Consuming Web APIs, the TDD way

Luis Solano
@luisobo
Pixable

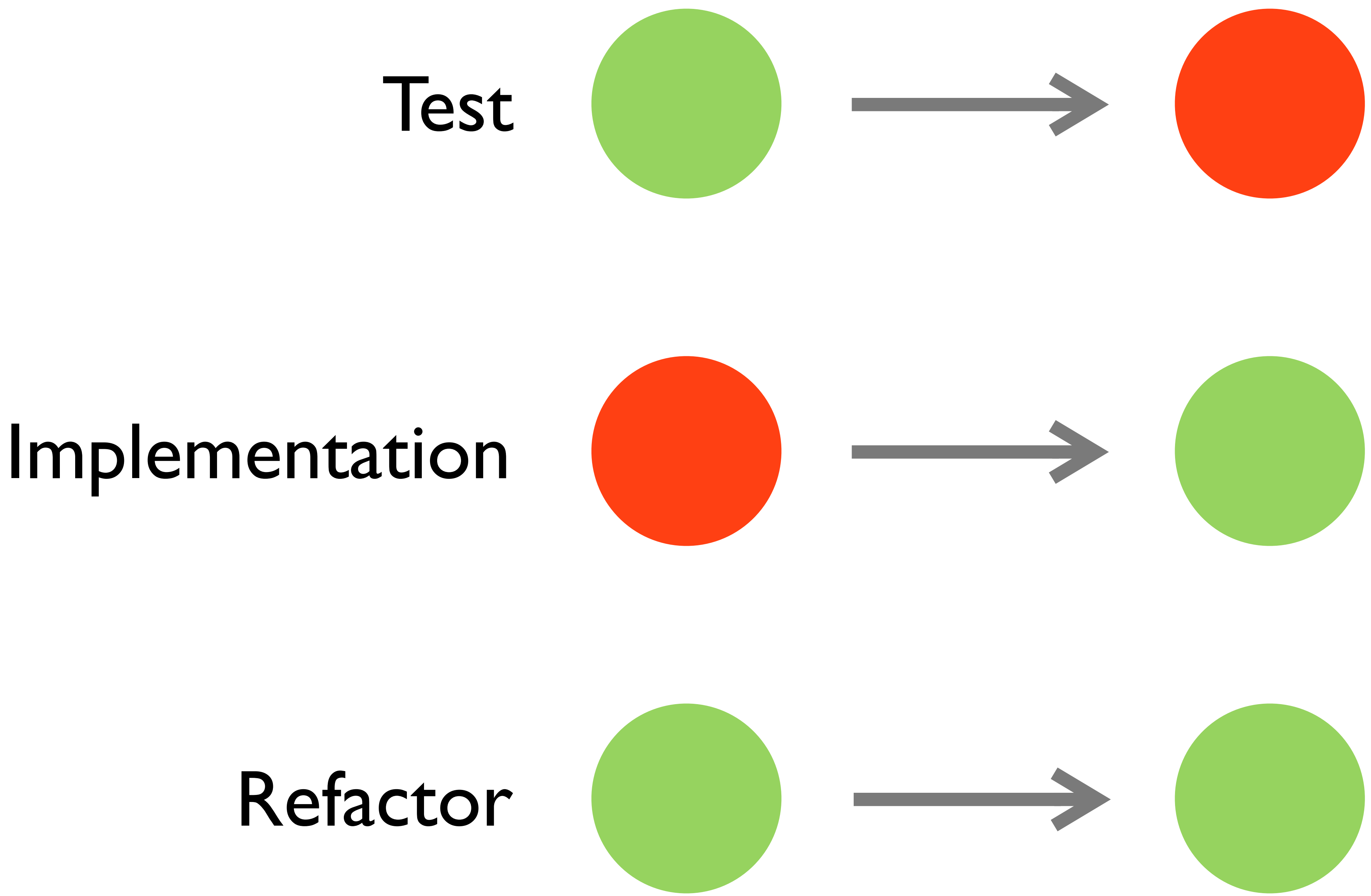
Good practices for networking code

Tackle real life testing scenarios

Underlaying principles of TDD

Error handling

Test-driven development


Test
Implementation
Refactor

Design tool

Tests are a nice side effect of TDD

Short feedback loop

Enable us to modify our code

API Client design

Retries

Business error handling

Business logic

State

Abstraction
API Client

HTTP

Caching

Authentication

Background

Parsing

Content type negotiation

HTTP error handling

GET /dogs.json


```json
{
  "dogs": [{
    "name": "perro",
    "color": "brown"
  },{
    "name": "tomas",
    "color": "black"
  }]
}
```

