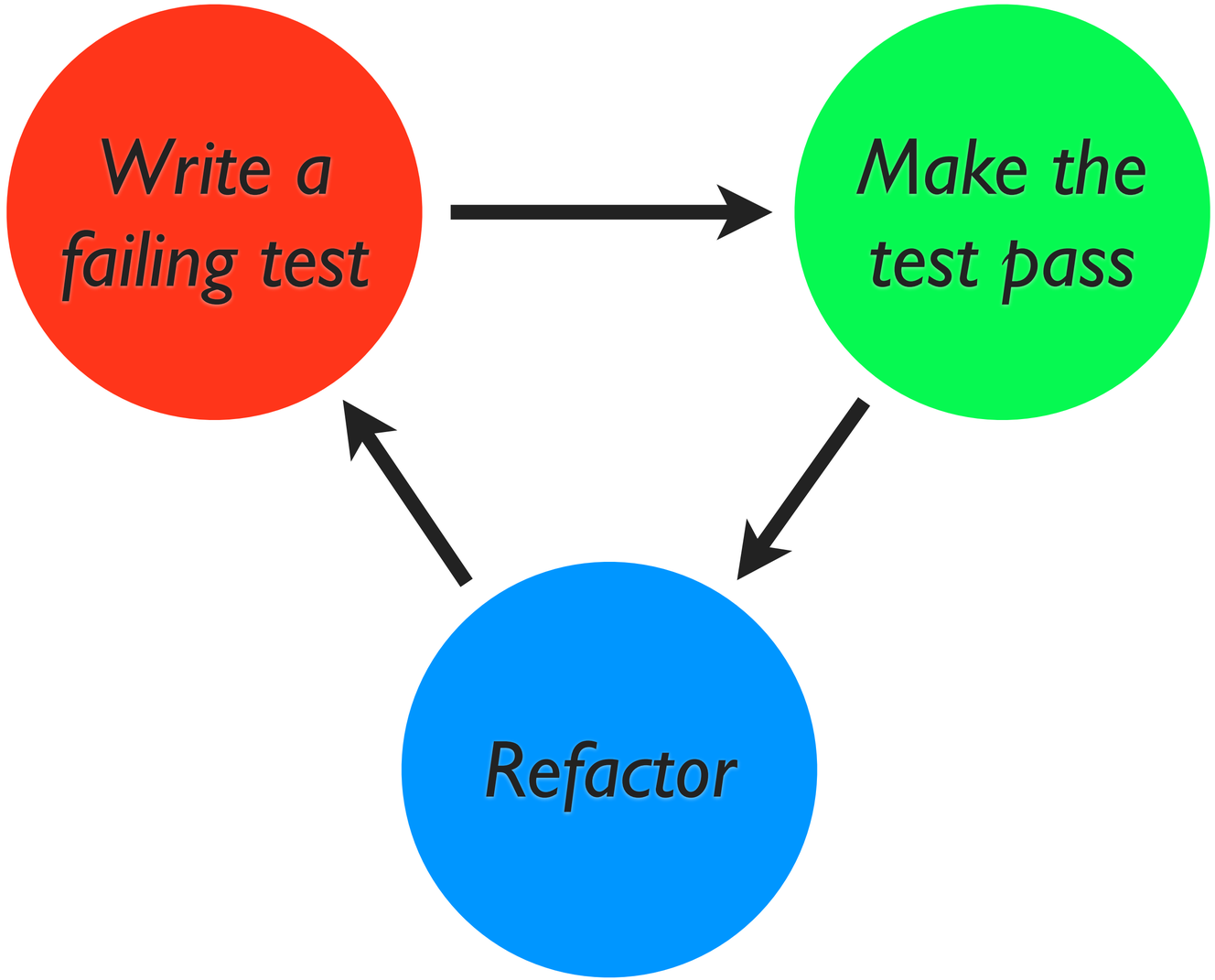
Test Driven Development

…by Controlling Dependencies

Jon Reid

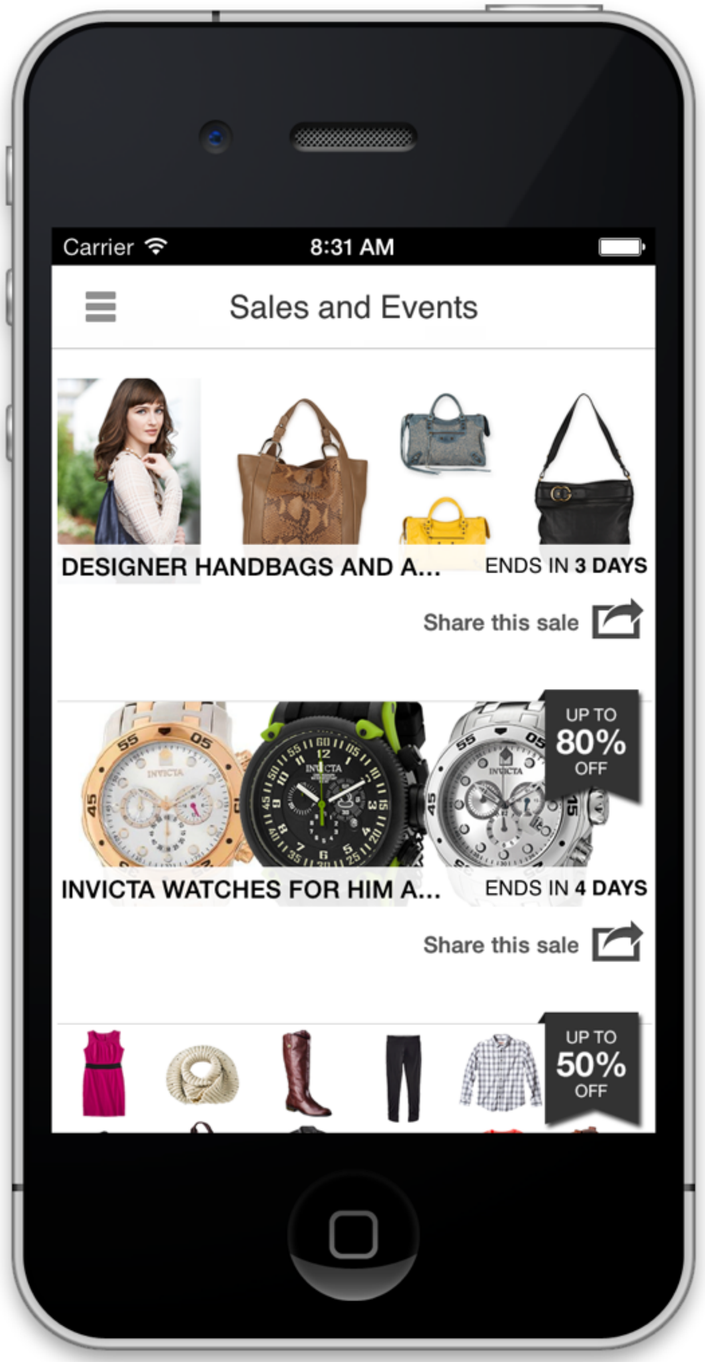

Write a failing test
Make the test pass
Refactor

