numbers 1`] = `3`;


4
get the output of the function
has saved snapshot?
y
same as snapshot?
Fail
n
4 !== 3
n
save it into a .snap file
Pass
y
Pass
Snapshot Summary
› 1 snapshot test failed in 1 test suite.
Inspect your code changes or press `u` to update them.

```
expect(value).toMatchSnapshot()

Received value does not match stored snapshot 1.

- Snapshot
+ Received

@@ -6,10 +6,16 @@
      </div>
    </li>
    <li
      className="Movie">
      <div>
+       ❄️👗✌️Frozen 2
+     </div>
+   </li>
+   <li
+     className="Movie">
+     <div>
```

Snapshot Testing

- Easy to write and maintain (`npm t - -u`)
- Simple way to detect side effects in other components

$> code snapshot/

Unit Testing Quality ?

- Code Coverage
- Mutation Testing with Stryker

Mutation testing involves modifying a program in small ways. Each mutated version is called a mutant and tests detect and reject mutants by causing the behavior of the original version to differ from the mutant.

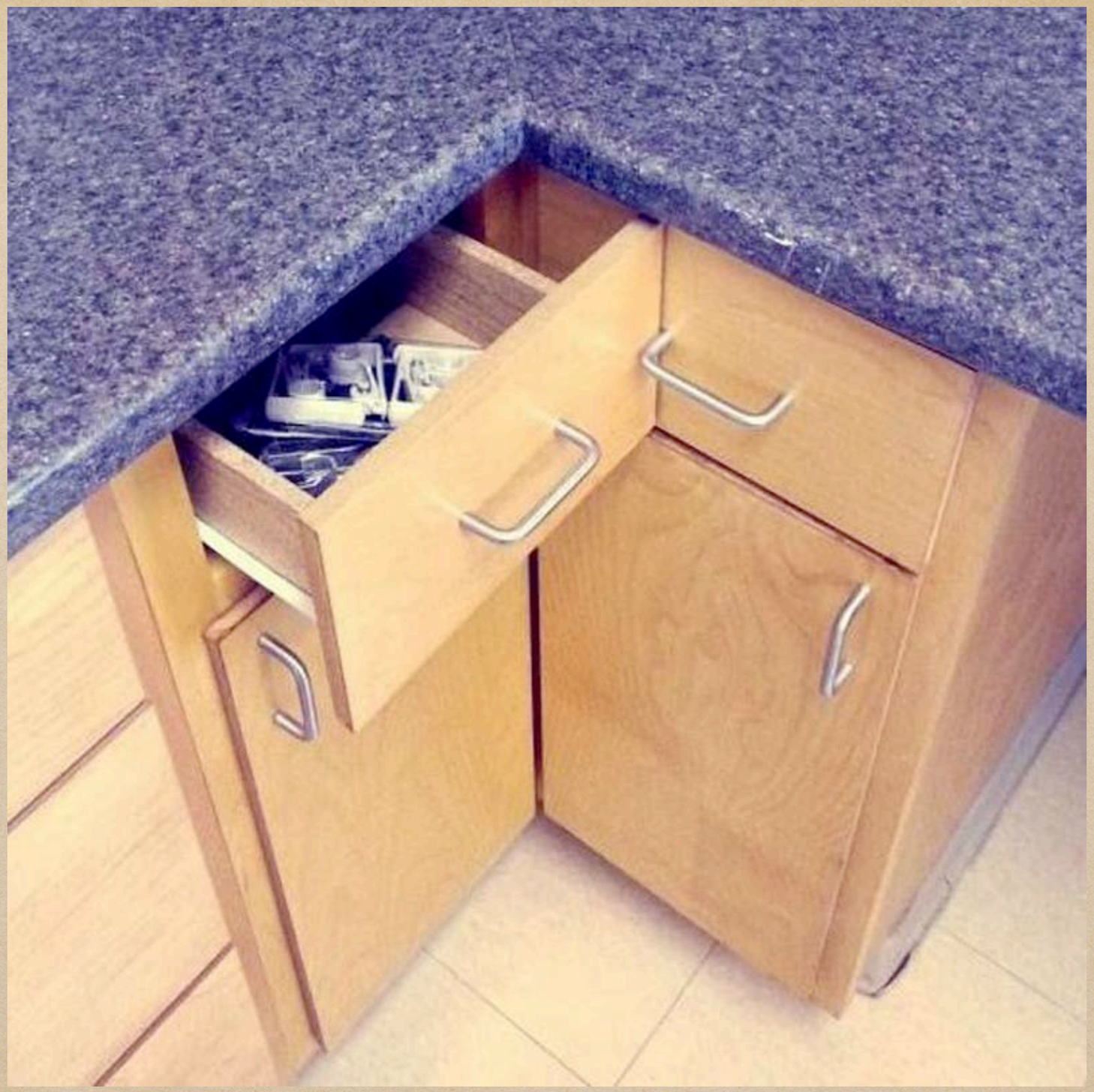
Second Level

The Link

Integration Testing

« Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. »

Wikipedia



2 Unit Tests

0 Integration Tests

2 Unit Tests

0 Integration Tests

Integration test for Web

enzyme

vs

react-testing-library

$> code integration/

Kent C. Dodds 🧢
@kentcdodds

The more your tests resemble the way your software is used, the more confidence they can give you.

♡ 531 4:05 AM - Mar 23, 2018

187 people are talking about this

Integration test for React Native

No deep rendering (Mount) in enzyme for React Native.

react-native-testing-library

```
import { render, fireEvent } from 'react-native-testing-library';
import { QuestionsBoard } from '../QuestionsBoard';
import { Question } from '../Question';

function setAnswer(question, answer) {
  fireEvent.changeText(question, answer);
}

test('should verify two questions', () => {
  const { getAllByType, getByText } = render(<QuestionsBoard {...props} />);
  const allQuestions = getAllByType(Question);

  setAnswer(allQuestions[0], 'a1');
  setAnswer(allQuestions[1], 'a2');

  fireEvent.press(getByText('submit'));

  expect(props.verifyQuestions).toBeCalledWith({
    '1': { q: 'q1', a: 'a1' },
    '2': { q: 'q2', a: 'a2' },
  });
});
```

Redux Testing

$> code redux/

Third Level

The Eye

E2ETesting with Katalon

UI Testing for Mobile App

Detox

no more sleep()

What is Detox?

In one sentence:

Gray box End-to-End testing and automation library for mobile apps.