Our first test

```
it(@"retrieves dogs", ^{

    __block NSArray *capturedDogs = nil;
    [[LSBDogCare new] allDogs:^(NSArray *dogs) {
        capturedDogs = dogs;
    } failure:^(NSError *error) {
    }];


    [[expectFutureValue(capturedDogs) shouldEventually]
      equal:@[@{@"name":@"perro",@"color":@"brown"},
              @{@"name":@"tomas",@"color":@"black"}
    ]];
});
```

```objc
@implementation LSBDogCare

- (void)allDogs:(void(^)(NSArray *dogs))success
        failure:(void(^)(NSError *error))failure {

    NSURL *url = [NSURL URLWithString:@"http://api.example.com/dogs.json"];
    NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url];

    AFJSONRequestOperation *op;
    op = [AFJSONRequestOperation JSONRequestOperationWithRequest:request
       success:^(NSURLRequest *request,
                 NSHTTPURLResponse *response,
                 NSDictionary *JSON) {

         success(JSON[@"dogs"]);

       } failure:^(NSURLRequest *request,
                   NSHTTPURLResponse *response,
                   NSError *requestError, id JSON) {
    }];

    [op start];

}

@end
```

```
it(@"retrieves dogs", ^{

    __block NSArray *capturedDogs = nil;
    [[LSBDogCare new] allDogs:^(NSArray *dogs) {
        capturedDogs = dogs;
    } failure:^(NSError *error) {
    }];

    [[expectFutureValue(capturedDogs) shouldEventually]
      equal:@[@{@"name":@"perro",@"color":@"brown"},
              @{@"name":@"tomas",@"color":@"black"}
    ]];

});
```

Isolate tests from **undeterministic** and **slow** depencencies.