UI?

Networking?

UI!

Networking!



3 Types of Unit Tests:

Return Value Test

State Test

Interaction Test

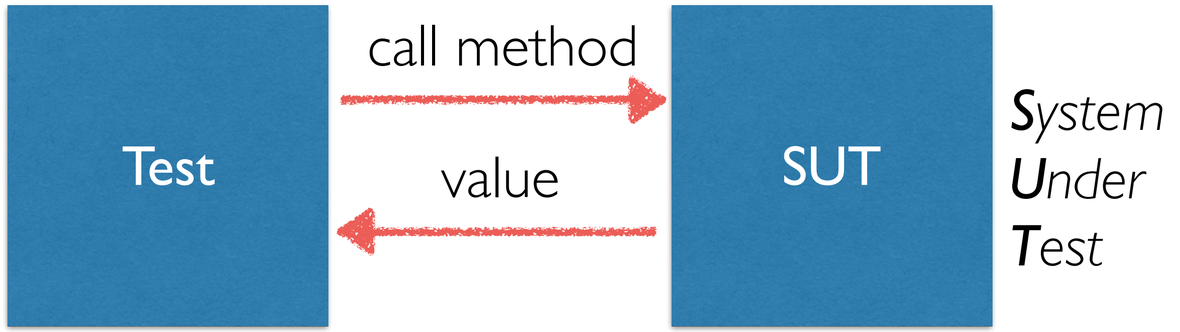
http://www.flickr.com/photos/jsmoorman/2298671281

