Testing? We don’t need no stinking testing!

# DEVintersection

## Session AB06

## Async JavaScript Testing

Ward Bell

WardB@ideablade.com

@wardbell

DEV intersection

# Agenda

- Basic Jasmine testing
- The Zza sample
- Stopping runaway apps
- Simple mock data
- Data mother
- Async for real

# Tools

Jasmine

Angular

Karma

Web Storm

# Jasmine 2.0 ~~1.3~~

Jasmine

Powered by PIVOTAL LABS

Jump To: ajax.js boot.js custom_equality.js custom_matcher.js introduction.js python_egg.py ruby_gem.rb

## introduction.js

Jasmine is a behavior-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that you can easily write tests. This guide is running against Jasmine version 2.0.0.

### Standalone Distribution

The standalone distribution contains everything you need to start running Jasmine. After downloading a particular version and unzipping, opening `SpecRunner.html` will run the included specs. You'll note that both the source files and their respective specs are linked in the `<head>` of the `SpecRunner.html`. To start using Jasmine, replace the source/spec files with your own.

### Suites: `describe` Your Tests

A test suite begins with a call to the global Jasmine function `describe` with two parameters: a string and a function. The string is a name or title for a spec suite – usually what is being tested. The function is a block of code that implements the suite.

### Specs

Specs are defined by calling the global Jasmine function `it`, which, like `describe` takes a string and a function. The string is the title of the spec and the function is the spec, or test. A spec contains one or more expectations that test the state of the code.

```
describe("A suite", function() {
  it("contains spec with an expectation", function() {
    expect(true).toBe(true);
  });
});
```

# jasmine matchers

GitHub, Inc. [US] https://github.com/pivotal/jasmine/wiki/Matchers

## Matchers

Jasmine has several built-in matchers. Here are a few:

`expect(x).toEqual(y);` compares objects or primitives `x` and `y` and passes if they are equivalent

`expect(x).toBe(y);` compares objects or primitives `x` and `y` and passes if they are the same object

`expect(x).toMatch(pattern);` compares `x` to string or regular expression `pattern` and passes if they match