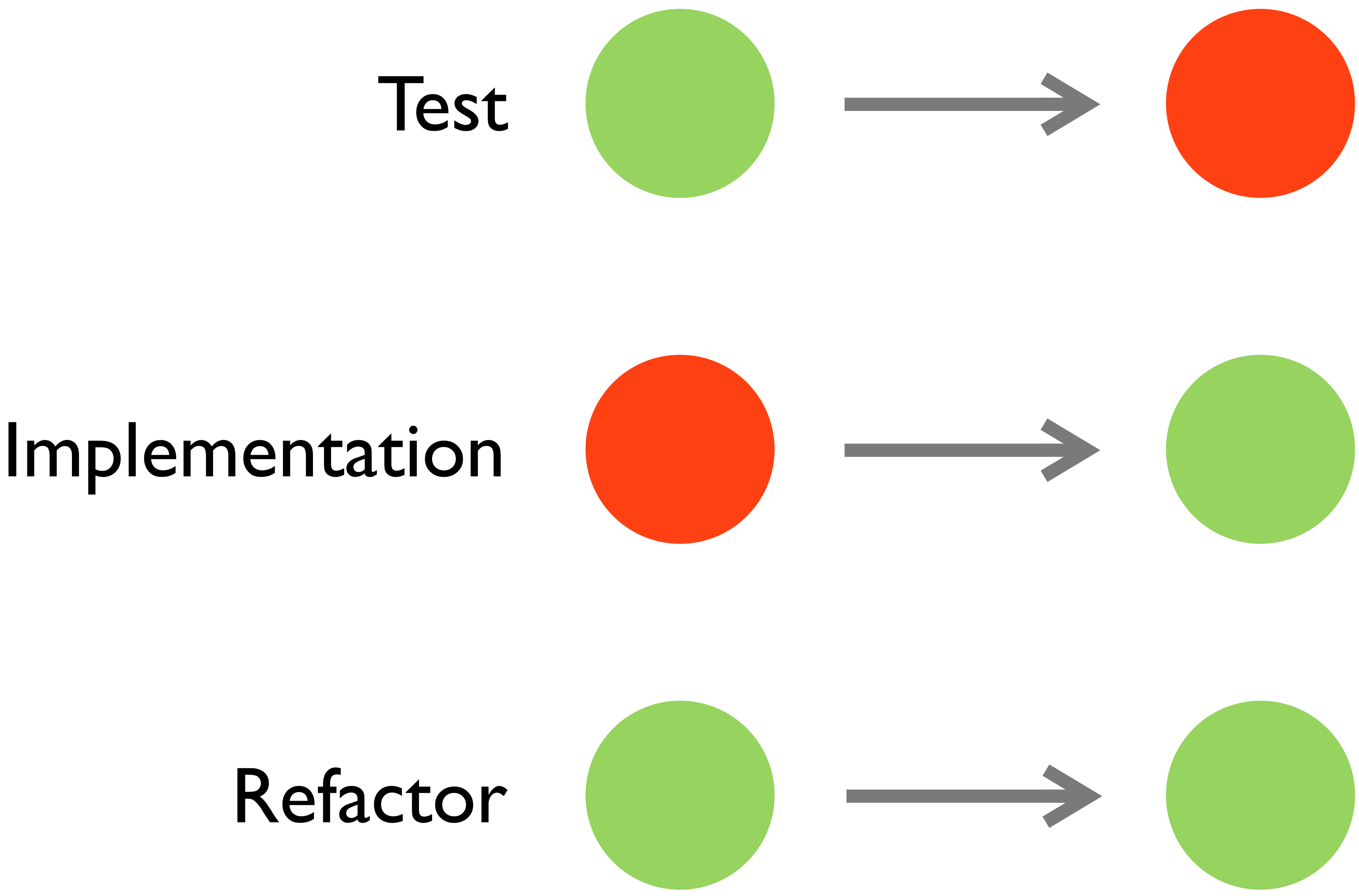
```
it(@"retrieves dogs", ^{
     AFJSONRequestOperation *mockOp = [AFJSONRequestOperation mock];

     __block NSURLRequest *capturedRequest;
     __block void (^capturedSuccess)(NSURLRequest *request,
                                     NSHTTPURLResponse *response,
                                     id JSON);
    [AFJSONRequestOperation
            stub:@selector(JSONRequestOperationWithRequest:success:failure:)
            withBlock:^id(NSArray *params) {
                capturedRequest = params[0];
                capturedSuccess = params[1];
                return mockOp;
        }];


    [mockOp stub:@selector(start) withBlock:^id(NSArray *params) {
        capturedSuccess(capturedRequest,
                        [NSHTTPURLResponse mock],
                        @{@"dogs":@[@{@"name":@"perro",@"color":@"brown"}]});
        return nil;
    }];

    __block NSArray *capturedDogs = nil;
    [[LSBDogCare new] allDogs:^(NSArray *dogs) {
        capturedDogs = dogs;
    } failure:^(NSError *error) {

    }];

    [[expectFutureValue(capturedDogs) shouldEventually]
      equal:@[@{@"name":@"perro",@"color":@"brown"}]];

});
```


