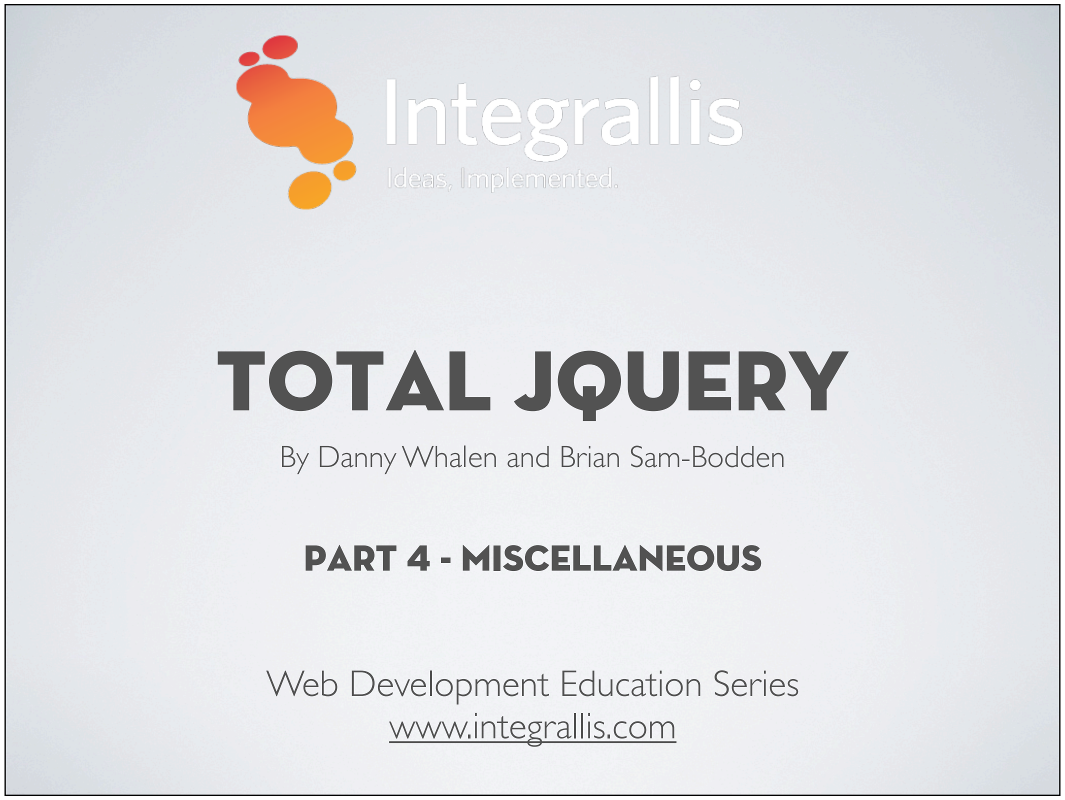


Integrallis
Ideas, Implemented.

TOTAL JQUERY

By Danny Whalen and Brian Sam-Bodden

PART 4 - MISCELLANEOUS

Web Development Education Series
www.integrallis.com

A publication of Integrallis Software, LLC.
www.integrallis.com/en/training
info@integrallis.com

OVERVIEW & OBJECTIVES

- Part 4 will cover:
  - Simplifying JavaScript with CoffeeScript
  - Backbone.js / Spine.js
  - jQuery Utilities
  - Themes and Theming
  - Avoiding conflicts with other libraries or versions of jQuery
  - Jasmine
  - Books and Tools

CoffeeScript

COFFEE SCRIPT

IT'S JUST JAVASCRIPT

COFFEE SCRIPT

IT'S JUST JAVASCRIPT

- As we have learned, JavaScript is powerful little language
- Yet, it has some areas that leave much too be desired
- One such are is its syntax - curly braces and semicolons

> ***"Underneath all of those embarrassing braces and semicolons, JavaScript has always had a gorgeous object model at its heart"***
>
> ***Jeremy Ashkenas***
> ***Creator of CoffeeScript***

COFFEE SCRIPT

IT'S JUST JAVASCRIPT

- CoffeeScript a veneer on top of JavaScript that makes it more palatable
- It makes JavaScript feel more like Ruby or Python by simplifying and clean up the language
- Programming concepts compile one-to-one to JavaScript so there is no runtime penalty
- The resulting JavaScript is as fast or faster than the hand-crafted JavaScript
- It works with any JS framework, including jQuery!