Return Value Test

Arrange: Set up object

Act: Call method that returns a value

Assert: Compare against expected value

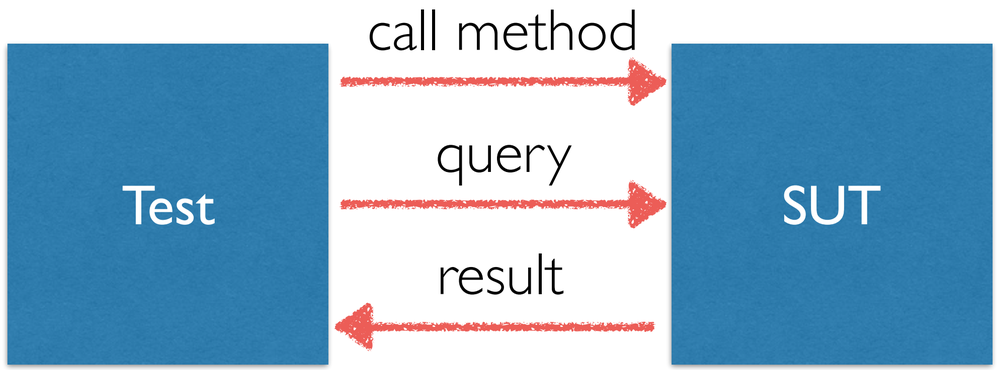


State Test

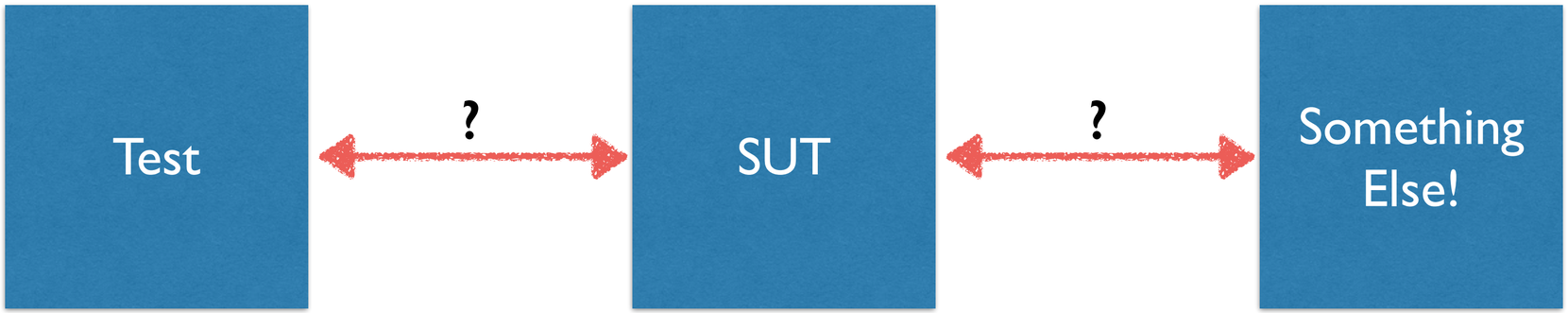
Arrange: Set up object

Act: Call method

Assert: Query object in some way



Interaction Test



Interaction Test

ProcessOrder

GetDishes

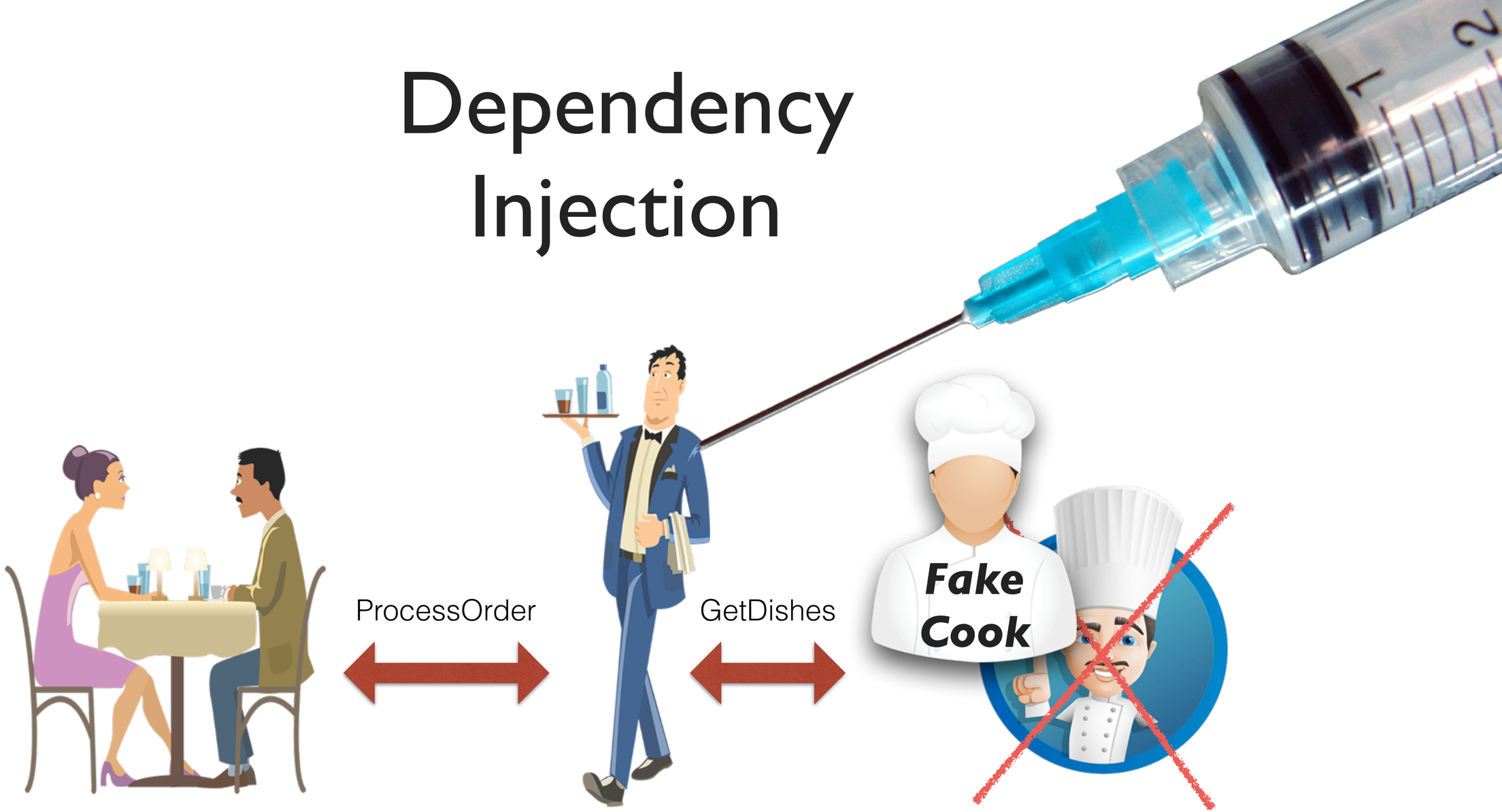
Table

Waiter

Cook

Interaction Test

Dependency Injection



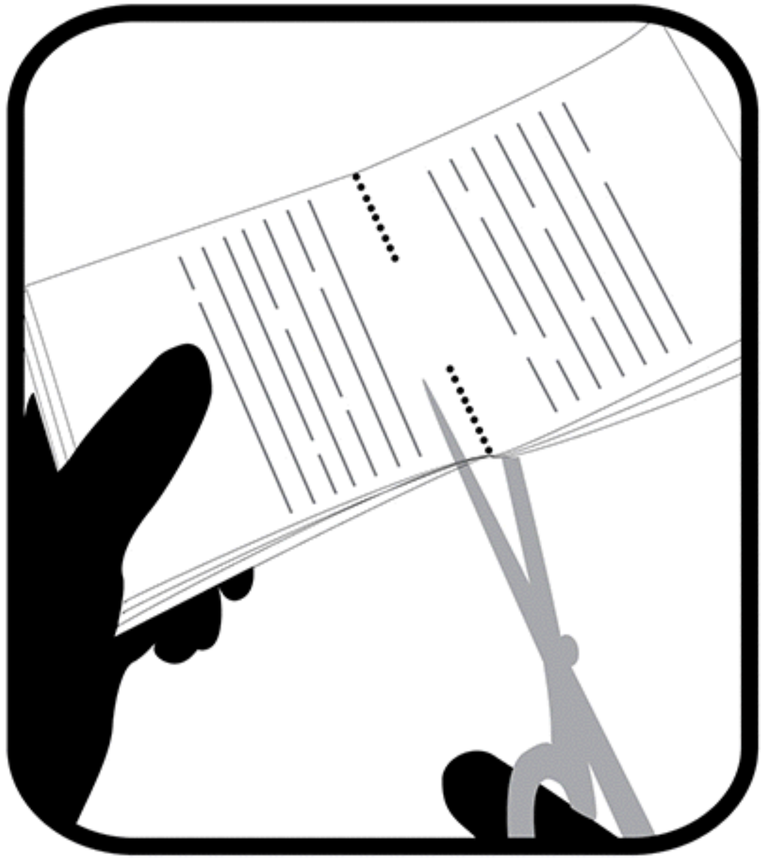
Dependency Injection

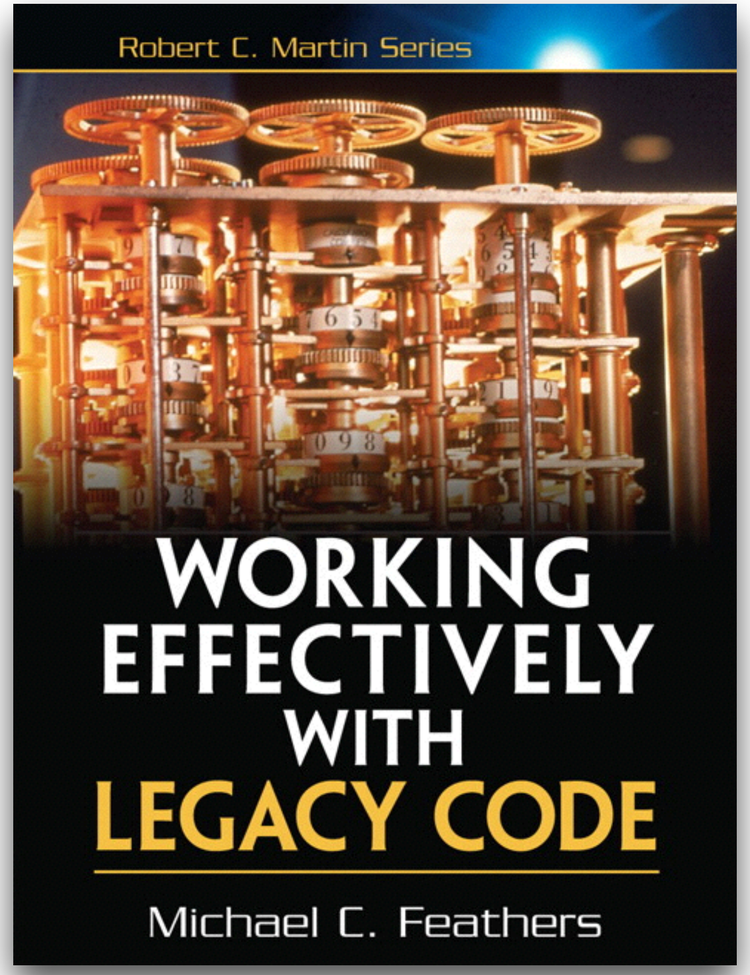
- Extract and Override
- Method Injection
- Property Injection
- Constructor Injection

http://www.flickr.com/photos/8499561@N02/2755481069

```objc
- (NSNumber *)nextID
{
    NSNumber *result = [[NSUserDefaults standardUserDefaults] objectForKey:@"reminderID"];
    if (!result) {
        result = @0;
    }
    return result;
}
```

Seams





http://www.flickr.com/photos/gileslane/2262575575

Extract and Override

```
- (NSNumber *)nextID
{
    NSNumber *result = [[NSUserDefaults standardUserDefaults] objectForKey:@"reminderID"];
    if (!result) {
        result = @0;
    }
    return result;
}
```

Extract and Override

```
- (NSNumber *)nextID
{
    NSNumber *result = [[self userDefaults] objectForKey:@"reminderID"];
    if (!result) {
        result = @0;
    }
    return result;
}

- (NSUserDefaults *)userDefaults
{
    return [NSUserDefaults standardUserDefaul
}
```

```
@interface TestingReminderID : ReminderID
@end