ONE DOES NOT SIMPLY
STUB IMPLEMENTATION DETAILS

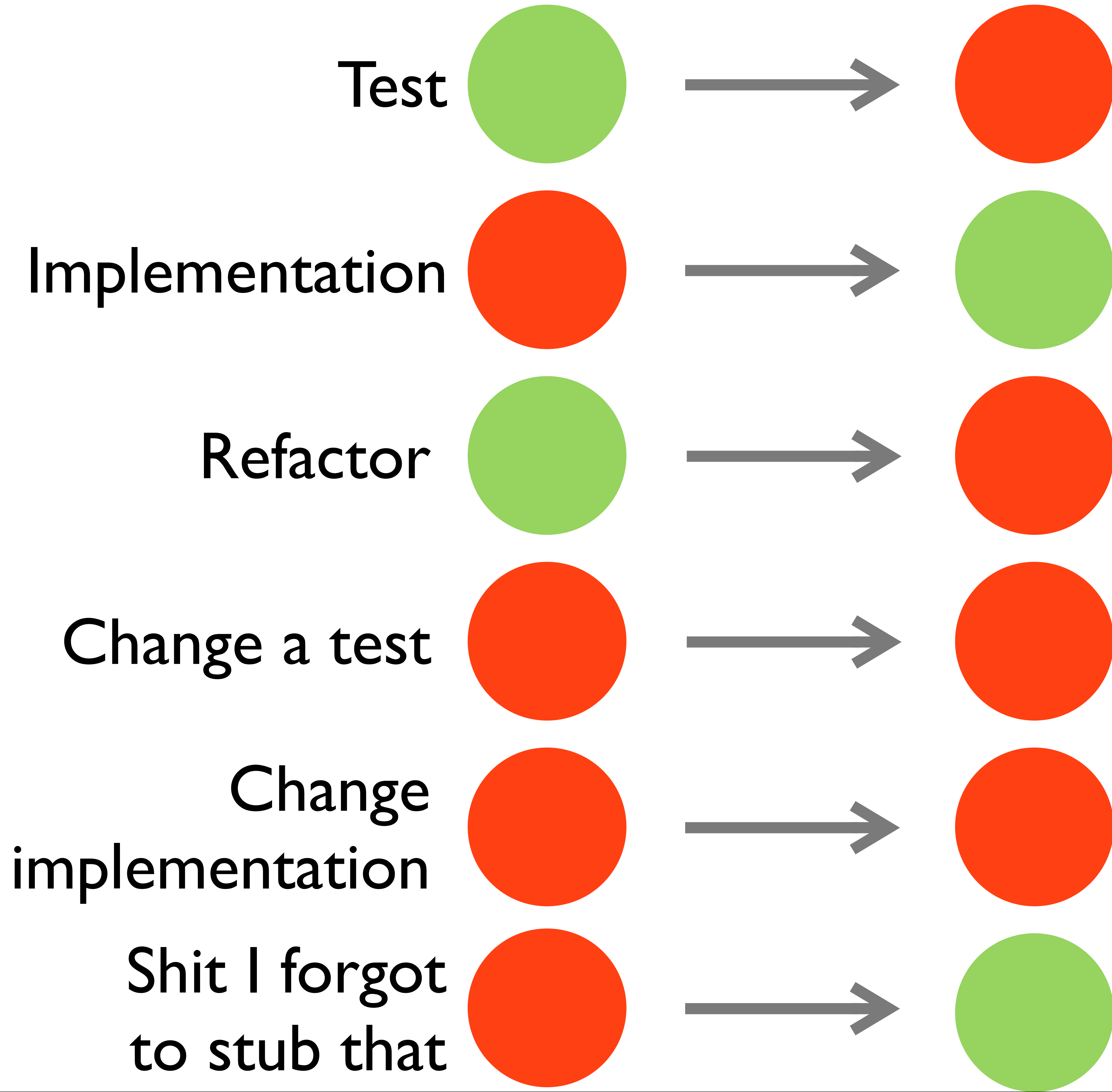
Don't couple **test** with **implementation details**

Tests exist to let us **modify our code.**

What my coworkers think I do



What I actually do



```
$ git reset --hard
```

Don’t modify **test** and **implementation** at the same time

```
it(@"retrieves dogs", ^{
     AFJSONRequestOperation *mockOp = [AFJSONRequestOperation mock];

     __block NSURLRequest *capturedRequest;
     __block void (^capturedSuccess)(NSURLRequest *request,
                                     NSHTTPURLResponse *response,
                                     id JSON);
    [AFJSONRequestOperation
            stub:@selector(JSONRequestOperationWithRequest:success:failure:)
            withBlock:^id(NSArray *params) {
                capturedRequest = params[0];
                capturedSuccess = params[1];
                return mockOp;
        }];


    [mockOp stub:@selector(start) withBlock:^id(NSArray *params) {
        capturedSuccess(capturedRequest,
                        [NSHTTPURLResponse mock],
                        @{@"dogs":@[@{@"name":@"perro",@"color":@"brown"}]});
        return nil;
    }];

    __block NSArray *capturedDogs = nil;
    [[LSBDogCare new] allDogs:^(NSArray *dogs) {
        capturedDogs = dogs;
    } failure:^(NSError *error) {

    }];

    [[expectFutureValue(capturedDogs) shouldEventually]
      equal:@[@{@"name":@"perro",@"color":@"brown"}]];

});
```