Black Box vs Gray Box

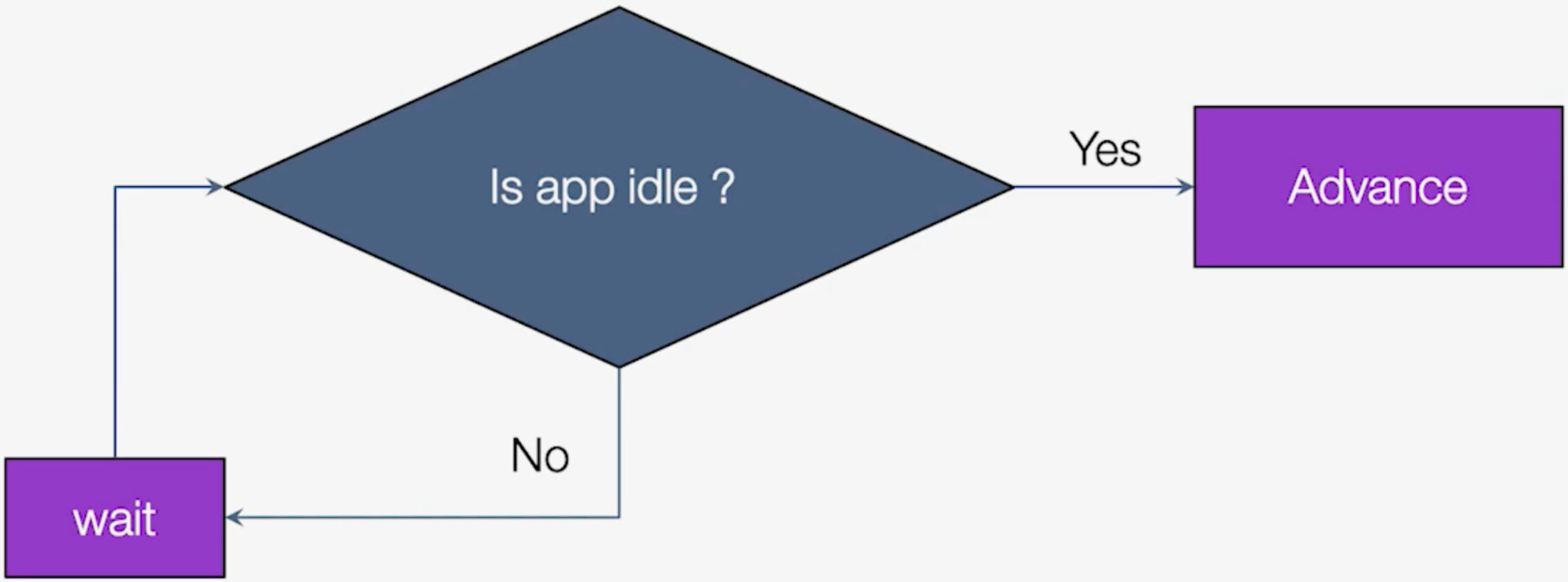
Black Box

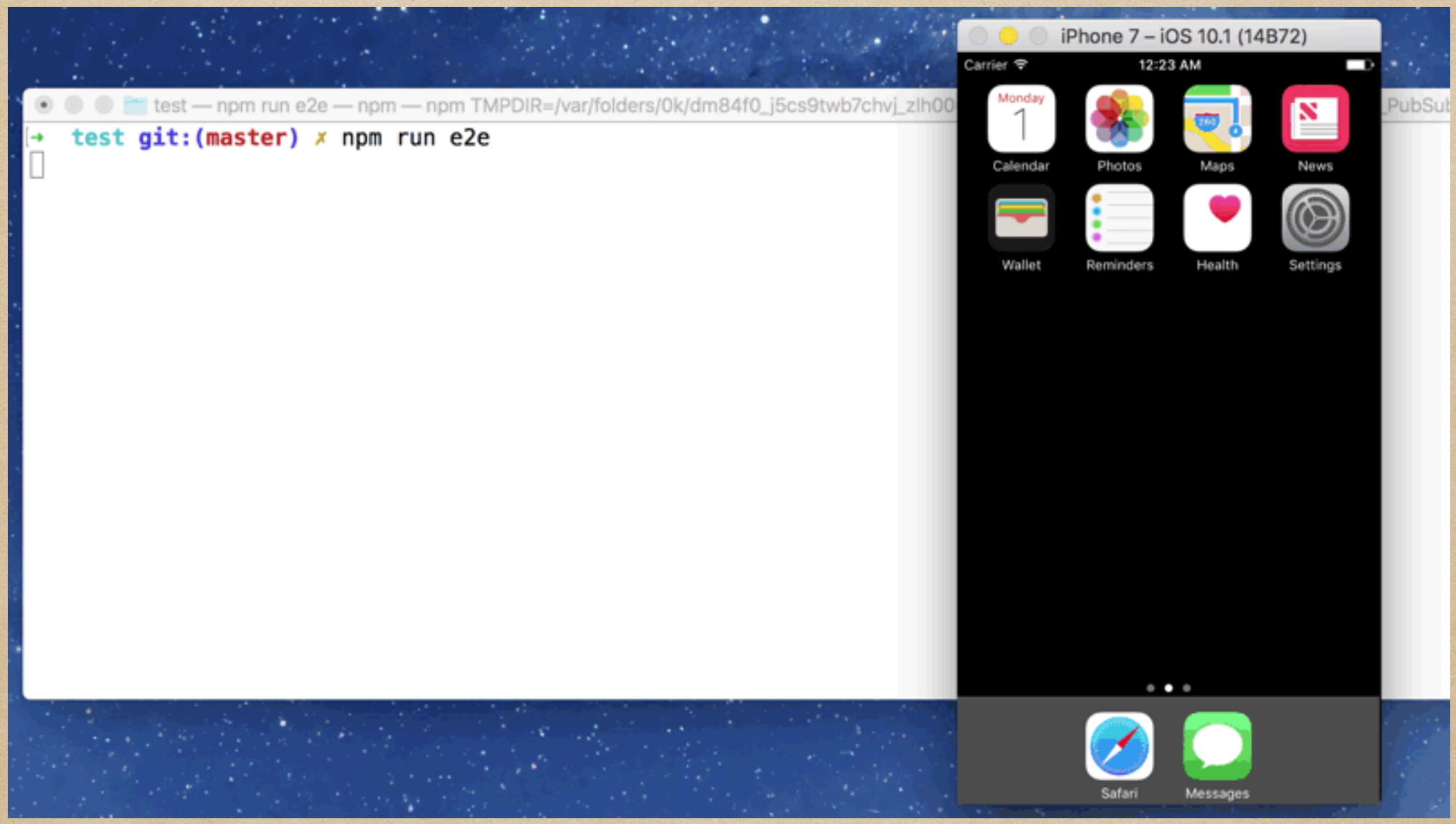
A method of testing stuff from the outside, without knowing what's going on internally.