@implementation TestingReminderID

- (NSUserDefaults *)userDefaults
{
    return nil; // Whatever you want!
}

@end
```

Method Injection

```
- (NSNumber *)nextID
{
    NSNumber *result = [[NSUserDefaults standardUserDefaults] objectForKey:@"reminderID"];
    if (!result) {
        result = @0;
    }
    return result;
}
```

Method Injection

```
- (NSNumber *)nextIDFromUserDefaults:(NSUserDefaults *)userDefaults
{
    NSNumber *result = [userDefaults objectForKey:@"reminderID"];
    if (!result) {
        result = @0;
    }
    return result;
}
```

Property Injection

```
@implementation ReminderID

- (NSNumber *)nextID
{
    NSNumber *result = [self.userDefaults objectForKey:@"reminderID"];
    if (!result) {
        result = @0;
    }
    return result;
}

@end
```

Property Injection

```
@interface ReminderID : NSObject

@property (nonatomic, strong) NSUserDefaults *userDefaults;

- (NSNumber *)nextID;

@end


@implementation ReminderID

- (NSNumber *)nextID
{
    NSNumber *result = [self.userDefaults objectForKey:@"reminderID"];
    if (!result) {
        result = @0;
    }
    return result;
}

@end
```

Property Injection

```
@implementation ReminderID

- (NSUserDefaults *)userDefaults
{
    if (!_userDefaults) {
        _userDefaults = [NSUserDefaults standardUserDefaults];
    }
    return _userDefaults;
}

- (NSNumber *)nextID
{
    NSNumber *result = [self.userDefaults objectForKey:@"reminderID"];
    if (!result) {
        result = @0;
    }
    return result;
}

@end
```