README.md

Nocilla build passing

Stunning HTTP stubbing for iOS and OS X. Testing HTTP requests has never been easier.

```
it(@"retrieves dogs", ^{
    stubRequest(@"GET", @"http://api.example.com/dogs.json")
    .andReturn(200)
    .withHeaders(@{
                @"Content-Type": @"application/json;charset=utf-8"
    })
    .withBody([@{@"dogs":@[
                 @{@"name":@"perro",@"color":@"brown"},
                 @{@"name":@"tomas",@"color":@"black"}]
             } JSONString]);

    __block NSArray *capturedDogs = nil;
    [[LSBDogCare new] allDogs:^(NSArray *dogs) {
        capturedDogs = dogs;
    } failure:^(NSError *error) {

    }];

    [[expectFutureValue(capturedDogs) shouldEventually]
      equal:@[@{@"name":@"perro",@"color":@"brown"},
              @{@"name":@"tomas",@"color":@"black"}]];
});
```

```
context(@"when retrieving dogs fail because access was revoked", ^{
    it(@"reports an error", ^{

        stubRequest(@"GET", @"http://api.example.com/dogs.json")
        .andReturn(401);

        __block NSError *capturedError;
        [[LSBDogCare new] allDogs:^(NSArray *dogs) {
        } failure:^(NSError *error) {
            capturedError = error;
        }];

        [[expectFutureValue(capturedError) shouldEventually]
          equal:[NSError errorWithDomain:@"com.luisobo.dogcare"
                code:23
                userInfo:@{
                  NSLocalizedDescriptionKey:
                  @"uh uh uh, you didn't say the magic word"
                }]];

    });
});
```