Gray Box

Allows the test framework to monitor the app from the inside and actually **synchronize** with it.

Gray Box Sync




test — npm run e2e — npm — npm TMPDIR=/var/folders/0k/dm84f0_j5cs9twb7chvj_zlh00
test git:(master) ✗ npm run e2e
iPhone 7 – iOS 10.1 (14B72)
Carrier
12:23 AM
Monday
1
Calendar
Photos
Maps
News
Wallet
Reminders
Health
Settings
Safari
Messages

device Object Functions

device.launchApp()

device.terminateApp()

device.sendToHome()

device.reloadReactNative()

device.installApp()

device.uninstallApp()

device.openURL(url)

device.sendUserNotification(params)

device.sendUserActivity(params)

device.setOrientation(orientation)

device.setLocation(lat, lon)

device.setURLBlacklist()

device.enableSynchronization()

device.disableSynchronization()

device.resetContentAndSettings()

device.getPlatform()

device.pressBack() Android Only

device.shake() iOS Only

Matchers 🔎

```
by.id()

by.text()

by.label()

by.type()

by.traits()
```

for more uniqueness:

```
.withAncestor()

.withDescendant()

.and()
```

Actions 👉

```
.tap()
.longPress()
.multiTap()
.tapAtPoint()
.typeText()
```

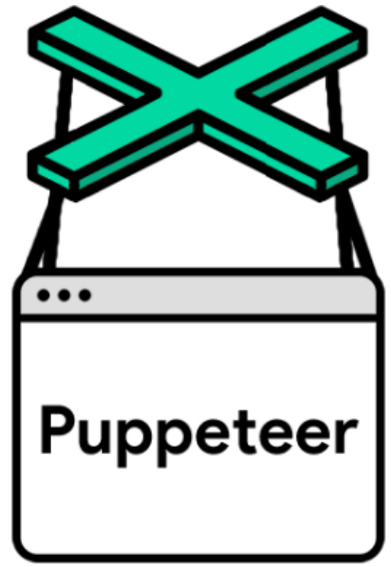
```
.replaceText()
.clearText()
.scroll()
.scrollTo()
.swipe()
```

Detox

- Fast feedback loop
- "`testID`" can also be used by Black-Box Testing framework like Appium
- Easily mock file with `.e2e.js` extension
- All in JavaScript with `await/async` (no `sleep`)

UI Testing for Web

Puppeteer



API | FAQ | Contributing | Troubleshooting

> Puppeteer is a Node library which provides a high-level API to control Chrome or Chromium over the DevTools Protocol. Puppeteer runs headless by default, but can be configured to run full (non-headless) Chrome or Chromium.

Puppeteer

- Generate screenshots and PDFs of pages.
- Crawl a SPA (Single-Page Application)
- Automate form submission, UI testing, keyboard input, etc.
- Run your tests directly in the latest version of Chrome using the latest JavaScript and browser features.
- Capture a timeline trace of your site to help diagnose performance issues.
- Mock cookies and network requests
- Record automations via Chrome plugins Katalon recorder or Puppeteer recorder