COFFEE SCRIPT

IT'S JUST JAVASCRIPT

- Some of examples of how CoffeeScript simplifies JavaScript by:
  - **Simplify Assignments:** Makes var implicit and therefore unnecessary, e.g. all declarations by default compile to a var declaration so you can just do `foo = 'bar'`
  - **No More Curlys or Semis:** Curly Braces and Semicolons are no longer required, ever!
  - **Function Syntax:** Does away with the need for the function keyword. A declaration like:

    ```
    square = function(x) { return x*x; };
    ```

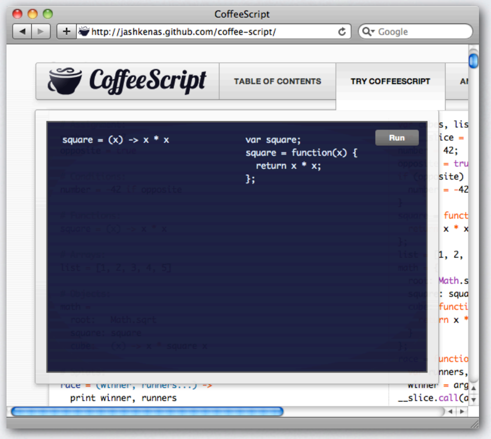
    becomes:

    ```
    square = (x) -> x*x
    ```

COFFEE SCRIPT

IT'S JUST JAVASCRIPT

- You can try CoffeeScript right on your browser. Go to http://jashkenas.github.com/coffee-script/ and click on the "Try CoffeeScript" tab:



```
square = (x) -> x * x
```

```
var square;
square = function(x) {
  return x * x;
};
```

TextMate Bundle for CoffeeScript => https://github.com/jashkenas/coffee-script-tmbundle

COFFEE SCRIPT

IT'S JUST JAVASCRIPT

- CoffeeScript encapsulates some common patterns:

```
alert "is that you bubba?" if bubba?
```

part_4/examples/coffeescript/bubba.coffee

- becomes...

```
if (typeof bubba !== "undefined" && bubba !== null) {
  alert("is that you bubba?");
}
```

COFFEE SCRIPT

IT'S JUST JAVASCRIPT

```
var today = new Date();
var day = today.getDay();
switch(day) {
    case 1:
        print("Looks like a case of the Mondays!");
        break;
    case 2: case 3: case 4:
        print("The middle of the week is for the birds!");
        break;
    case 5:
        print("Just pretend you're working until 4pm ;-)");
        break;
    default:
        print("The flipping weekend is here!");
}
```

part_1/examples/switch.js

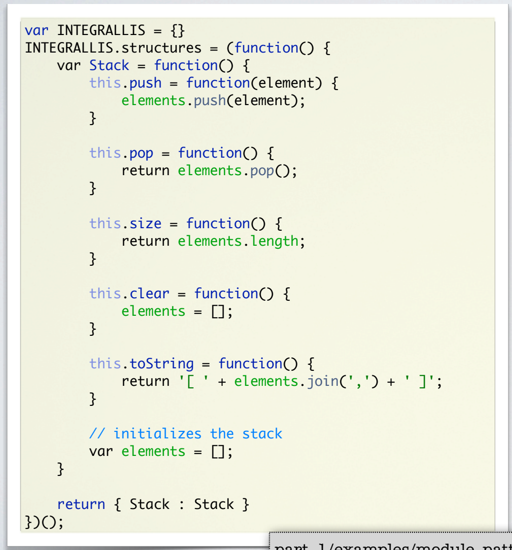
- Let's rewrite the simple implementation of the Switch example from part 1 of the course

```
today = new Date()
day = today.getDay()
switch day
  when 1 then print "Looks like a case of the Mondays!"
  when 2,3,4 then print "The middle of the week is for the birds!"
  when 5 then print "Just pretend you're working until 4pm ;-)"
  else print "The flipping weekend is here!"
```

part_4/examples/coffeescript/switch.coffee

COFFEE SCRIPT

IT'S JUST JAVASCRIPT



```
var INTEGRALLIS = {}
INTEGRALLIS.structures = (function() {
    var Stack = function() {
        this.push = function(element) {
            elements.push(element);
        }

        this.pop = function() {
            return elements.pop();
        }

        this.size = function() {
            return elements.length;
        }

        this.clear = function() {
            elements = [];
        }

        this.toString = function() {
            return '[ ' + elements.join(',') + ' ]';
        }

        // initializes the stack
        var elements = [];
    }

    return { Stack : Stack }
})();
```

part_1/examples/module_pattern/integrallis.structures.js

- Let's rewrite the simple implementation of the module pattern from part 1 using CoffeeScript

