iOS BDD with Kiwi

https://github.com/process255/insta-test

Sean Dougherty
sean@process255.com
@sdougherty

process255.com
simpletomato.com

Who am I?

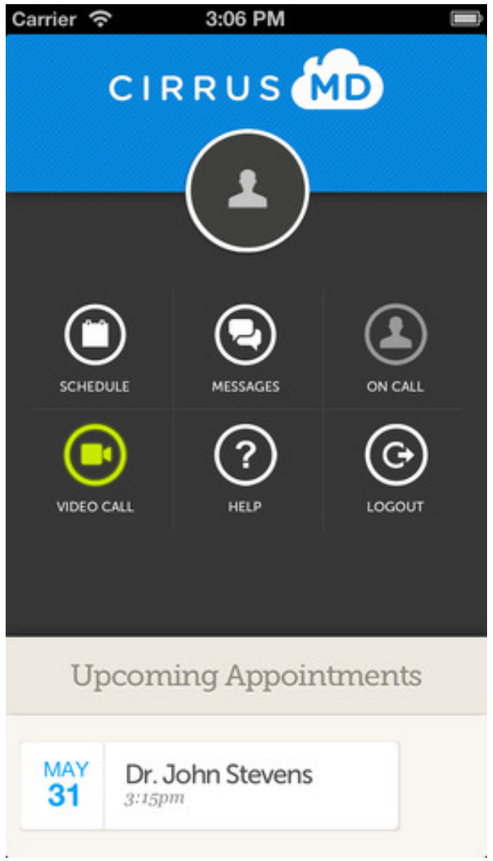
iOS engineer in Denver
writing iOS apps since 2010
web apps before that

process/255

WE BLEED RGB


Carrier
6:06 PM
snocru
Update Status
cru
near ME
resorts
tracking
9
newsfeed
chatter
snocru LEADERBOARDS
New!


GAME
TiME


Carrier
3:06 PM
CIRRUS MD
SCHEDULE
MESSAGES
ON CALL
VIDEO CALL
HELP
LOGOUT
Upcoming Appointments
MAY
31
Dr. John Stevens
3:15pm

iOS
Behavior Driven Development
with Kiwi

Insta-Test

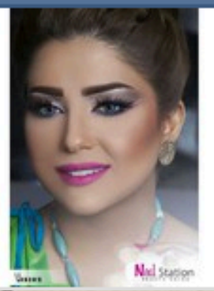
An iOS App that displays the feed of popular photos on Instagram.