`expect(x).toBeDefined();` passes if `x` is not `undefined`

`expect(x).toBeUndefined();` passes if `x` is `undefined`

`expect(x).toBeNull();` passes if `x` is `null`

`expect(x).toBeTruthy();` passes if `x` evaluates to true

`expect(x).toBeFalsy();` passes if `x` evaluates to false

`expect(x).toContain(y);` passes if array or string `x` contains `y`

`expect(x).toBeLessThan(y);` passes if `x` is less than `y`

`expect(x).toBeGreaterThan(y);` passes if `x` is greater than `y`

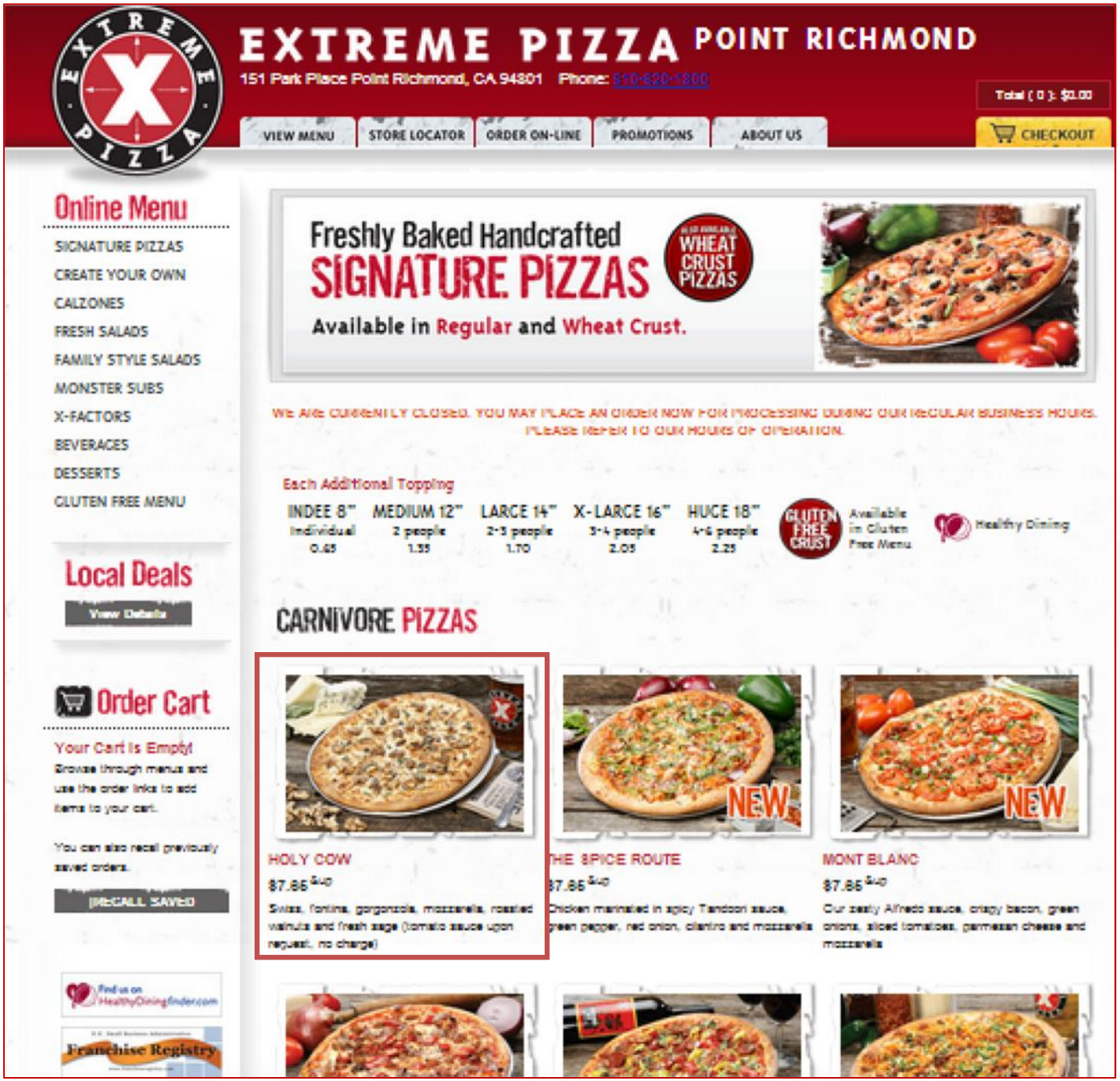
`expect(function(){fn();}).toThrow(e);` passes if function `fn` throws exception `e` when executed

**Can negate them with the ." .not" prefix**

# Let’s see some tests

intro.spec.js

I don’t test ’3 > 4’