$> code puppeteer/

Manual Testing

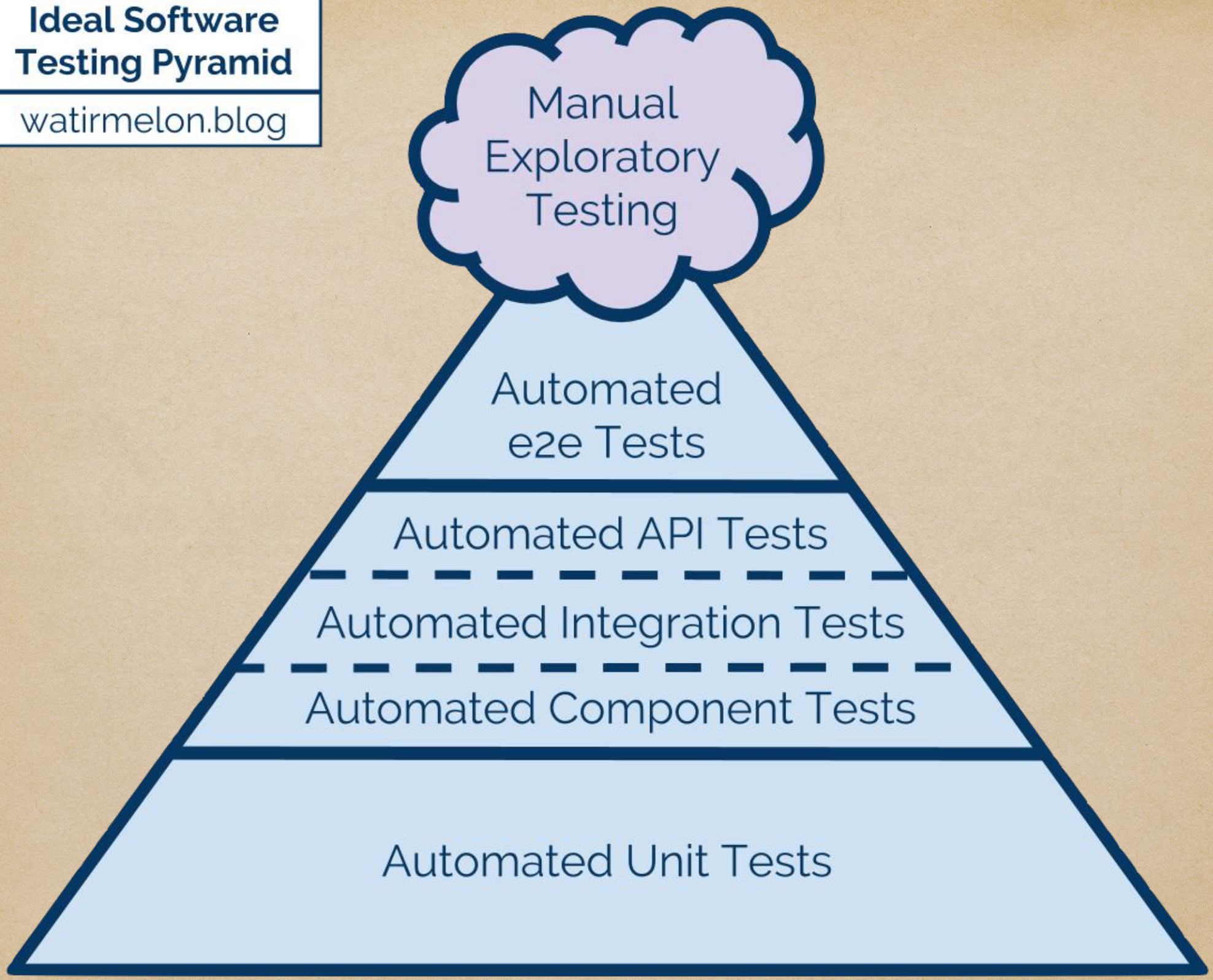
Ideal Software Testing Pyramid

Manual
Exploratory Testing

watirmelon.blog

Ideal Software Testing Pyramid

watirmelon.blog

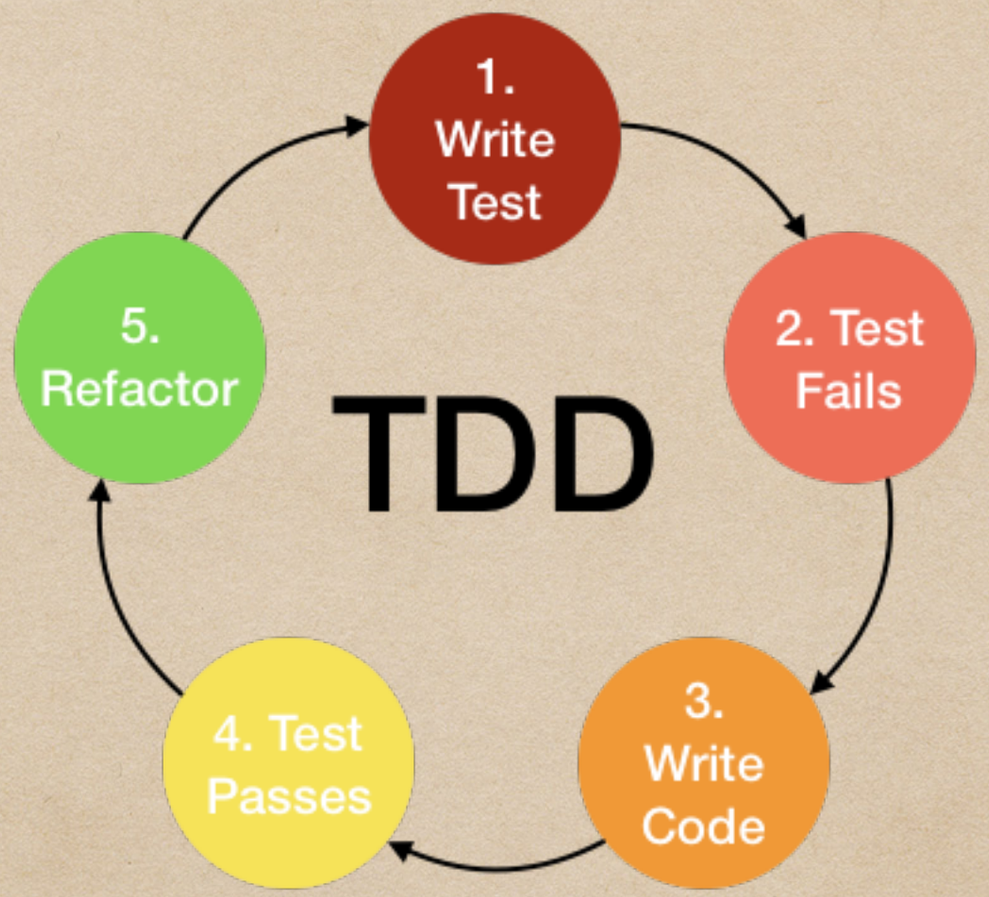


Exploratory testing is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.

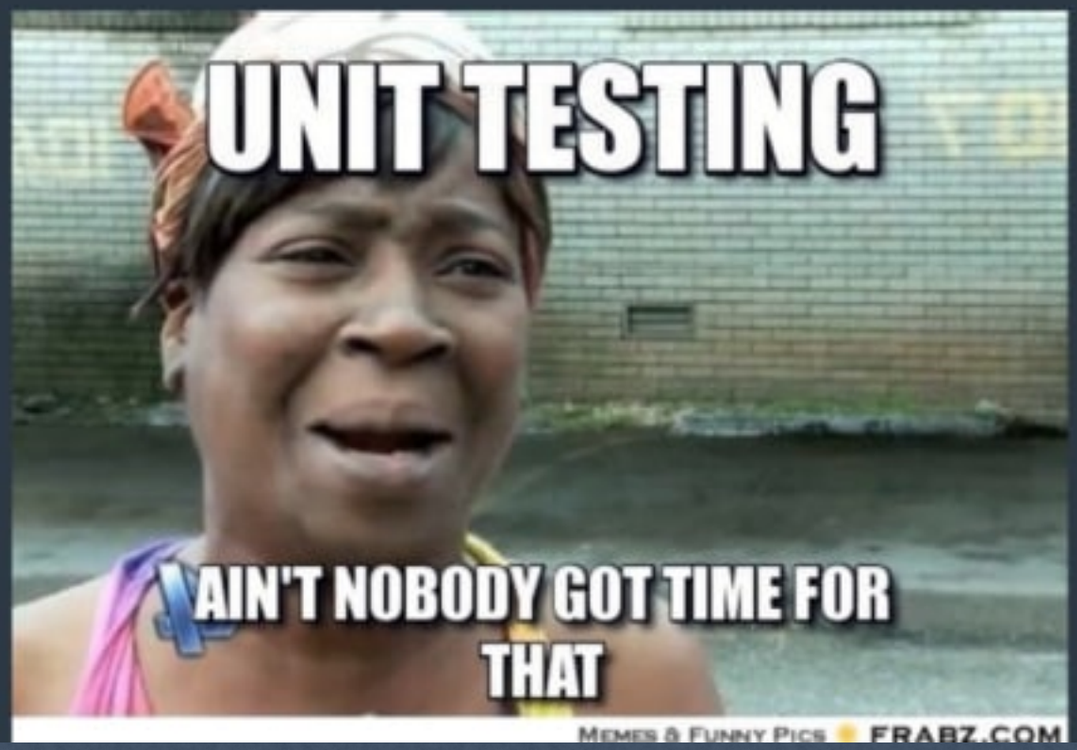
It is defined as a type of testing where Test cases are not created in advance but testers check system on the fly.