Constructor Injection

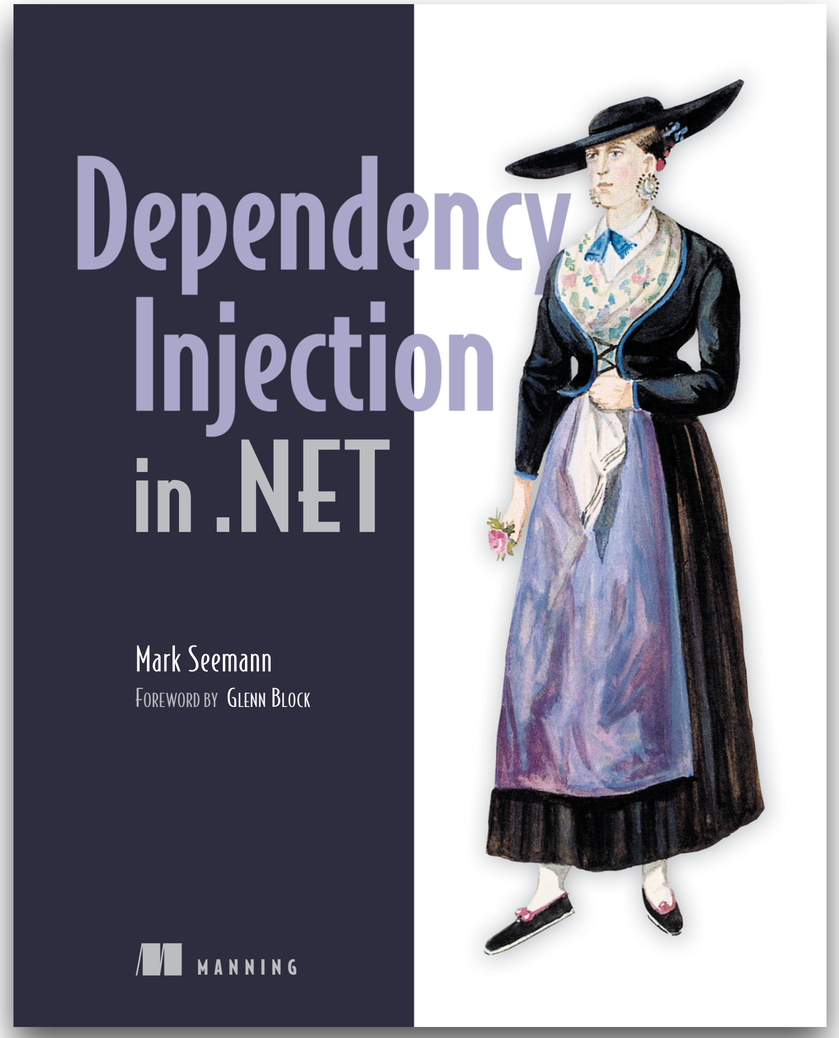
```
@implementation ReminderID
{
    NSUserDefaults *_userDefaults;
}

- (instancetype)initWithUserDefaults:(NSUserDefaults *)userDefaults
{
    self = [super init];
    if (self) {
        _userDefaults = userDefaults;
    }
    return self;
}

- (NSNumber *)nextID
{
    NSNumber *result = [_userDefaults objectForKey:@"reminderID"];
    if (!result) {
        result = @0;
    }
    return result;
}

@end
```

Dependency Injection

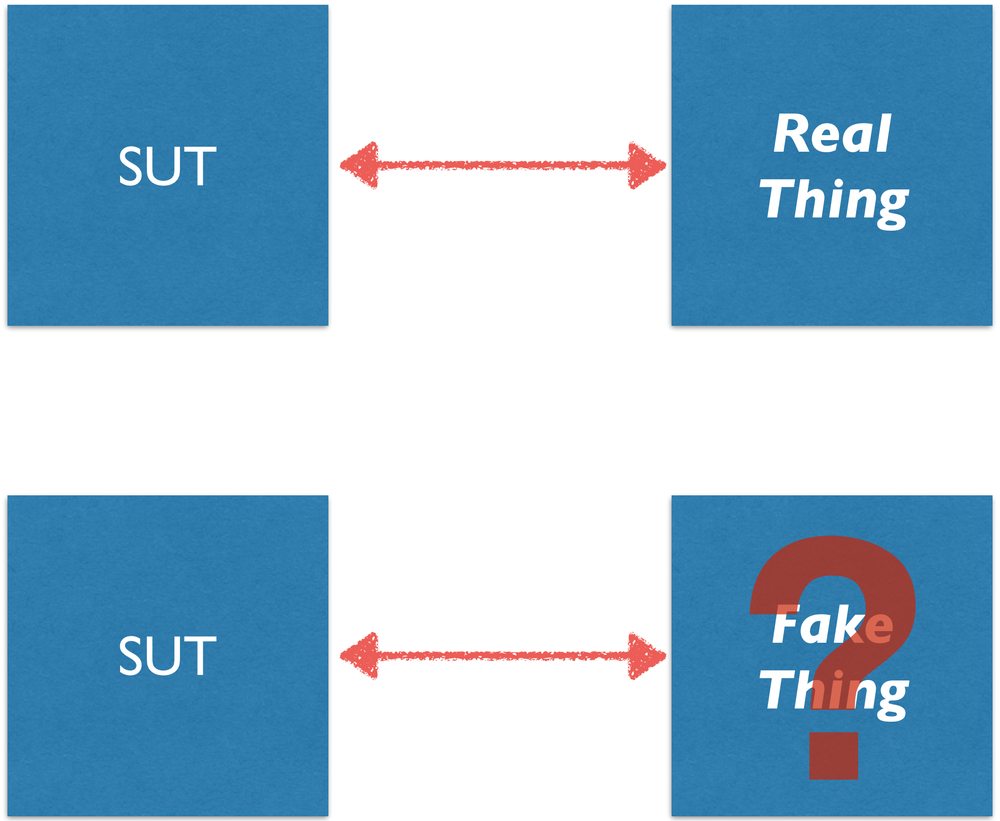
- Extract and Override
- Method Injection
- Property Injection
- Constructor Injection
- Ambient Context