EXTREME PIZZA POINT RICHMOND
151 Park Place Point Richmond, CA 94801 Phone: 510-620-1800
Total ( 0 ): $0.00
VIEW MENU
STORE LOCATOR
ORDER ON-LINE
PROMOTIONS
ABOUT US
CHECKOUT
Online Menu
SIGNATURE PIZZAS
CREATE YOUR OWN
CALZONES
FRESH SALADS
FAMILY STYLE SALADS
MONSTER SUBS
X-FACTORS
BEVERAGES
DESSERTS
GLUTEN FREE MENU
Local Deals
View Details
Order Cart
Your Cart is Empty!
Browse through menus and use the order links to add items to your cart.
You can also recall previously saved orders.
RECALL SAVED
Find us on HealthyDiningfinder.com
Franchise Registry
Freshly Baked Handcrafted
SIGNATURE PIZZAS
WHEAT CRUST PIZZAS
Available in Regular and Wheat Crust.
WE ARE CURRENTLY CLOSED. YOU MAY PLACE AN ORDER NOW FOR PROCESSING DURING OUR REGULAR BUSINESS HOURS. PLEASE REFER TO OUR HOURS OF OPERATION.
Each Additional Topping
INDEE 8" Individual 0.65
MEDIUM 12" 2 people 1.35
LARGE 14" 2-3 people 1.70
X-LARGE 16" 3-4 people 2.05
HUGE 18" 4-6 people 2.25
GLUTEN FREE CRUST
Available in Gluten Free Menu
Healthy Dining
CARNIVORE PIZZAS
HOLY COW
$7.65 &up
Swiss, fontina, gorgonzola, mozzarella, roasted walnuts and fresh sage (tomato sauce upon request, no charge)
NEW
THE SPICE ROUTE
$7.65 &up
Chicken marinated in spicy Tandoori sauce, green pepper, red onion, cilantro and mozzarella
NEW
MONT BLANC
$7.65 &up
Our zesty Alfredo sauce, crispy bacon, green onions, sliced tomatoes, parmesan cheese and mozzarella

Zza! Home **Order** Customer About Cart

Menu
- Pizza
- Salad
- Drinks

Order Summary

| Item | Size | Qty |
|---|---|---|

Cart

Recently Viewed

| Item | Size | Qty |
|---|---|---|

# Order Pizza

Handcrafted with fresh ingredients and baked in a woodburning oven. Our pizzas are so tasty and authentic you'll shout, "*Mamma Mia!*"

## Plain Cheese

Your basic plain cheese pizza with no toppings. Includes tomato sauce, mozzarella, oregano and a sprinkling of cheddar

## Make Your Own

Make your own Includes tomato sauce, mozzarella, oregano and a sprinkling of cheddar

## Holy Smokes

Swiss, fontina, gorgonzola, mozzarella, roasted walnuts and fresh sage

## Taj Mahal

Chicken marinated in spicy Tandoori sauce, green pepper, red onion, cilantro and mozzarella

# EXTREME PIZZA POINT RICHMOND

151 Park Place Point Richmond, CA 94801 Phone: 510-620-5800

Total ( 0 ): $0.00

VIEW MENU | STORE LOCATOR | ORDER ON-LINE | PROMOTIONS | ABOUT US | CHECKOUT

## Online Menu

- SIGNATURE PIZZAS
- CREATE YOUR OWN
- CALZONES
- FRESH SALADS
- FAMILY STYLE SALADS
- MONSTER SUBS
- X-FACTORS
- BEVERAGES
- DESSERTS
- GLUTEN FREE MENU

## Local Deals

View Details

## Order Cart

Your Cart Is Empty!

Browse through menus and use the order links to add items to your cart.

You can also recall previously saved orders.

RECALL SAVED

Find us on HealthyDiningFinder.com

Franchise Registry

## Holy Cow

Swiss, fontina, gorgonzola, mozzarella, roasted walnuts and fresh sage (tomato sauce upon request, no charge)

| Size | Price |
|---|---|
| Indee-8" | $7.65 |
| M-12" | $16.95 |
| L-14" | $20.95 |
| XL-16" | $23.45 |
| Huge-18" | $26.95 |

Quantity: 1

Who is this for: (optional)

Crust | Homemade Sauces | Cheeses | Fruits & Veggies | Meats | Fresh Spices | Remove

Meats

- Pepperoni
- Salami
- Canadian Bacon
- Thai Chicken*
- Beef Meatball
- Anchovies
- Italian Sausage
- Smoked Bacon
- BBQ Chicken*
- Grilled Chicken* [x1]
- Shredded BBQ Pork*
- Linguica

< Prev > Next

Instructions:
Add any additional instructions here. Do not use for ordering items or toppings.