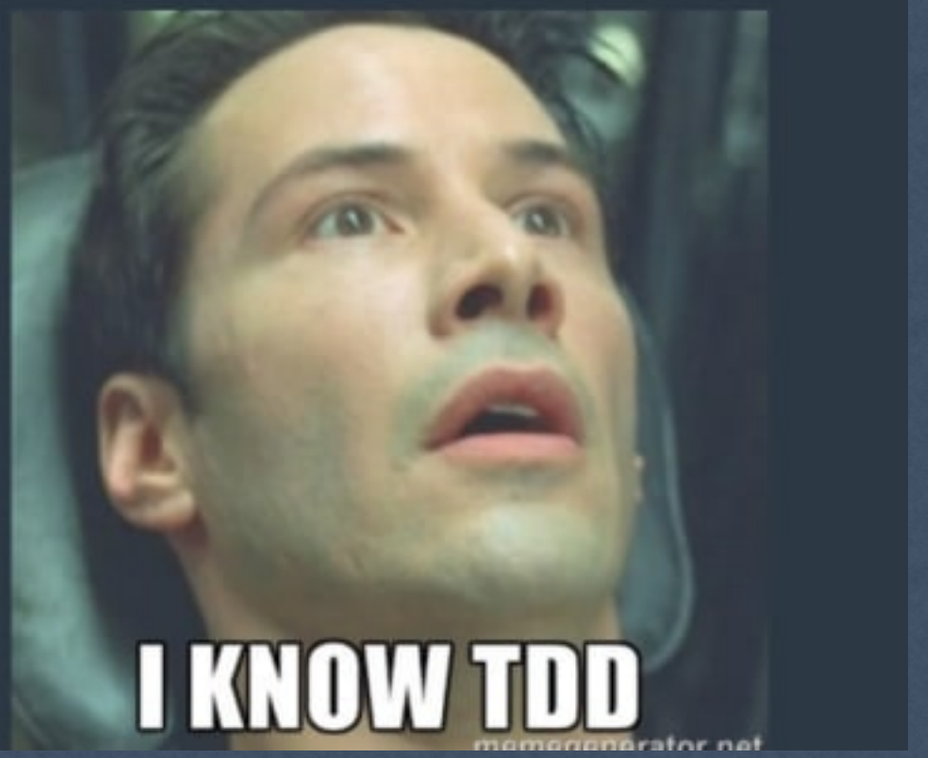
Box Testing,
Monkey Testing
Performance Testing
Security Testing
Negative Testing
...

The Legend


1. Write Test
2. Test Fails
3. Write Code
4. Test Passes
5. Refactor
TDD

TAKE THE RED PILL





Step 1: describe it #RDD 📖

```
describe('rendering', () => {
  it('should render a <TouchableOpacity/>')
  it('should render a label')
  describe('no type', () => {
    it('should have the default style')
  })
  describe('primary type', () => {
    it('should have the primary style')
  })
})

describe('interaction', () => {
  describe('clicking the button', () => {
    it('should call the onClick callback')
  })
})
```

Step 2: make it fail 🚨

```
// button.js
import React from 'react'

export default () => null

// button.test.js
import Button from './button'

const wrapper = shallow(<Button />)
it('should render a <TouchableOpacity/>', () => {
  expect(wrapper.find('TouchableOpacity')).toHaveLength(1)
})
```

Dan Abramov
@dan_abramov

Abonné

TDD paralyzes me. I'm all for writing tests early in the process — especially in library code. But I can't write them before I *play*. I need to write a shitty draft and play with the behavior to understand what I really want. Then rewrite guided by tests.

Traduire le Tweet

01:23 - 19 janv. 2019

1298 Retweets **6889** J'aime

321 1,3 k 6,9 k

Jon Eaves @joneaves · 19 janv.

En réponse à @dan_abramov @delfick

I completely agree. I wrote about this in 2004.

Traduire le Tweet

Spike to learn, TDD to build

I don’t consider myself a coding genius, and as my brain is fairly small when I start on new programming problems, I invariably come across things that I don’t...