https://github.com/process255/insta-test

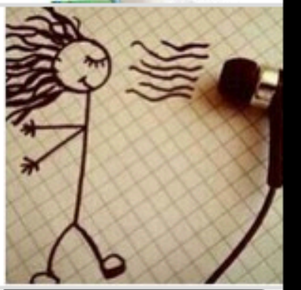
Carrier 10:06 PM

Popular

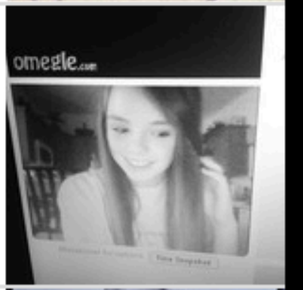
likes 11,378 :حصه اللوغاني



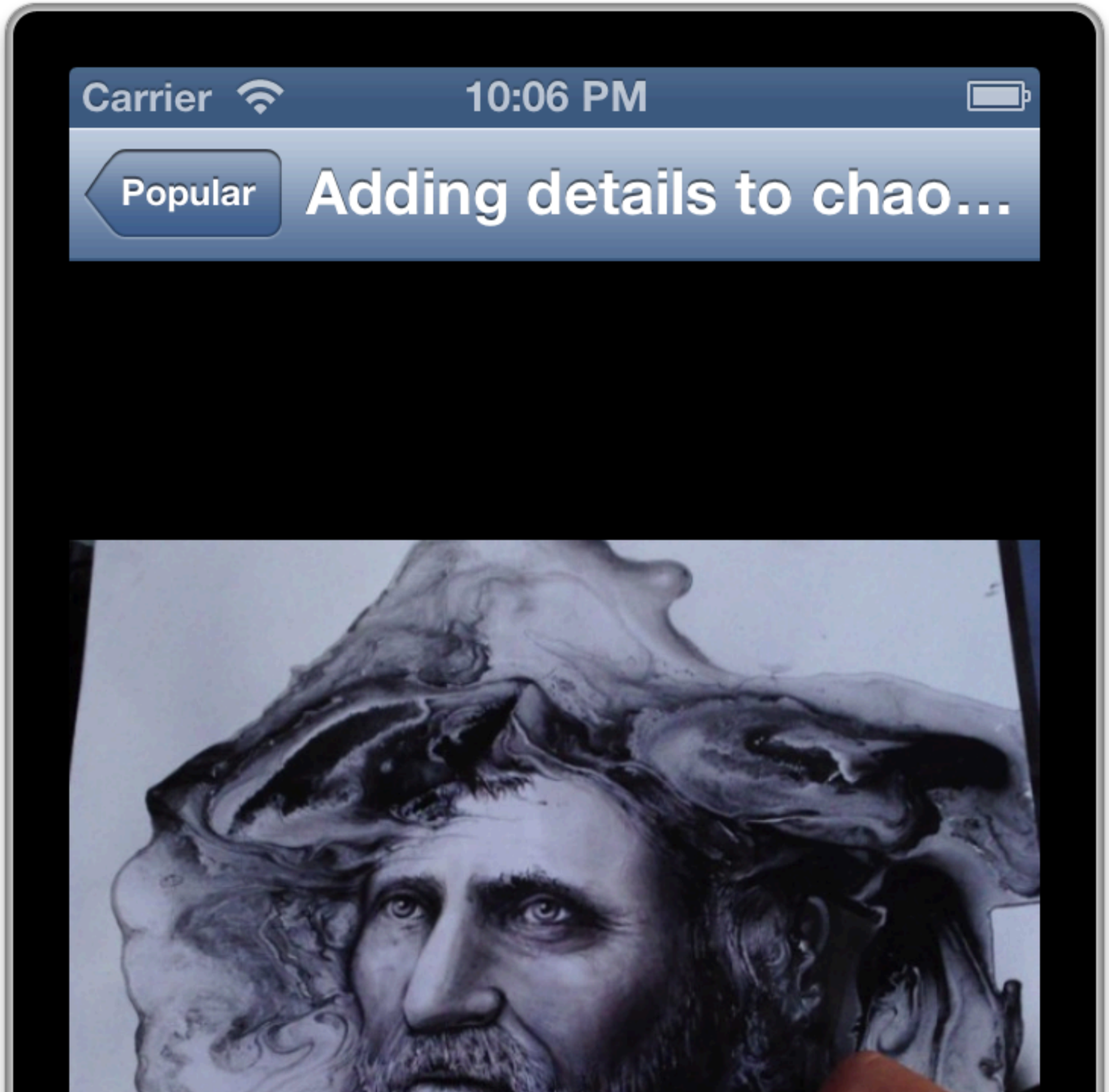
! TEST ILLUSION: 9,193 likes



Desiree Williams †: 851 likes



"Chaos Whisperer": 1,267 likes


Carrier
10:06 PM
Popular
Adding details to chao…

Culture of testing.

Ruby On Rails

NO Culture of testing.

Objective-C

Ok, ALMOST no Culture of testing.

But... that is changing.

Lots of Ruby developers are building iOS apps.

But... that is changing.

And they are bringing their tests with them.

What is TDD?

"Test driven development (TDD) is a software development approach in which a test is written before writing the code."

http://www.techopedia.com/

Why test our code?

“TDD encourages simple designs and inspires confidence.”

Kent Beck, who is credited with having developed or 'rediscovered' the technique.

Why test our code?

Testing is another tool in our toolbox that helps us build high quality software.

Why test our code?

A well maintained suite of tests gives
us confidence to refactor.

Why test our code?

Tests help new developers learn the code base and help prevent new code from unknowingly breaking old code.

Why test our code?

Tests help us write better code.
Poorly written code is hard to test. Test driving our code forces us to do better.

What is BDD?

Behavior Driven Development focuses and associates behavioral specifications with each unit of software under development.

http://www.techopedia.com/

What is the difference?

Apple's OCUnit follows the traditional xUnit format.

Kiwi uses a specification format made popular by Ruby's RSpec BDD library.