This is for: Pickup Delivery

× Cancel + Add to Order

Zza! Home **Order** Customer About Cart

Menu
- Pizza
- Salad
- Drinks

Order Summary

| Item | Size | Qty |
|---|---|---|
| Holy Smokes | Large 14" | 1 |

Cart

Recently Viewed

| Item | Size | Qty |
|---|---|---|

# Holy Smokes

Swiss, fontina, gorgonzola, mozzarella, roasted walnuts and fresh sage

| Size | Price |
|---|---|
| Personal 8" | $7.65 |
| Medium 12" | $16.95 |
| Large 14" | $20.95 |
| X-Large 16" | $23.45 |
| Huge 18" | $26.95 |
| Gluten Free 12" | $21.45 |

Quantity 1

Instructions

Crust Sauce Cheese Veggie **Meat** Spice

*Choices with an (*) after the name count as 2.*

- [ ] Pepperoni
- [ ] Smoked Bacon
- [ ] Thai Chicken*
- [ ] Shredded BBQ Pork*
- [ ] Italian Sausage
- [ ] Canadian Bacon
- [x] Grilled Chicken*
- [ ] Anchovies
- [ ] Salami
- [ ] BBQ Chicken*
- [ ] Beef Meatball
- [ ] Linguica

# EXTREME PIZZA POINT RICHMOND

151 Park Place Point Richmond, CA 94801 Phone: 510-620-1800

VIEW MENU | STORE LOCATOR | ORDER ON-LINE | PROMOTIONS | ABOUT US

## Online Menu

- SIGNATURE PIZZAS
- CREATE YOUR OWN
- CALZONES
- FRESH SALADS
- FAMILY STYLE SALADS
- MONSTER SUBS
- X-FACTORS
- BEVERAGES
- DESSERTS
- GLUTEN FREE MENU

## Local Deals

View Details

Find Out How to Apply Coupon (Read Instructions)

## Review Your Pickup Order

The following items are in this pickup order. Please review all items and adjust the quantities if desired. **Average Picku time is 25 minutes.**

Switch to **DELIVERY**

| | Item | Size | Qty | Price |
|---|---|---|---|---|
| Delete | Holy Cow | L-14" | 1 | $20.95 |
| | Classic Crust | | | $0.00 |
| | No Pizza Sauce | | | $0.00 |
| | Pizza Sauce: Pesto Sauce* | | | $3.40 |
| | Topping: Grilled Chicken* | | | $3.40 |
| | | | Sum: | $27.75 |
| | | | Items Total: | $27.75 |
| | | | Subtotal: | $27.75 |
| | | | Sales Tax (8.75%): | $2.43 |
| | | | **Order Total:** | **$30.18** |

PICKUP

Zza! Home **Order** Customer About Cart

Menu
- Pizza
- Salad
- Drinks

Order Summary

| Item | Size | Qty |
|---|---|---|
| Holy Smokes | Large 14" | 1 |

Cart

Recently Viewed

| Item | Size | Qty |
|---|---|---|

# Your Order

| | Item | Size | Qty | Price |
|---|---|---|---|---|
| Remove | Holy Smokes | Large 14" | 1 | $20.95 |
| | Classic Crust | | | $0.00 |
| | Grilled Chicken* | | | $3.40 |
| | Pesto* | | | $3.40 |
| | | | Item Total | $27.75 |
| | | | Order Total | $27.75 |

*Options with an (*) count as 2.*

# Testing and Boundaries

Browser ---------- Database

Browser | Bootstrap | Routing | View (html) |  ViewModel | DataContext | AJAX
--
web service | ORM | Database