COFFEE SCRIPT

IT'S JUST JAVASCRIPT

- Brief and to the point, yet much more readable:

```
INTEGRALLIS = {}

do INTEGRALLIS.structures = -> (
  Stack = (options) -> (
    @push = (element) -> elements.push element
    @pop = () -> elements.pop()
    @size = () -> elements.length
    @clear = () -> elements = []
    @toString = () -> '[  #{elements.join ','}  ]'
    # initializes the stack
    elements = []
  )
  return { Stack : Stack }
)
```

part_4/examples/coffeescript/integrallis.structures.coffee

LAB 4.0

COFFESCRIPT

- In Lab **4.0** using the "Try CoffeeScript" tab:
  - Rewrite the **factorial.js** file from Part 1 of the course
  - Rewrite Lab 1.2 from Part 1 of the course

Backbone.js

BACKBONE.JS

MVC FOR JAVASCRIPT

BACKBONE.JS

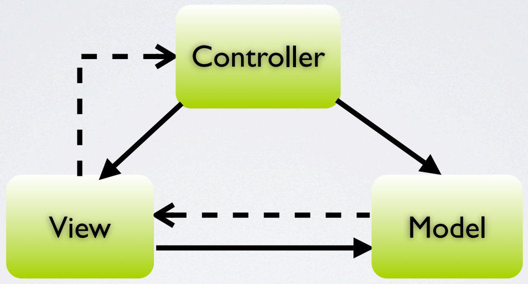
MVC FOR JAVASCRIPT

- Backbone.js is a small library provides a micro-MVC framework for your JavaScript applications
- It allow you to create a separation between the model (business JavaScript code) and the view (which renders the DOM)
- When models change their associated views are re-rendered without tightly coupling them to the DOM
- Backbone.js keeps state/model in a single space, model changes propagate automatically to the views with very little glue code

BACKBONE.JS

MVC FOR JAVASCRIPT

- Backbone.js allows you to separate your application into Models, Views and Controllers:



- Backbone.js core consists of 5 Prototypes; Model, View, Controller, Collection and Event. Model and Collection together make up the M part of the MVC pattern

BACKBONE.JS

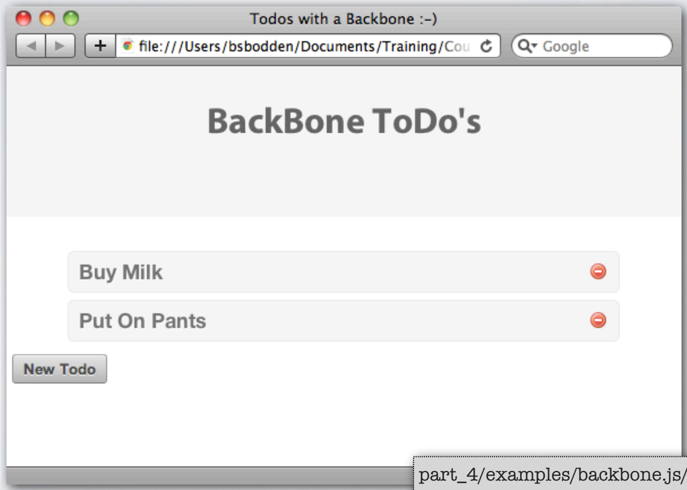
MVC FOR JAVASCRIPT

- Backbone.js helps you build complex single page applications that if built solely with something like jQuery can quickly become a big mess
- Avoids tight coupling with the current structure of the page
- Avoid having to create an application subsystem of callbacks to be able to maintain synchronization between your Models and the DOM
- Backbone.js also provides mechanism for RESTfully saving and updating models and communicating with a server using AJAX and JSON