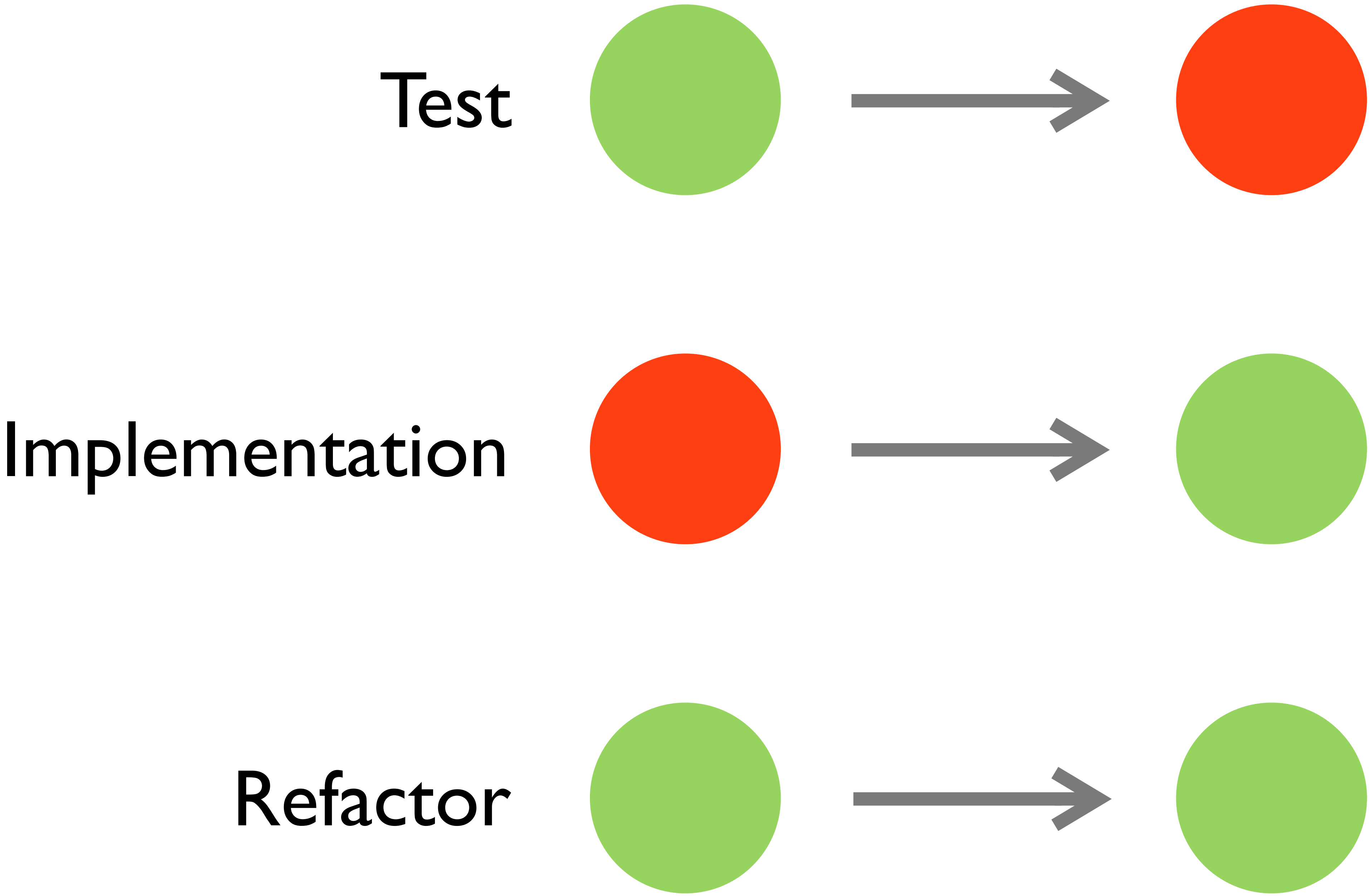
```
- (void)allDogs:(void(^)(NSArray *dogs))success
        failure:(void(^)(NSError *error))failure {

    // ...
    op = [AFJSONRequestOperation JSONRequestOperationWithRequest:request
       success:^(NSURLRequest *request,
                 NSHTTPURLResponse *response,
                 NSDictionary *JSON) {
        success(JSON[@"dogs"]);
       } failure:^(NSURLRequest *request,
               NSHTTPURLResponse *response,
               NSError *error, id JSON) {
         if (response.statusCode == 401) {
               error = [NSError errorWithDomain:@"com.luisobo.dogcare"
                        code:23
                        userInfo:@{
                          NSLocalizedDescriptionKey:
                          @"uh uh uh, you didn't say the magic word"
                        }];
           }
          failure(error);
    }];

    [op start];
}
```

```
context(@"when the request fails because there is no internet", ^{
    it(@"reports an error", ^{
        stubRequest(@"GET", @"http://api.example.com/dogs.json")
        .andFailWithError([NSError errorWithDomain:NSURLErrorDomain
                                              code:NSURLErrorCannotConnectToHost
                                          userInfo:nil]);

        __block NSError *capturedError;
        [[LSBDogCare new] allDogs:^(NSArray *dogs) {
        } failure:^(NSError *error) {
            capturedError = error;
        }];

        [[expectFutureValue(capturedError) shouldEventually]
          equal:[NSError errorWithDomain:@"com.luisobo.dogcare"
                 code:446
                 userInfo:@{
                   NSLocalizedDescriptionKey:
                   @"eeeer, check your internet dumbass"
                 }]];

    });
});
```

README.md

Nocilla build passing

Stunning HTTP stubbing for iOS and OS X. Testing HTTP requests has never been easier.