http://www.flickr.com/photos/8499561@N02/2755481069

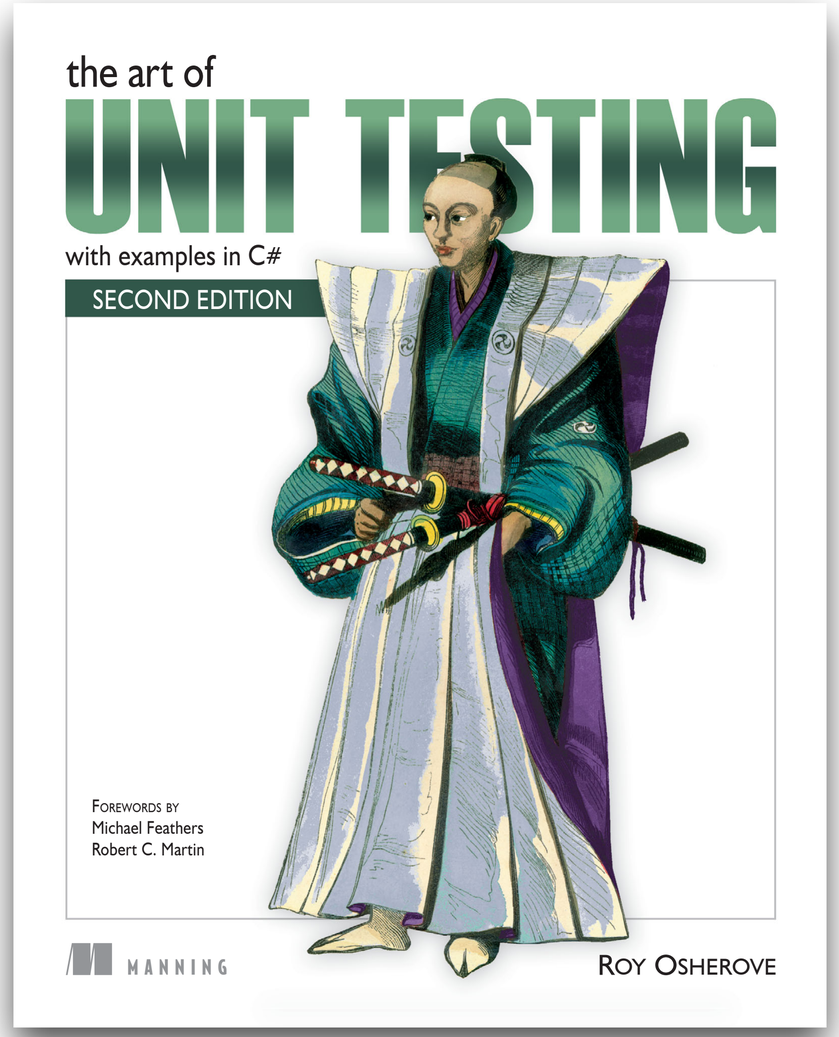
Dependency Injection



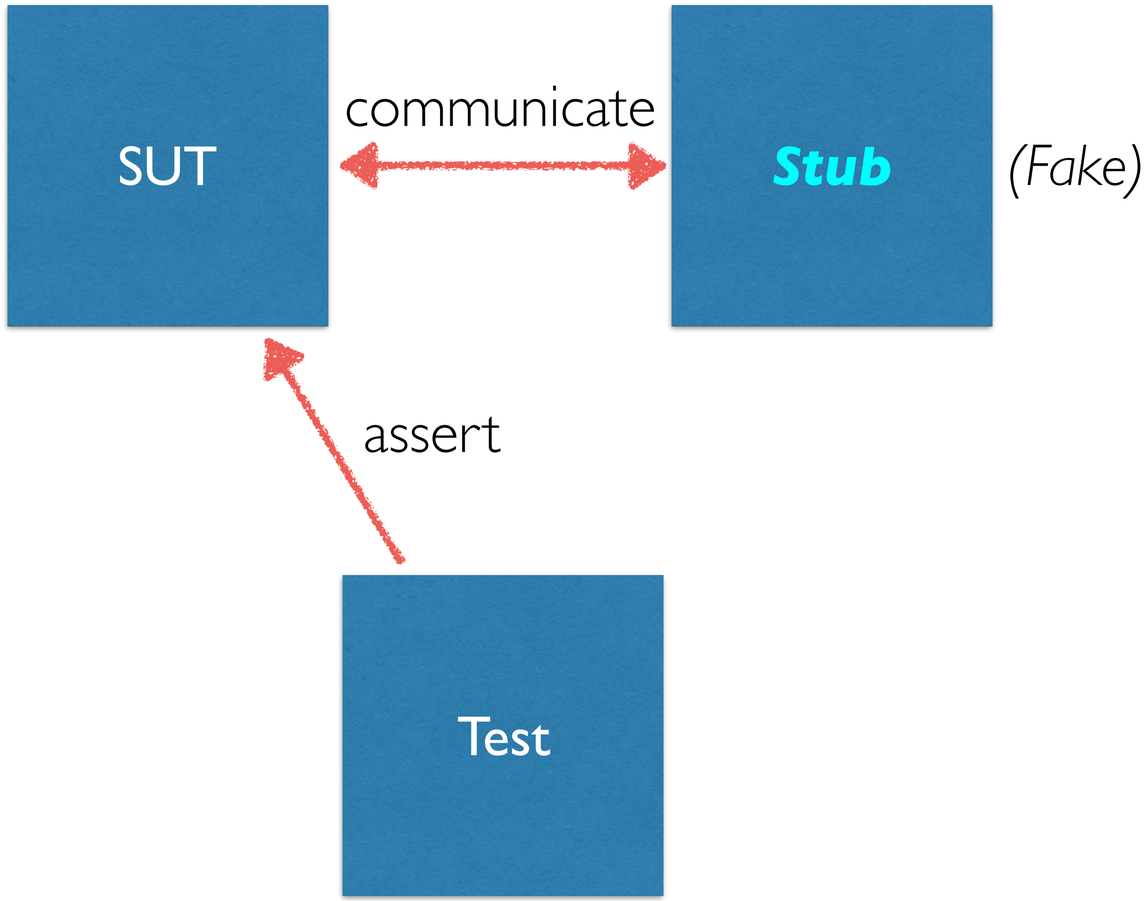
Interaction Test



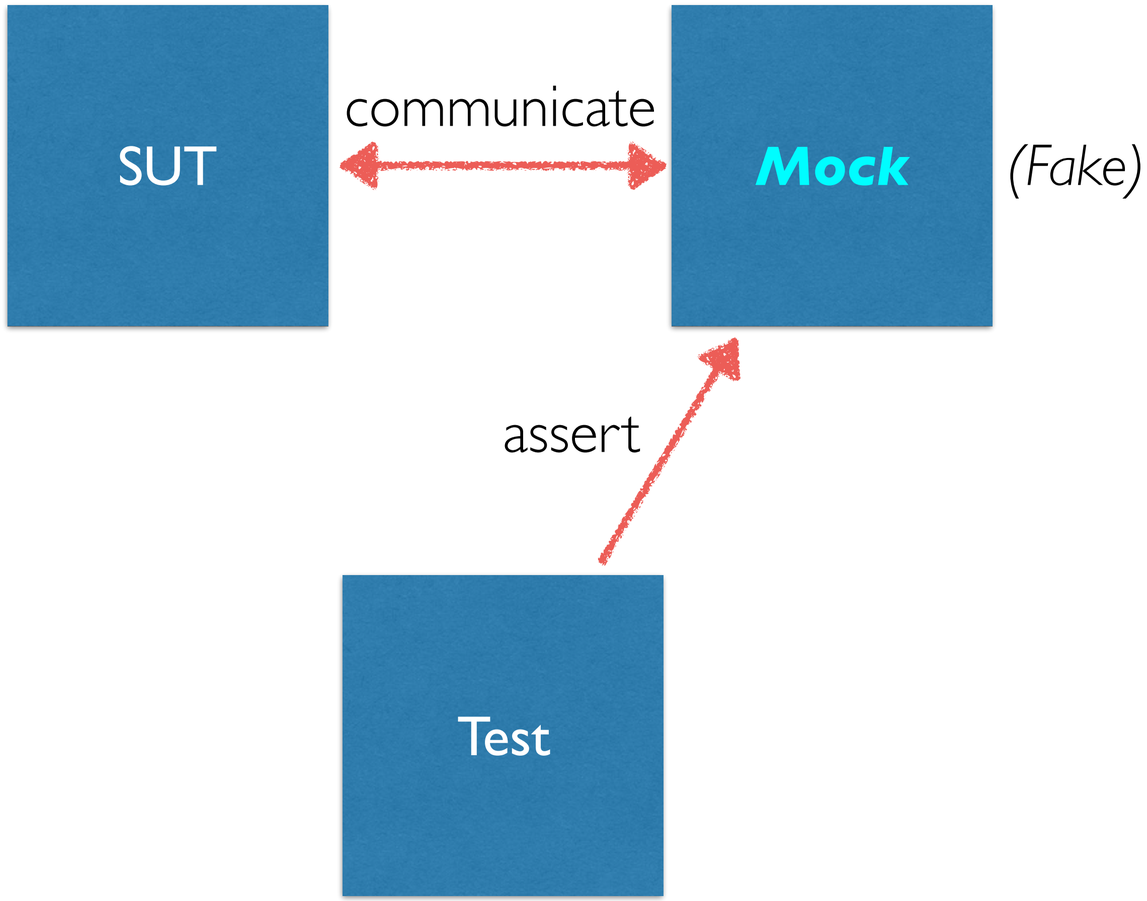
Types of Fakes



Stub



Mock



“That’s great, Jon.
…But how do I use this for networking?”

Networking Call

```
- (void)fetchResources
{
    AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
    [manager GET:@"http://example.com/resources.json"
      parameters:nil
         success:^(AFHTTPRequestOperation *operation, id responseObject) { /* handle success */ }
         failure:^(AFHTTPRequestOperation *operation, NSError *error) { /* handle failure */ }
    ];
}
```

Inject dependency

Networking Call

```
- (void)fetchResources
{
    AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
    [manager GET:@"http://example.com/resources.json"
      parameters:nil
         success:^(AFHTTPRequestOperation *operation, id responseObject) { /* handle success */ }
         failure:^(AFHTTPRequestOperation *operation, NSError *error) { /* handle failure */ }
    ];
}
```

Record number of calls

Networking Call

```
- (void)fetchResources
{
    AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
    [manager GET:@"http://example.com/resources.json"
      parameters:nil
         success:^(AFHTTPRequestOperation *operation, id responseObject) { /* handle success */ }
         failure:^(AFHTTPRequestOperation *operation, NSError *error) { /* handle failure */ }