BACKBONE.JS

MVC FOR JAVASCRIPT

- Let's do a walkthrough of a sample Backbone.js application, the BackBone ToDo's:

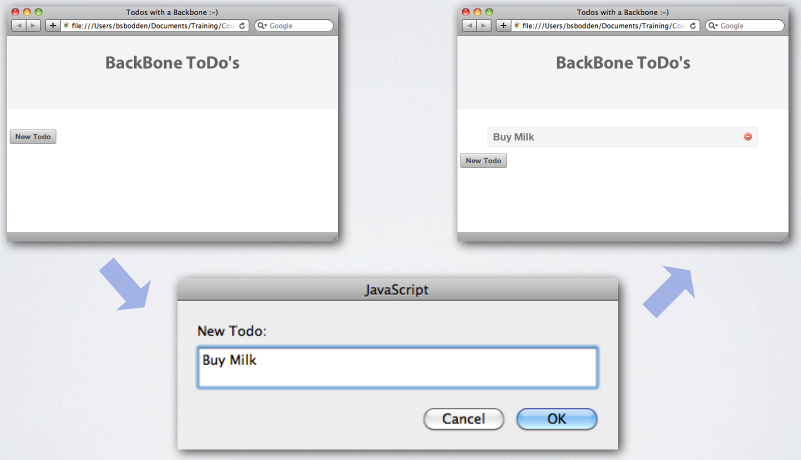

Todos with a Backbone :-)
file:///Users/bsbodden/Documents/Training/Cou
Google
BackBone ToDo's
Buy Milk
Put On Pants
New Todo

part_4/examples/backbone.js/todos

BACKBONE.JS

MVC FOR JAVASCRIPT

- The BackBone ToDo's allows you add a ToDo and check it off



part_4/examples/backbone.js/todos

BACKBONE.JS

MVC FOR JAVASCRIPT

- With Backbone we start by creating a model: `Todo`
- To manipulate a list of models we need a collection to contain said models: `Todos`

```
var Todo = Backbone.Model.extend({
   defaults: {
      name: null
   }
});

var Todos = Backbone.Collection.extend({
   model: Todo
});
```

part_4/examples/backbone.js/todos/todos.js

BACKBONE.JS

MVC FOR JAVASCRIPT

```
var TodoView = Backbone.View.extend({
     tagName: 'li',
     className: 'todo',
     //
     events: {
          'click a.delete': 'remove'
     },
     //
     initialize: function(){
          _.bindAll(this, 'render', 'unrender', 'remove');

          this.model.bind('change', this.render);
          this.model.bind('remove', this.unrender);
     },
     //
     render: function(){
          $(this.el).html('<div class="text">' +
                         this.model.get('name') +
                        '</div><div class="actions">' +
                        '<a href="#" class="delete">Done</a></div>');
          return this;
     },
     //
     unrender: function(){
          $(this.el).remove();
     },
     //
     remove: function(){
          this.model.destroy();
     }
});
```

part_4/examples/backbone.js/todos/todos.js

- Each Backbone model has an associated view
- For the `Todo` model there is the associated `TodoView`
- A view defines the element that wraps the view, CSS class name, events it responds to and method to render, unrender and remove itself

BACKBONE.JS

MVC FOR JAVASCRIPT

```
var TodosView = Backbone.View.extend({
      el: $('body'),
      //
      events: {
            'click button#add': 'showPrompt'
      },
      //
      initialize: function(){
            _.bindAll(this, 'render', 'appendTodo');

            this.todos = new Todos();
            this.todos.bind('add', this.appendTodo);
            this.render();
      },
      //
      render: function(){
            $(this.el).append("<ul class='todoList'></ul>");
            $(this.el).append("<button id='add'>New Todo</button>");
            _(this.todos.models).each(function(todo) {
              appendTodo(todo);
            }, this);
      },
      //
      showPrompt: function () {
            var name = prompt("New Todo:");
            var todo = new Todo({ name: name });
            this.todos.add(todo);
      },
      //
      appendTodo: function(todo){
            var view = new TodoView({
                  model: todo
            });
            $('ul', this.el).append(view.render().el);
      }
});
```

part_4/examples/backbone.js/todos/todos.js

- For the **Todos** model there is the associated **TodosView**
- The **TodosView** adds a button and binds its onclick event to a method that shows a prompt to create a new **Todo** model