Test
Implementation
Refactor

https://github.com/luisobo/Nocilla

Error handling

The operation could not be completed. (com.facebook.sdk error 2)

NSError = domain + code

1. Developer screw-ups
2. Automatic recoverable errors (e.g. reauthorize)
3. User recoverable errors (e.g. validation, ask for more permissions)
4. Known non-recoverable errors. (e.g. server is down, rate limiting)
5. “How the fu*k did I get here?” errors

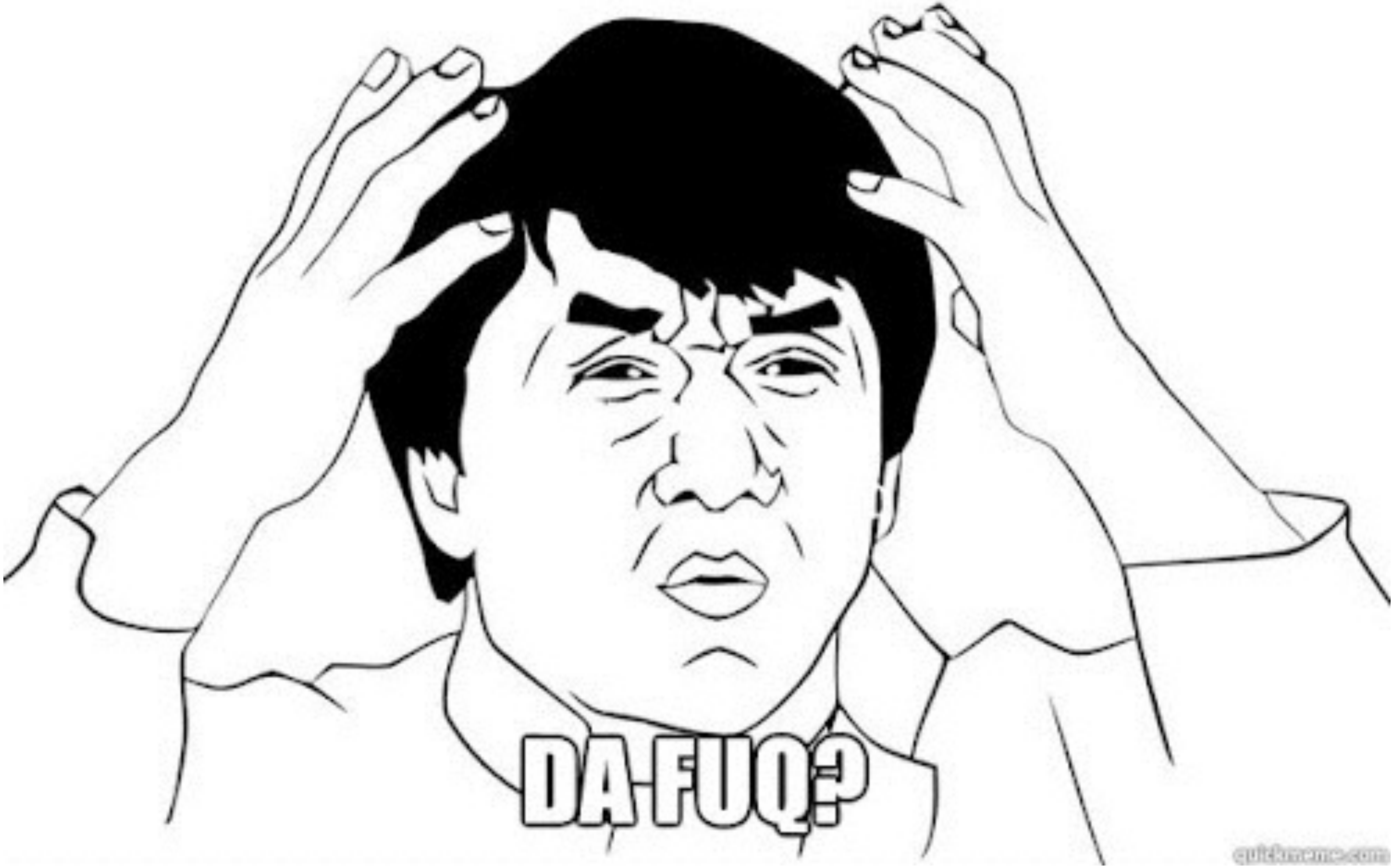
Group errors based on how the **client** is going to **recover** from them