    ];
}
```

Fake return value

Networking Call

```
- (void)fetchResources
{
    AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
    [manager GET:@"http://example.com/resources.json"
      parameters:nil
         success:^(AFHTTPRequestOperation *operation, id responseObject) { /* handle success */ }
         failure:^(AFHTTPRequestOperation *operation, NSError *error) { /* handle failure */ }
    ];
}
```

Capture arguments

Let's Make a Fake:
1. Stub the method

```
@interface MockAFNetworkingGET : NSObject
@end


@implementation MockAFNetworkingGET

- (AFHTTPRequestOperation *)GET:(NSString *)URLString
                     parameters:(NSDictionary *)parameters
                        success:(void (^)(AFHTTPRequestOperation *operation, id responseObject))success
                        failure:(void (^)(AFHTTPRequestOperation *operation, NSError *error))failure
{
    return nil;
}

@end
```

Let's Make a Fake:
2. Record number of calls

```
@interface MockAFNetworkingGET : NSObject
@property (nonatomic) int callCount;
@end


@implementation MockAFNetworkingGET

- (AFHTTPRequestOperation *)GET:(NSString *)URLString
                     parameters:(NSDictionary *)parameters
                        success:(void (^)(AFHTTPRequestOperation *operation, id responseObject))success
                        failure:(void (^)(AFHTTPRequestOperation *operation, NSError *error))failure
{
    self.callCount += 1;
    return nil;
}

@end
```