Crossing boundaries = hi-fi + hard
Minimize these crossings

# Test Types

Unit

Integration

End-to-end

# Unit Test

A test with no real dependencies

- every dependency is faked
- easy to write
- usually too trivial

# Integration Test

At least one dependency is real

even if dependency uses fakes

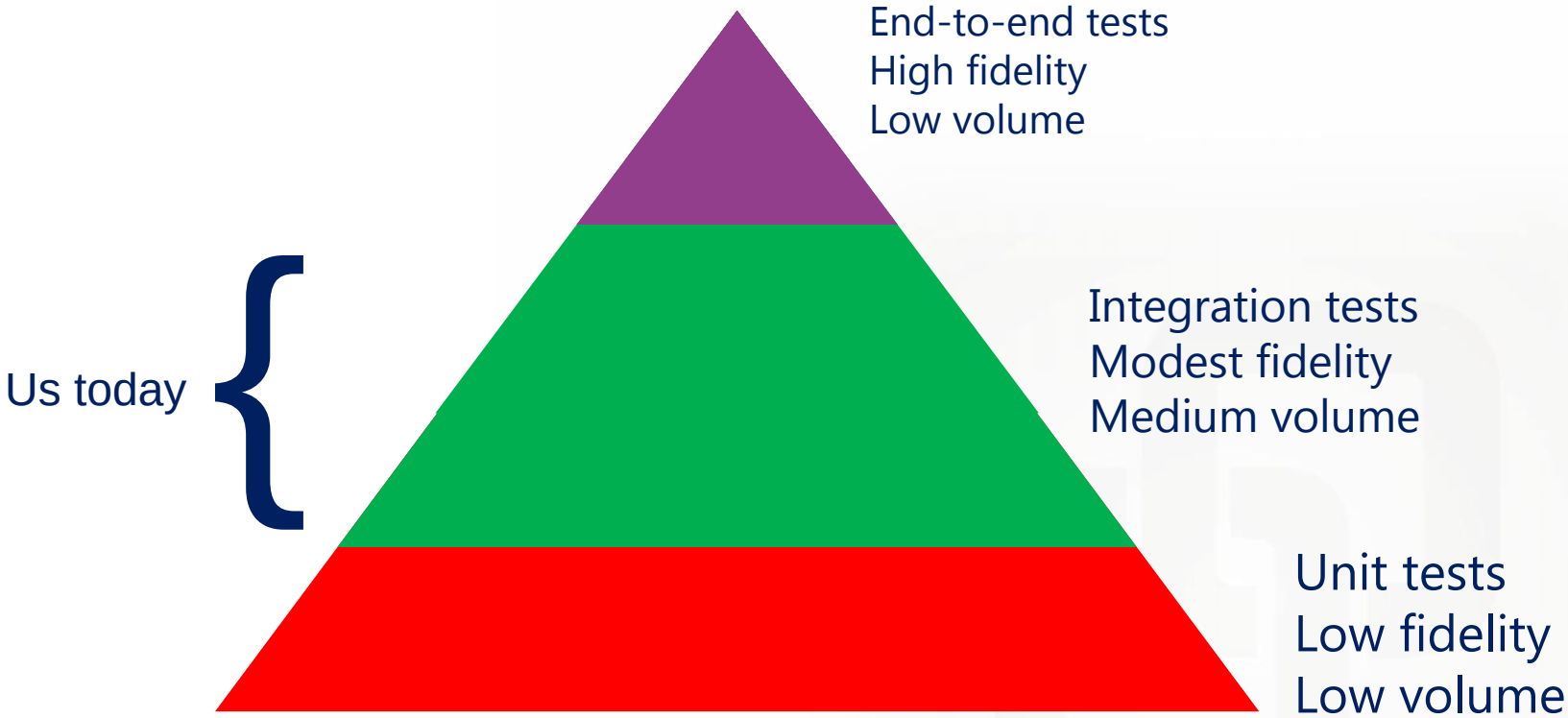
most of the tests I write

# End-to-end tests (E2E)

Test the running app

mimic user activity

# My Testing Spectrum



# The “runaway app”

## Original

```
 // dataservice autoloads data from server
app.run( ['util', 'dataservice', function run ( util ) {
        util.logger.info( "Zza SPA is loaded and running on " +
        util.config.server );
 }]);
```