codes

Name	Code	Recovery Tactic	Note
OAuth	190	C unless a subcode is present	Most OAuth errors include a subcode. Use the subcode table to determine the correct recovery tactic.
API Session	102	C unless a subcode is present	Most OAuth errors include a subcode. Use the subcode table to determine the correct recovery tactic.
API Unknown	1	A	Server-side problem; app should retry after waiting, up to some app-defined threshold
API Service	2	A	Server-side problem; app should retry after waiting, up to some app-defined threshold
API Too Many Calls	4	A	Server-side throttling; app should retry after waiting
API User Too Many Calls	17	A	Server-side throttling; app should retry after waiting
API Permission Denied	10	D	User either has not granted a permission or removed a permission
API Permission (range)	200-299	D	User either has not granted a permission or removed a permission

subcodes

Name	Code	Recovery Tactic	Note
App Not Installed	458	C	User removed the app from user settings
User Checkpointed	459	B	User needs to log onto www.facebook.com, or m.facebook.com
Password Changed	460	B (iOS 6) or C	On iOS 6, if the user is authorized using integrated authentication, the user should be directed to Facebook settings on the device to set the new password; otherwise the user needs to reauthorize
Expired	463	C	Token has expired and a new one needs to be requested
Unconfirmed User	464	B	User needs to log onto www.facebook.com, or m.facebook.com
Invalid access token	467	C	Token is invalid and a new one needs to be requested


DA FUQ?
quickmeme.com

A. **Retry the operation after waiting**
   Example: code 17 implies too many user calls, and an application should pause and retry
B. **Notify the user of an out-of-band action that is required**
   Example: on iOS 6 a password change means that the user must update the password for the device in order to re-enable app tokens
C. **Authorize/reauthorize the user**
   Example: an expired token requires reauthorization of the application by the user
D. **Request permissions**
   Example: the user may have removed a permission for an application after having previously granted it - so the app must re-request the permission

```
@interface NSError (FacebookAPI)
+ (instancetype)facebookAPIErrorWithResponse:(NSDictionary *)response;

- (BOOL)isFacebookAPIError;

- (BOOL)shouldRecoverByReauthorizing;
- (BOOL)shouldRecoverByRequestingMorePermissions;
- (BOOL)shouldRecoverByNotifyingUser;
@end
```

```
failure:^(NSURLRequest *request, NSHTTPURLResponse *response,
              NSError *error, id JSON) {

    NSError *facebookError = [NSError facebookAPIErrorWithResponse:JSON];

    if ([facebookError shouldRecoverByReauthorizing]) {
        // Reauthorize
    } else if ([facebookError shouldRecoverByRequestingMorePermissions]) {
        // Request more permissions
    } else if ([facebookError shouldRecoverByNotifyingUser]) {

    }


}
```

```
@interface NSError (FacebookSDK)
+ (instancetype)facebookSDKErrorWithError:(NSError *)error;

- (BOOL)isFacebookSDKError;

// ... Same deal
@end
```

```
failure:^(NSURLRequest *request, NSHTTPURLResponse *response,
              NSError *error, id JSON) {

    NSError *facebookError = [NSError facebookAPIErrorWithResponse:JSON];

    if ([facebookError shouldRecoverByReauthorizing]) {
        // Reauthorize
    } else if ([facebookError shouldRecoverByRequestingMorePermissions]) {
        // Request more permissions
    } else if ([facebookError shouldRecoverByNotifyingUser]) {
       failure([NSError facebookSDKErrorWithError:error]);
    } else {
       failure([NSError genericError]);
       // Ensure consistent state
       // Log facebookError in a remote service
    }
   }
```

Charles Proxy

charlesproxy.com

Nocilla

Integration tests

Build an abstraction on top of of an stateless wrapper

Isolate test from undeterministic and slow dependencies.

Don’t modify implementation and tests at the same time.

Don’t couple test with implementation details

Group errors by recover strategy

Thanks

Say hi @luisobo