BACKBONE.JS

MVC FOR JAVASCRIPT

- To kick start everything into gear we instantiate our top level view, the **`TodosView`** which will be attached to the body element of the page as defined by the **`el`** property of the view

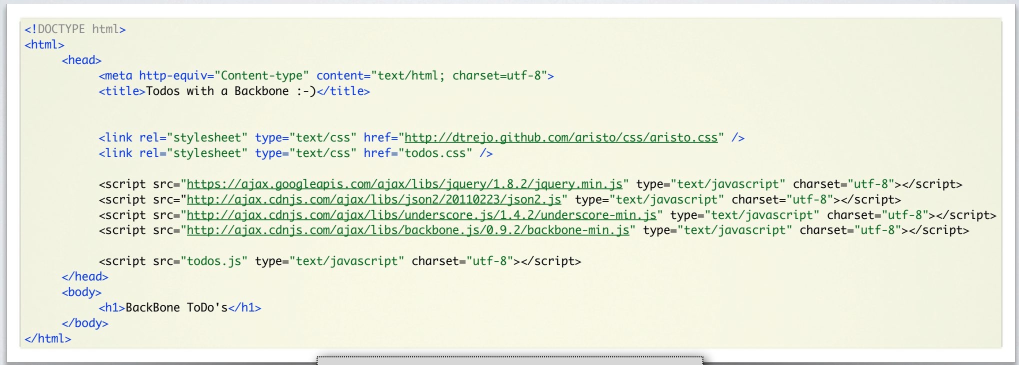
```
var todosView = new TodosView();
```

part_4/examples/backbone.js/todos/todos.js

BACKBONE.JS

MVC FOR JAVASCRIPT

- On the HTML page we load jQuery, Underscore (a dependency of BackBone.js) and BackBone.js



```html
<!DOCTYPE html>
<html>
      <head>
            <meta http-equiv="Content-type" content="text/html; charset=utf-8">
            <title>Todos with a Backbone :-)</title>


            <link rel="stylesheet" type="text/css" href="http://dtrejo.github.com/aristo/css/aristo.css" />
            <link rel="stylesheet" type="text/css" href="todos.css" />

            <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js" type="text/javascript" charset="utf-8"></script>
            <script src="http://ajax.cdnjs.com/ajax/libs/json2/20110223/json2.js" type="text/javascript" charset="utf-8"></script>
            <script src="http://ajax.cdnjs.com/ajax/libs/underscore.js/1.4.2/underscore-min.js" type="text/javascript" charset="utf-8"></script>
            <script src="http://ajax.cdnjs.com/ajax/libs/backbone.js/0.9.2/backbone-min.js" type="text/javascript" charset="utf-8"></script>

            <script src="todos.js" type="text/javascript" charset="utf-8"></script>
      </head>
      <body>
            <h1>BackBone ToDo's</h1>
      </body>
</html>
```

part_4/examples/backbone.js/todos/todos.html

BACKBONE.JS

MVC FOR JAVASCRIPT

- Other Resources:
  - Backbone.js main site: http://documentcloud.github.com/backbone/
  - Good Tutorial: http://arturadib.com/hello-backbonejs/
  - Sample Applications:
    - https://github.com/elfsternberg/The-Backbone-Store
    - http://addyosmani.com/resources/backbonegallery/

LAB 4.1 (OPTIONAL)

BACKBONE MVC

- In Lab `4.1` use the Backbone.js framework to create a JavaScript calculator:
  - Should support the basic operations +, -, *, /, =
  - The display should behave like a traditional calculator
  - Bonus: Add memory functions: **M+**, **M-**, **MC** and **MR**
  - Work in teams of 3-4 students

SPINE.JS

MVC FOR JAVASCRIPT

SPINE.JS

MVC FOR JAVASCRIPT

- Spine.js is partly based on Backbone's API yet Spine's take on the MVC pattern is slightly different
- In this section we will recreate the drag and drop shopping cart from Lab 3.2
- Problems with the Lab 3.2 solution:
  - Using the back button or the refresh button loses the carts contents
  - UI callbacks and manual event handling and triggering can quickly get out of hand

SPINE.JS

MVC FOR JAVASCRIPT

- **Models**: In Spine, models are created using the setup method of Spine.Model, which takes the name of the model and an array of properties

```
// Create the Item model.
var Item = Spine.Model.setup("Item", ["name", "pid", "price", "quantity"]);
```

part_4/examples/jquery-spine-shopping-cart/app/models/item.js

SPINE.JS

MVC FOR JAVASCRIPT

- To make the model persists between page reloads we extend the Item with the Spine.Model.Local module

```
// Persist model between page reloads.
Item.extend(Spine.Model.Local);
```

part_4/examples/jquery-spine-shopping-cart/app/models/item.js

- The extend method adds class properties to the model. We now have an object that can be created, saved and retrieved from the browser local storage.