OCUnit

```
- (void)testHasElevenPlayers
{
    Team team = [Team team];
    STAssertTrue(team.players == 11, @"should have 11 players");
}
```

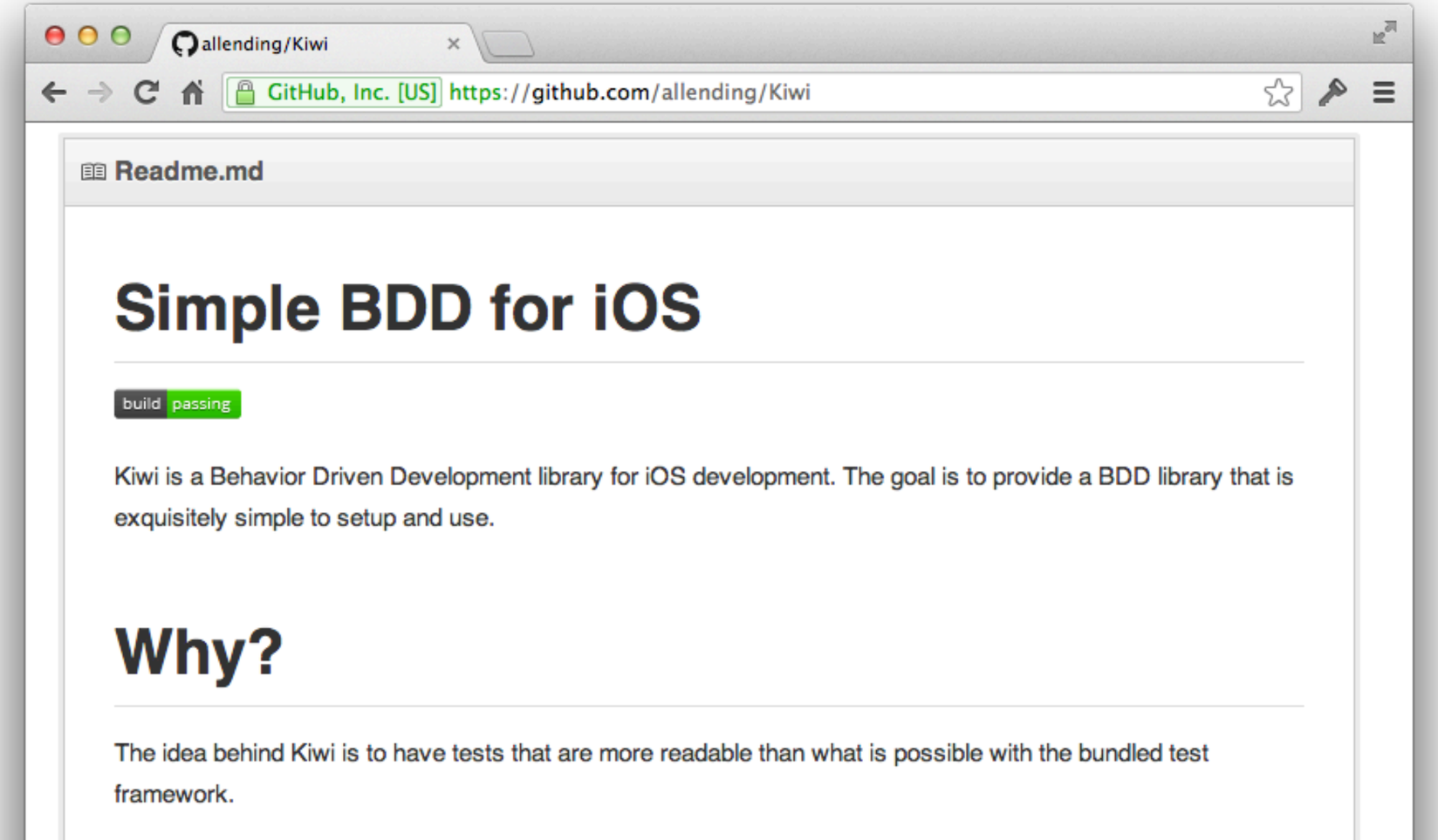
Kiwi

```
it(@"should have 11 players", ^{

   Team team = [Team team];
   [[[team should] have:11] players];

});
```

Kiwi


allending/Kiwi
GitHub, Inc. [US] https://github.com/allending/Kiwi
Readme.md
Simple BDD for iOS
build passing
Kiwi is a Behavior Driven Development library for iOS development. The goal is to provide a BDD library that is exquisitely simple to setup and use.
Why?
The idea behind Kiwi is to have tests that are more readable than what is possible with the bundled test framework.