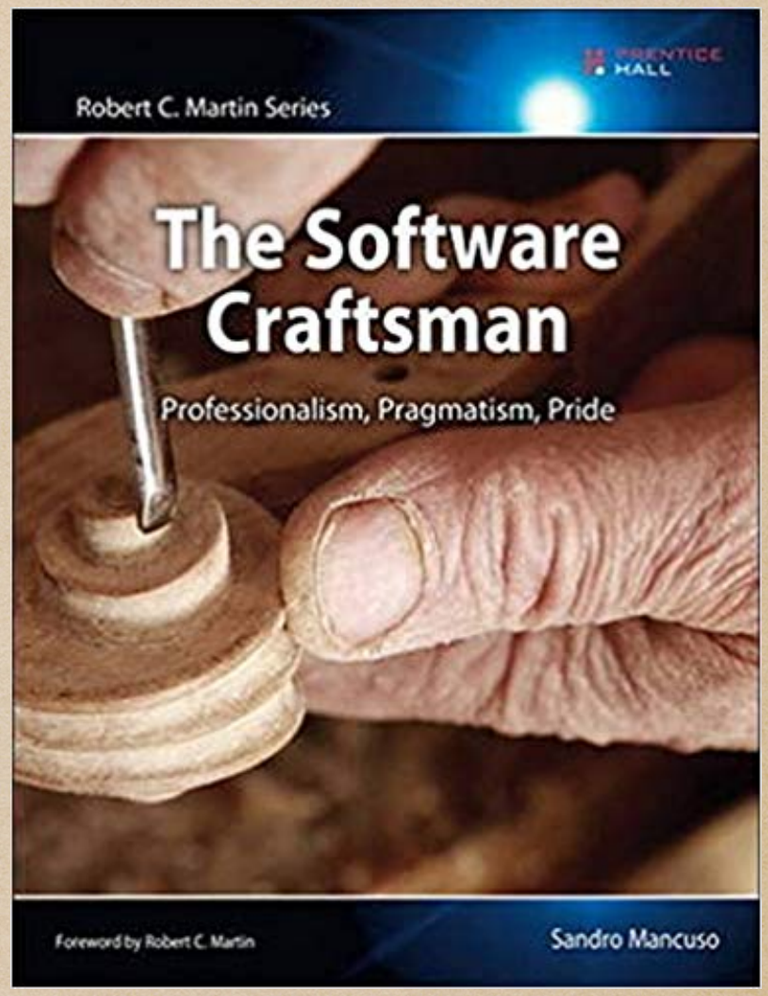
joneaves.wordpress.com

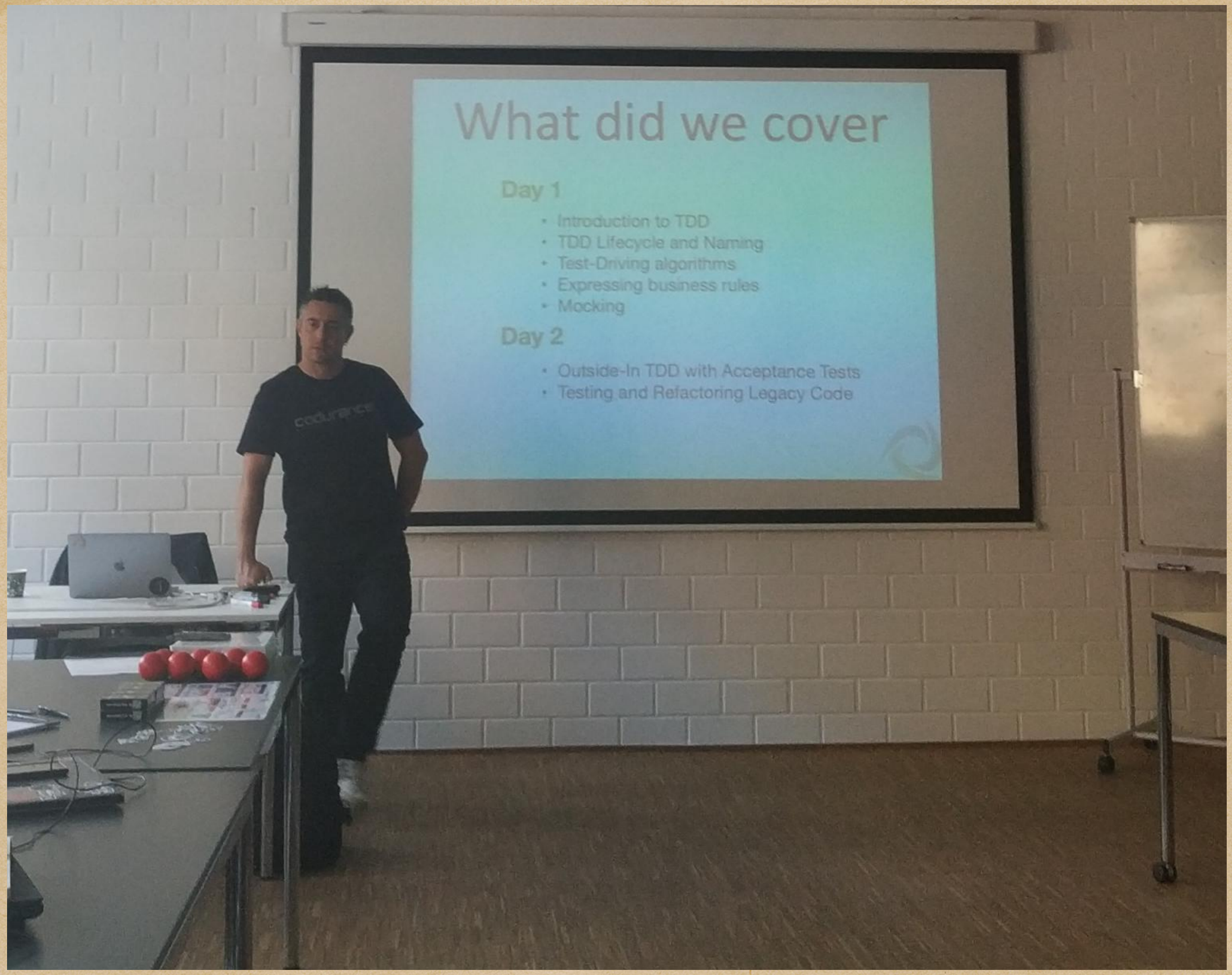
1 6

« TDD is a design activity,
and if you don't know how to design something,
you need to have a better understanding
of the problem and solution domain,
or you'll end up in a horrible mess. »