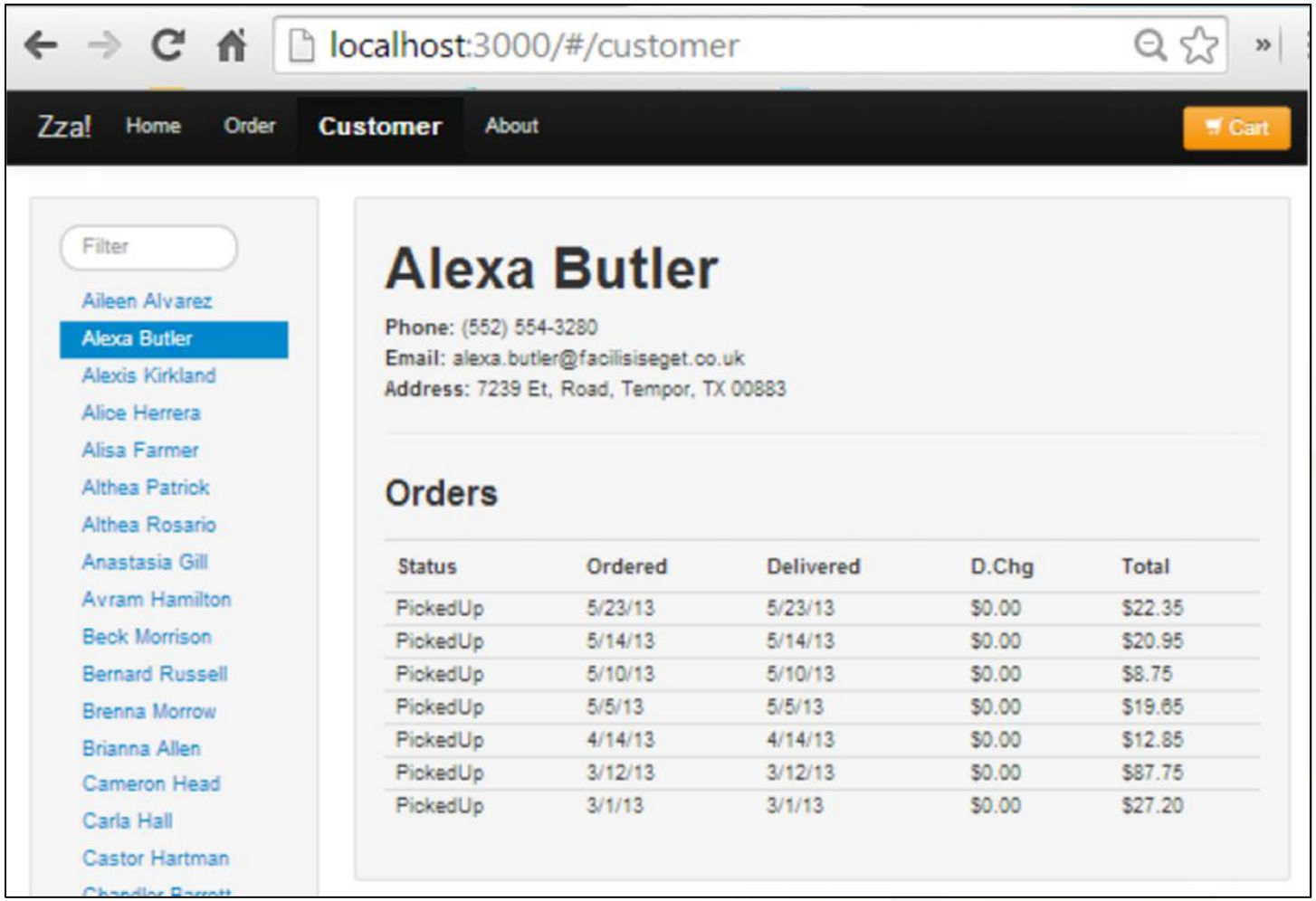
## Now

```
// move run block to'appStart'service
app.run( ['appStart', function ( appStart ) {
        appStart.start();
 }]);
```