SPINE.JS

MVC FOR JAVASCRIPT

```
// Instance methods
Item.include({
    //
    total: function() {
        return (this.price * this.quantity);
    },
    //
    increase: function(quantity) {
        quantity = (typeof(quantity) != 'undefined') ? quantity : 1;
        this.quantity = this.quantity + quantity;
        this.save();
    },
    //
    decrease: function(quantity) {
        quantity = (typeof(quantity) != 'undefined') ? quantity : 1;
        if (this.quantity >= quantity) {
            this.quantity = this.quantity - quantity;
        }
        else {
            this.quantity = 0;
        }
        this.save();
    },
    //
    label: function() {
        return (this.name + " - $" + this.price);
    }

});
```

part_4/examples/jquery-spine-shopping-cart/app/models/item.js

- To add behavior to our model we use the **include** method which adds instance properties
- The four methods that we need for our Item model; **increase**, **decrease**, **total** and **label**

SPINE.JS

MVC FOR JAVASCRIPT

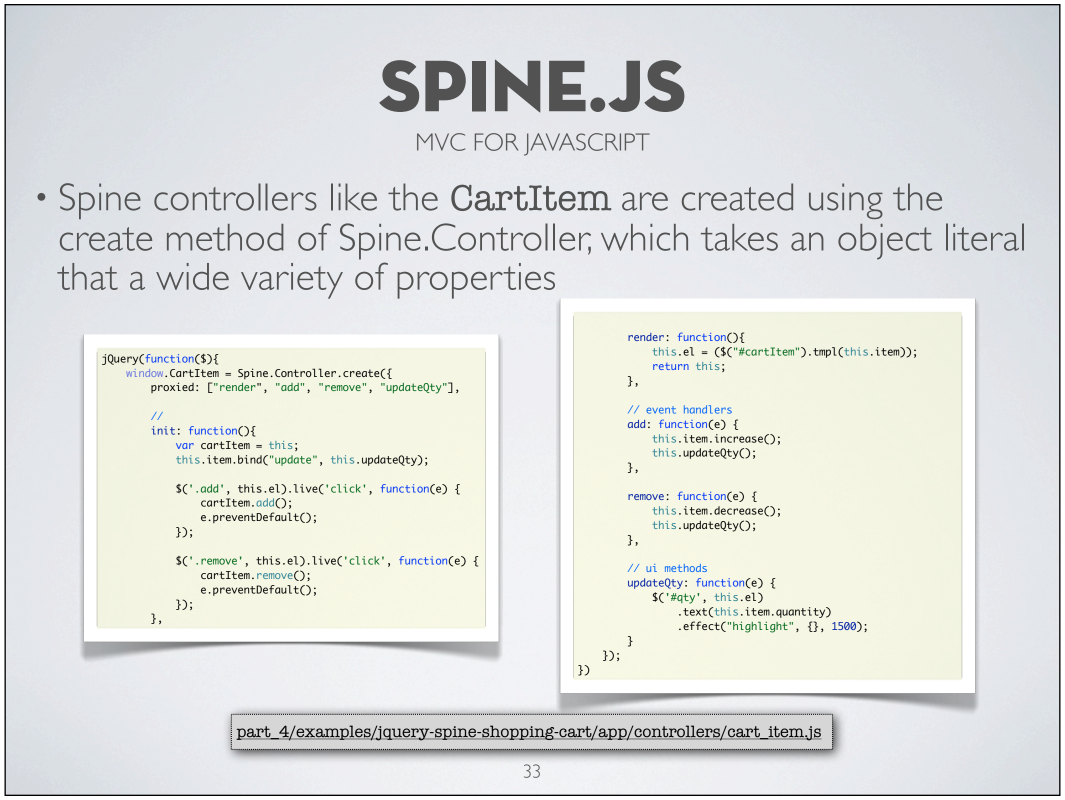
- To instantiate an `Item` we use the create method which takes an object literal for the parameters. A new model can be persisted using the save method. When a model is saved it is assigned an identifier that can be retrieved via the `id` property.

```
var item = Item.create({name: "Product 1", pid: "0001" , price: 100.0, quantity: 1});
item.save();
alert("Just saved Item with id => " + item.id);
```