Jon Eaves

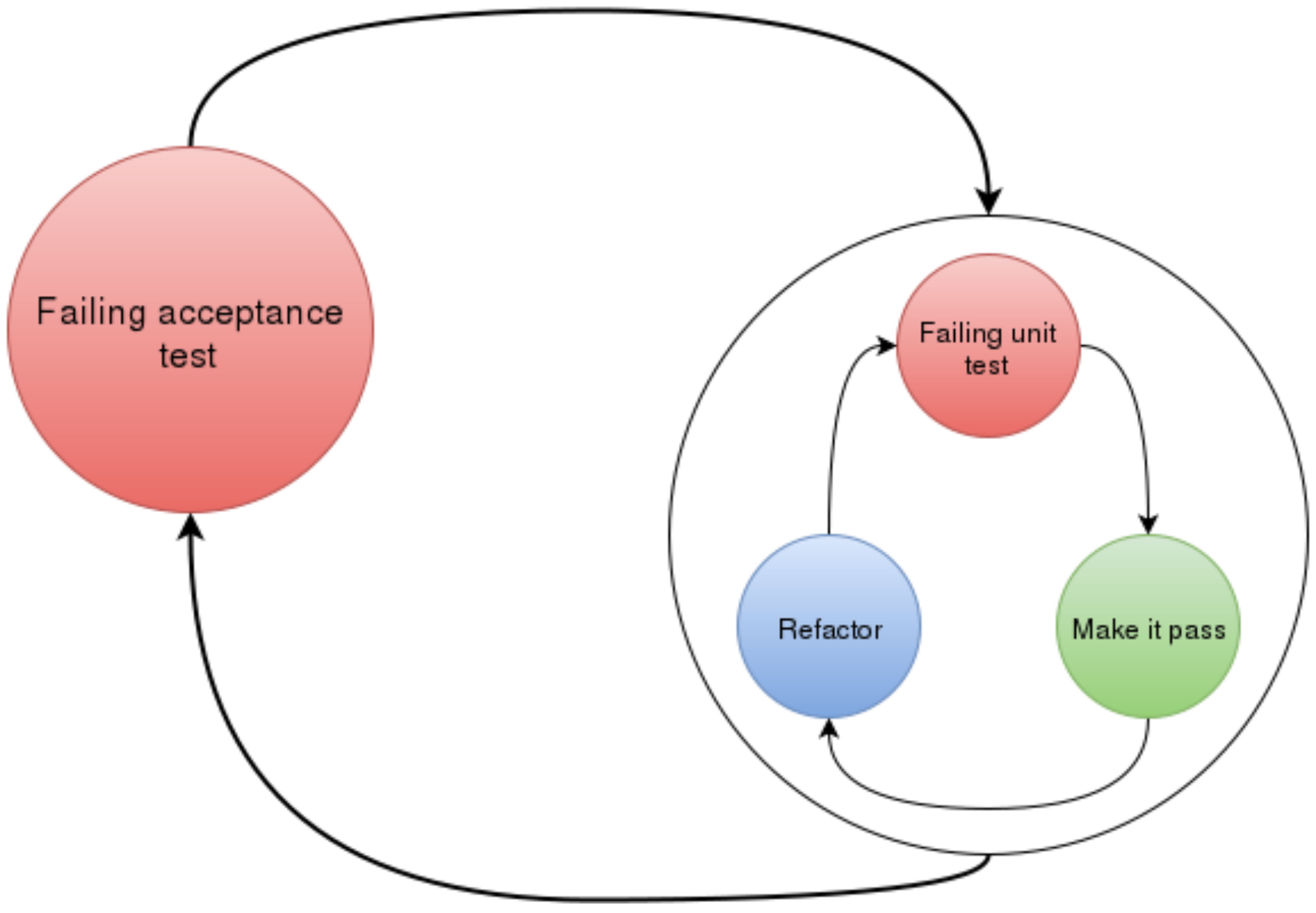

codurance
craft at heart


Robert C. Martin Series
PRENTICE HALL
The Software Craftsman
Professionalism, Pragmatism, Pride
Foreword by Robert C. Martin
Sandro Mancuso


What did we cover
Day 1
Introduction to TDD
TDD Lifecycle and Naming
Test-Driving algorithms
Expressing business rules
Mocking
Day 2
Outside-In TDD with Acceptance Tests
Testing and Refactoring Legacy Code

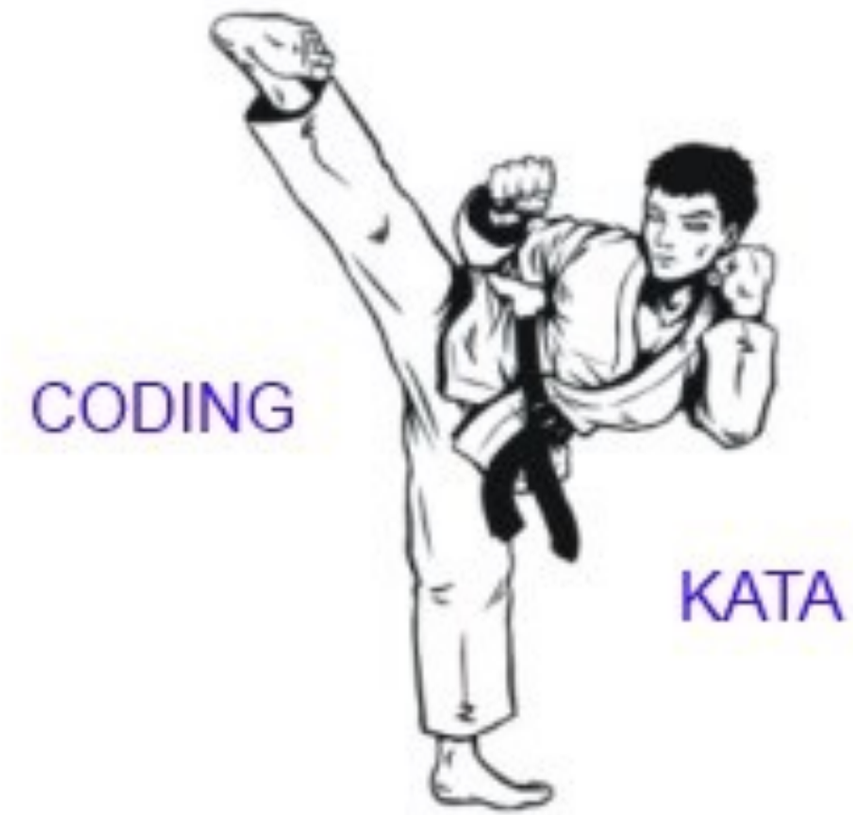
Outside-In TDD

1. Write a high-level (Outside) business value example that goes red
2. Write a lower-level (Inside) example for the first step of implementation that goes red
3. Implement the minimum code to pass that lower-level example, see it go green
4. Write the next lower-level example pushing towards passing step 1
5. Repeat steps 3 and 4 until the high-level test (1) goes green
6. Start over by writing a new high-level test


Failing acceptance test
Failing unit test
Refactor
Make it pass

When your low-level tests are green, you have permission to write new examples or refactor existing implementation.

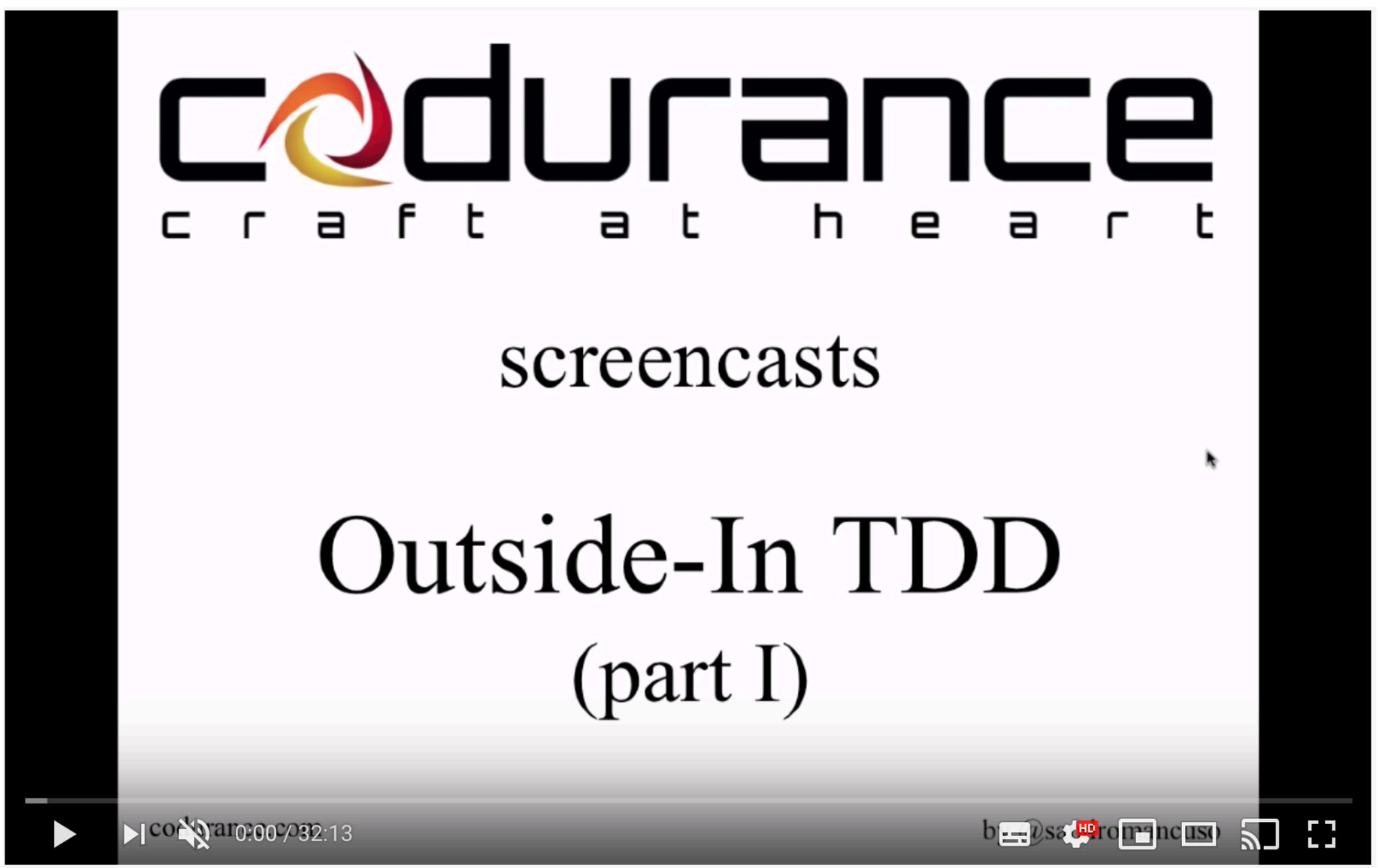
When your low-level tests are red, you have permission to write or change implementation code only for the purpose of making the existing tests go green.