Let's Make a Fake:
3. Fake return value

```
@interface MockAFNetworkingGET : NSObject
@property (nonatomic) int callCount;
@property (nonatomic, strong) AFHTTPRequestOperation *fakeReturnValue;
@end


@implementation MockAFNetworkingGET

- (AFHTTPRequestOperation *)GET:(NSString *)URLString
                     parameters:(NSDictionary *)parameters
                        success:(void (^)(AFHTTPRequestOperation *operation, id responseObject))success
                        failure:(void (^)(AFHTTPRequestOperation *operation, NSError *error))failure
{
    self.callCount += 1;
    return self.fakeReturnValue;
}

@end
```

Let's Make a Fake:
4. Capture arguments

```
@interface MockAFNetworkingGET : NSObject
@property (nonatomic) int callCount;
@property (nonatomic, strong) AFHTTPRequestOperation *fakeReturnValue;
@property (nonatomic, copy) NSString *URLString;
@property (nonatomic, copy) NSDictionary *parameters;
@property (nonatomic, copy) void (^success)(AFHTTPRequestOperation *operation, id responseObject);
@property (nonatomic, copy) void (^failure)(AFHTTPRequestOperation *operation, NSError *error);
@end


@implementation MockAFNetworkingGET

- (AFHTTPRequestOperation *)GET:(NSString *)URLString
                     parameters:(NSDictionary *)parameters
                        success:(void (^)(AFHTTPRequestOperation *operation, id responseObject))success
                        failure:(void (^)(AFHTTPRequestOperation *operation, NSError *error))failure
{
    self.callCount += 1;
    self.URLString = URLString;
    self.parameters = parameters;
    self.success = success;
    self.failure = failure;
    return self.fakeReturnValue;
}

@end
```

Let's Write a Test (1):
Does it call GET exactly once?

```
@interface ResourceFetcherTests : XCTestCase
@end


@implementation ResourceFetcherTests

- (void)testFetchResources_ShouldCallGET
{
    id mockGET = [[MockAFNetworkingGET alloc] init];
    ResourceFetcher *sut = [[ResourceFetcher alloc] initWithGETManager:mockGET];

    [sut fetchResources];

    XCTAssertEqual([mockGET callCount], 1);
}

@end
```

1. Arrange
2. Act
3. Assert

Let's Write a Test (2):
Simulate a JSON Response!

1. Arrange
2. Act
3. Assert

```
- (void)testFetchResources_WithSuccess_ShouldSetName
{
    id mockGET = [[MockAFNetworkingGET alloc] init];
    ResourceFetcher *sut = [[ResourceFetcher alloc] initWithGETManager:mockGET];

    [sut fetchResources];
    mockGET.success(nil, [@"{\"name\":\"Jon Reid\"}" dataUsingEncoding:NSUTF8StringEncoding]);

    XCTAssertEqualObjects(sut.name, @"Jon Reid");
}
```

Refactor Test Code:
Spot Duplication

```
- (void)testFetchResources_ShouldCallGET
{
    id mockGET = [[MockAFNetworkingGET alloc] init];
    ResourceFetcher *sut = [[ResourceFetcher alloc] initWithGETManager:mockGET];

    [sut fetchResources];

    XCTAssertEqual([mockGET callCount], 1);
}

- (void)testFetchResources_WithSuccess_ShouldSetName
{
    id mockGET = [[MockAFNetworkingGET alloc] init];
    ResourceFetcher *sut = [[ResourceFetcher alloc] initWithGETManager:mockGET];

    [sut fetchResources];
    mockGET.success(nil, [@"{\"name\":\"Jon Reid\"}" dataUsingEncoding:NSUTF8StringEncoding]);

    XCTAssertEqualObjects(sut.name, @"Jon Reid");
}
```

Refactor Test Code:
Move Variables into Test Fixture

```
@interface ResourceFetcherTests : XCTestCase
{
    MockAFNetworkingGET *mockGET;
    ResourceFetcher *sut;
}

- (void)setUp
{
    [super setUp];
    mockGET = [[MockAFNetworkingGET alloc] init];
    sut = [[ResourceFetcher alloc] initWithGETManager:mockGET];
}

- (void)tearDown
{
    sut = nil;
    [super tearDown];
}
```

Refactor Test Code:
Use Test Fixture

```
- (void)testFetchResources_ShouldCallGET
{
    [sut fetchResources];

    XCTAssertEqual([mockGET callCount], 1);
}

- (void)testFetchResources_WithSuccess_ShouldSetName
{
    [sut fetchResources];
    mockGET.success(nil, [@"{\"name\":\"Jon Reid\"}" dataUsingEncoding:NSUTF8StringEncoding]);

    XCTAssertEqualObjects(sut.name, @"Jon Reid");
}
```

Refactor Test Code:
Extract Helper Method

```
- (void)fetchSucceededWithJSON:(NSString *)JSONString
{
    [sut fetchResources];
     mockGET.success(nil, [JSONString dataUsingEncoding:NSUTF8StringEncoding]);
}

- (void)testFetchResources_WithSuccess_ShouldSetName
{
     [self fetchSucceededWithJSON:@"{\"name\":\"Jon Reid\"}"];

     XCTAssertEqualObjects(sut.name, @"Jon Reid");
}
```

Refactor test code to *express* the scenario

Let's Make a Fake

- Stub the method
- Record number of calls
- Fake return value
- Capture arguments

Dependency Injection

- Extract and Override
- Method Injection
- Property Injection
- Constructor Injection

3 Types of Unit Tests:

Return Value Test

State Test

Interaction Test

http://www.flickr.com/photos/jsmoorman/2298671281

Resources

- *Working Effectively with Legacy Code* by Michael Feathers
- *Dependency Injection in .NET* by Mark Seemann
- *The Art of Unit Testing* by Roy Osherove

QualityCoding.org
@qcoding