What is it good at?

readable tests

testing asynchronous code

built in stubbing and mocking

partial mocking

What is it bad at?

helper methods

code reuse

clickable test failure messages

Kiwi specs read like a sentence.

The team, when newly created,
should have a name.

The team, when newly created, should have 11 players.

```
describe(@"Team", ^{

    context(@"when newly created", ^{

        it(@"should have a name", ^{

            Team team = [Team team];
            [[team.name should] equal:@"Avalanche"];

        });

        it(@"should have 11 players", ^{

            Team team = [Team team];
            [[[team should] have:11] players];

        });

    });

});
```

```
describe(@"Subject", ^{
    beforeAll(^{
        ...
    });

    afterAll(^{
        ...
    });

    beforeEach(^{
        ...
    });

    afterEach(^{
        ...
    });

});
```

A pragmatic approach

What do I want to test?

Simple methods

Methods that take time (async)

RestKit's Object Mapping

Testing Storyboards

A pragmatic approach

How can I test it?

Simple methods

```
- (NSString *)prettyTitle
```

```
context(@"-prettyTitle", ^{

   it(@"should return 'Sean Dougherty: 1,000 likes", ^{

      Instagram* instagram = [[Instagram alloc] init];
      instagram.fullName = @"Sean Dougherty";
      instagram.likeCount = 1000;

      [[[instagram prettyTitle] should] equal:@"Sean Dougherty: 1,000 likes"];

   });

});
```

Methods that take time (async)

```
- (void)loadPopularWithSuccess:failure:
```

```
it(@"should load 16 photos", ^{

   __block RKMappingResult *result;

   [service loadPopularWithSuccess:...)
   {
       result = mappingResult;
    }
   failure:nil];

   [[expectFutureValue([result array]) shouldEventually] haveCountOf:16];

});
```

RestKit's Object Mapping

```
specify(^{ [[mappingTest should] mapKeyPath:@"id"
                                  toKeyPath:@"instagramID"
                                  withValue:@"1"];});
```

Testing Storyboards

```
it(@"the tableView should exist", ^{
    [vc.tableView shouldNotBeNil];
});
```

Mocking & Stubbing

Mocking

```
__block InstaService *service;

beforeEach(^{

  service = [KWMock mockForClass:[InstaService
  class]];

});
```

Stubbing

```
beforeEach(^{

   Instagram *instagram = [[Instagram alloc] init];
    instagram.thumbPath = @"thumb path";
    instagram.prettyTitle = @"pretty title";

    [vc stub:@selector(instagrams) andReturn:@[instagram]];

});
```

Testing Private Methods and Private Properties

Use a class extension in your spec file

```
@interface InstaTableViewController ()

@property (nonatomic, copy) NSArray *instagrams;

- (void)loadPopular;

@end
```

Demo Time

Other Options

OCUnit (built into Xcode)

Cedar
https://github.com/pivotal/cedar

Expecta
https://github.com/specta/expecta

Specta
https://github.com/specta/specta

OCMock
https://github.com/erikdoe/ocmock

OCMockito
https://github.com/jonreid/OCMockito

Lots of others.

Resources

Insta-Test
https://github.com/process255/insta-test

Kiwi
https://github.com/allending/Kiwi

nsscreencast
http://nsscreencast.com/episodes/4-automated-testing-with-kiwi

Test Driving iOS Development with Kiwi by Daniel H Steinberg
https://itunes.apple.com/us/book/test-driving-ios-development/id502345143?mt=11

Attribution

Kiwi by Allen Ding
https://github.com/allending/Kiwi

AFNetworking by Matt Thompson and Scott Raymond
https://github.com/AFNetworking/AFNetworking

RestKit by Blake Watters
https://github.com/RestKit/RestKit

OHHTTPStubs by Olivier Halligon
https://github.com/AliSoftware/OHHTTPStubs

SDWebImage by Olivier Poitrey
https://github.com/rs/SDWebImage

SVProgressHUD by Sam Vermette
https://github.com/samvermette/SVProgressHUD

CocoaPods by Eloy Durán
http://cocoapods.org/

Thank You