https://github.com/sandromancuso/trip-service-kata/tree/master/javascript

https://codurance.com/2018/06/17/frontend-outside-in/

https://medium.com/craft-academy/outside-in-testing-in-react-bfb058ee4b7c



https://www.youtube.com/watch?v=XHnuMjah6ps

Conclusion

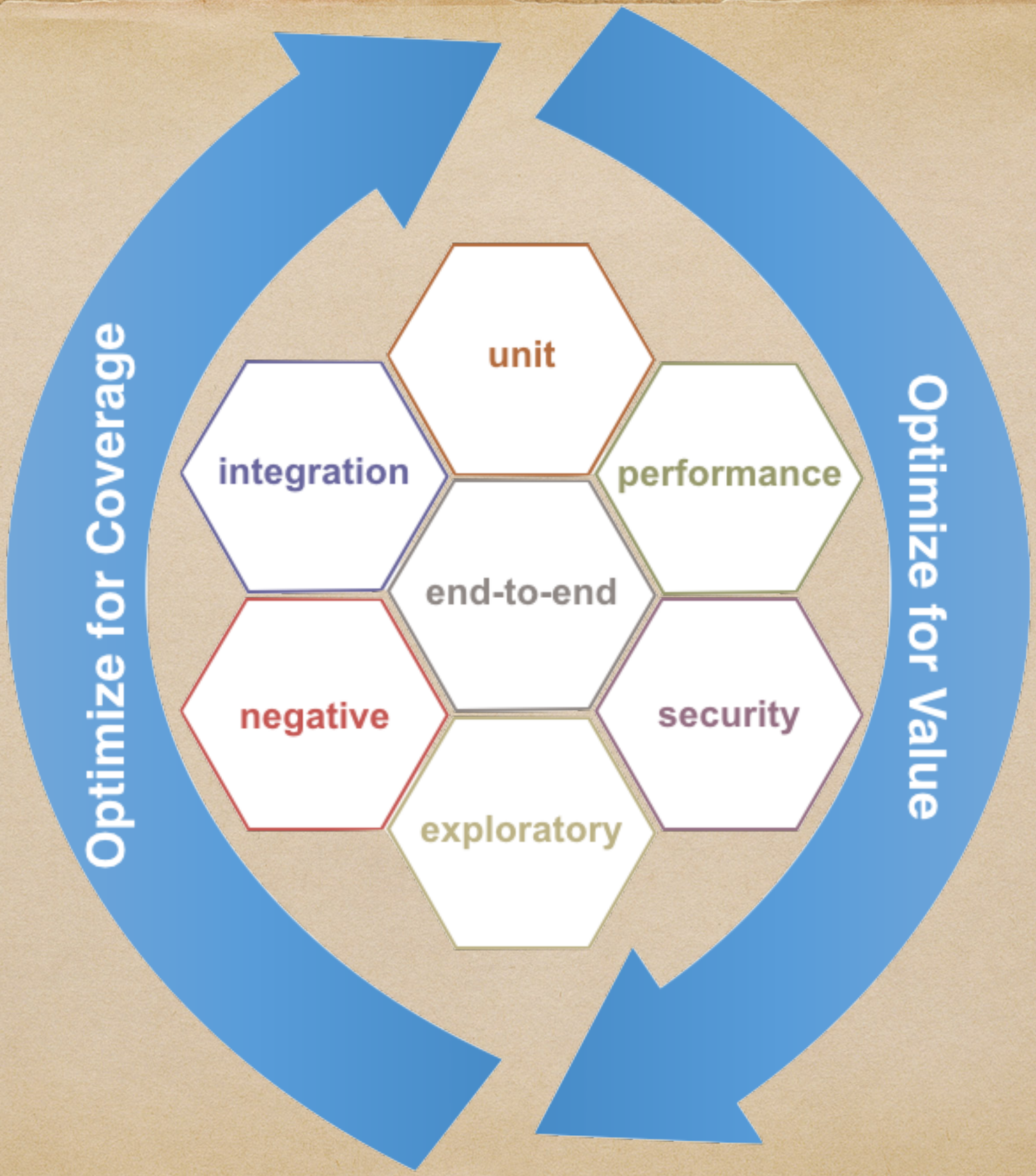
TDD is not a legend.

You need practice (kata)
and good specifications
(user story + examples).

«Testing is Good.
Pyramids are Bad.
Ice Cream Cones are the Worst. »

Stephen H Fishman

https://medium.com/@fistsOfReason/testing-is-good-pyramids-are-bad-ice-cream-cones-are-the-worst-ad94b9b2f05f


Optimize for Coverage
Optimize for Value
unit
integration
performance
end-to-end
negative
security
exploratory

Thank you.

Slides and source code will be posted tonight on
github.com/ou2S

LinkedIn: Oussama Ghalbzouri

Twitter : ou2s