# Mock the app start

appstart.spec.js

# Controller needs data




Should show “customers” and “orders”

Don’t care about the data themselves

# Customer Controller

```
// Customer controller state that survives
// the controller's routine creation and destruction.
angular.module( "app" ).value( 'customer.state', {});

// Customer controller
angular.module( "app" ).controller( 'customer',
    ['customer.state', 'dataservice', controller] );

function controller(customerState, dataservice) {
    var vm  = this;
    vm.customerFilterText = customerState.customerFilterText || '';
    vm.customers = [];
    vm.filteredCustomers = filteredCustomers;
    vm.isLoadingCustomers = false;
    vm.isLoadingOrders = false;
    vm.isSelected =isSelected;
    vm.orderHeaders = orderHeaders;
    vm.select = select;
    vm.selectedCustomer = null;
```

# Mock the dataservice for the customer controller

customer-controller.spec.js

# When data-faking fails

Logic cares about object attributes

Lots of object attributes

Interrelated objects (object graphs)

takes too long to fake ‘em

Zza! Home **Order** Customer About Cart

Menu
- Pizza
- Salad
- Drinks

Order Summary

| Item | Size | Qty |
|---|---|---|
| Holy Smokes | Large 14" | 1 |

Cart

Recently Viewed

| Item | Size | Qty |
|---|---|---|

# Holy Smokes

Swiss, fontina, gorgonzola, mozzarella, roasted walnuts and fresh sage

| Size | Price |
|---|---|
| Personal 8" | $7.65 |
| Medium 12" | $16.95 |
| Large 14" | $20.95 |
| X-Large 16" | $23.45 |
| Huge 18" | $26.95 |
| Gluten Free 12" | $21.45 |

Quantity 1

Instructions

Crust Sauce Cheese Veggie **Meat** Spice

*Choices with an (*) after the name count as 2.*

- [ ] Pepperoni
- [ ] Smoked Bacon
- [ ] Thai Chicken*
- [ ] Shredded BBQ Pork*
- [ ] Italian Sausage
- [ ] Canadian Bacon
- [x] Grilled Chicken*
- [ ] Anchovies
- [ ] Salami
- [ ] BBQ Chicken*
- [ ] Beef Meatball
- [ ] Linguica

| Key | Value |
|---|---|
| [0] | |
| "_id" | 51f06ded06a7baa417000065 |
| "customerId" | 51f06ded06a7baa417000001 |
| "name" | "Derek Puckett" |
| "statusId" | 2 |
| "status" | "PickedUp" |
| "ordered" | Tue Mar 12 10:30:00 PDT 2013 |
| "phone" | "(954) 594-9355" |
| "delivered" | Tue Mar 12 11:07:00 PDT 2013 |
| "deliveryCharge" | 0 |
| "itemsTotal" | 31.95 |
| "items" | |
| [0] | |
| "productId" | 3 |
| "name" | "Holy Smokes" |
| "type" | "pizza" |
| "sizeId" | 3 |
| "size" | "Large 14"" |
| "qty" | 1 |
| "unitPrice" | 20.95 |
| "totalPrice" | 26.05 |
| "options" | |
| [0] | |
| "optionId" | 41 |
| "name" | "Pepperoni" |
| "qty" | 1 |
| "price" | 1.7 |
| [1] | { "optionId" : 13 , "name" : "Feta" , "qty" : 1 , "price" : 1.7} |
| [2] | { "optionId" : 42 , "name" : "Italian Sausage" , "qty" : 1 , "price" : 1.7} |
| [3] | { "optionId" : 54 , "name" : "Cilantro" , "qty" : 1 , "price" : 0} |
| [1] | { "productId" : 22 , "name" : "Caesar Salad" , "type" : "salad" , "sizel...: [ { " |
| [2] | { "productId" : 31 , "name" : "Cola" , "type" : "drink" , "sizeId" : 11 , "size" |

# Zza Schema

## Volatile

- Customer
- Order
- OrderItem
- OrderItemOption

## Constant

- OrderStatus
- Product
- ProductOption
- ProductSize

# Separate two kinds of "data service"

## volatile data loaded *ad hoc*

```
angular.module( "app" ).factory( 'dataservice',
    ['breeze', 'entityManagerFactory', 'lookups', 'model', 'util', factory]);

function factory( breeze, entityManagerFactory, lookups, model, util ) {
    var logger   = util.logger,
        isReady  = null,// becomes a promise, not a boolean
        manager  = entityManagerFactory.getManager(),
        $timeout = util.$timeout;
    var service = {
        cartOrder: null,
        draftOrder: null,
        getCustomers: getCustomers,
        getOrderHeadersForCustomer: getOrderHeadersForCustomer,
        lookups: lookups,
        ready: ready,
        saveChanges: saveChanges
    };
    return service;
```

# Separate the two kinds of "data service"

## lookups loaded at start

```
angular.module( "app" ).factory( 'lookups',
    ['breeze', 'entityManagerFactory', 'util', factory]);

function factory( breeze, emFactory, util ) {
   var isReady  = null,// becomes a promise
   var service = {
       ready: ready
       // extended during initialization
   };
   return service;

   ... extended with lookups data fetched from server ...

   service.OrderStatus     = ...
   service.products        = manager.getEntities('Product');
   service.productOptions  = manager.getEntities('ProductOption');
   service.productSizes    = manager.getEntities('ProductSize');
```

OrderStatus

Product

ProductOption

ProductSize

# Data mother

# Fake http traffic with $httpBackend

don’t go async until you have to

sample the traffic

replay it with $httpBackend

lookupsHttpResponse.js

# Data mother for lookups spec

lookups.spec.js

# Priming w/ in-mem data … in BreezeJS

don’t go async until you have to

capture data from live app

subset it

emFactoryMock

lookupsExport.js
emFactory-mock.js
emFactory-mock.spec.js

# Data mother
# for optionItems ViewModel spec

orderItemOptionVm.js
orderItemOptionVm.spec.js

# Async for real

# Asynchronous tests

*Pause the testrunner until test finishes*

Sources of Asynchrony

- DOM manipulation
- setTimeout/setInterval
- AJAX requests (XHR)

1st rule: fake it when you can

# Angular Midway Tester

intro.angular-dodging.async.spec.js

vs

intro.ngMidwayTesting.spec.js

# Exploring the web api w/ tests

*with Breeze*

can the client access the server?

no need to wander through the UI

api.breeze.async.spec.js

# Take Aways

- Testing is easy; testability is hard(er)
- Minimize test setup
- Write tests you can read
- Write at least one test for each app js file
- Test async code synchronously (when you can)
- Dependency Injection is your friend

# Resources

Zza sample code:
https://github.com/Breeze/breeze.js.samples/tree/master/node/zza-node-mongo

jasmine: http://pivotal.github.io/jasmine/

angular: http://angularjs.org/

breeze: http://breezejs.com

PluralSight

Misko Hevery: "Code Testability"

Joe Eames: "Testing clientside JavaScript"

Elijah Manor: "Front-End First: Testing and Prototyping JavaScript Apps"

# Questions?

**Don’t forget to enter your evaluation of**

**AB06 Async JavaScript Testing**

**using EventBoard!**

## Thank you!

# Promises make chained async easier

## *"I promise to call you back … as soon as I hear"*

"*start spinner, get metadata, then get lookups, then get sessions, then stop spinner*"

```
// The async pyramid of doom
someMethod(callback1(callback2(callback3(callback4))))
```

What if one of the callbacks fails?

```
// async promise goodness
someMethod.then(callback1)
          .then(callback2)
          .then(callback3)
          .then(callback4, failHandler)
```

```
$q   // angular promises
```