SPINE.JS

MVC FOR JAVASCRIPT

- **Controllers**: Controllers in Spine are a combination of a traditional MVC controller and a view. Therefore controllers are in charge of rendering and manipulating one or more models in the context of controller's functionality
- Our first Spine controller will deal with the rendering and manipulation of an individual Item. The CartItem controller will deal with user interface events to increase and decrease the quantity of a Item while keeping the user abreast of the changes.



SPINE.JS

MVC FOR JAVASCRIPT

- Spine controllers like the **`CartItem`** are created using the create method of Spine.Controller, which takes an object literal that a wide variety of properties

```
jQuery(function($){
    window.CartItem = Spine.Controller.create({
        proxied: ["render", "add", "remove", "updateQty"],

        //
        init: function(){
            var cartItem = this;
            this.item.bind("update", this.updateQty);

            $('.add', this.el).live('click', function(e) {
                cartItem.add();
                e.preventDefault();
            });

            $('.remove', this.el).live('click', function(e) {
                cartItem.remove();
                e.preventDefault();
            });
        },
```

```
        render: function(){
            this.el = ($("#cartItem").tmpl(this.item));
            return this;
        },

        // event handlers
        add: function(e) {
            this.item.increase();
            this.updateQty();
        },

        remove: function(e) {
            this.item.decrease();
            this.updateQty();
        },

        // ui methods
        updateQty: function(e) {
            $('#qty', this.el)
                .text(this.item.quantity)
                .effect("highlight", {}, 1500);
        }
    });
})
```

part_4/examples/jquery-spine-shopping-cart/app/controllers/cart_item.js

The first property passed is named proxied and takes an array of Strings. The values are the name of the controller methods that need to be guaranteed to execute in the controller's context (otherwise you'll noticed that sometimes the this variable points to something other than the controller).

Events
In the controller's init method we bind the update event of the enclosing Item model to the controller's updateQty method. The model's update method is fired after the model record is updated in the persistence storage.

Also, in the init method, we wire the 'add' and 'remove' links of the CartItem to the add and remove methods which invoke the underlying Item model add and remove as well as the CartItem's updateQty method.

Rendering
To render the associated view for the CartItem controller we use the render method. In the render method we use jQuery to locate the element with id cartItem

SPINE.JS

MVC FOR JAVASCRIPT

- The **cartItem** template is an Underscore.js template (see http://underscorejs.org/#template) that enables the creation of markup templates containing binding expressions. The template is created by calling **_.template("Hello <%= name %>")**. The returned function is them compiled by passing an object containing the properties to be interpolated.

```
<!-- jQuery Template for CartItem -->
<script type="text/template" id="cartItem">
    <li class="product" id="item_<%= pid %>" price="<%= price %>">
        <%= label() %>
        (<span id="qty"><%= quantity %></span>)
        <a href="#" class="add">+</a>
        <a href="#" class="remove">-</a>
    </li>
</script>
```

part_4/examples/jquery-spine-shopping-cart/shopping_cart.html

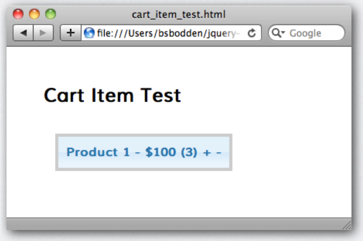
SPINE.JS

MVC FOR JAVASCRIPT

- To test our controller we need to instantiate an Item model and use it to instantiate the controller. We can then render the controller on the DOM of an HTML page

```
var item = Item.create({name: "Product 1", pid: "0001" , price: 100.0, quantity: 1});
var view = CartItem.init({item: item});
$("#item").html(view.render().el);
```

part_4/examples/jquery-spine-shopping-cart/cart_item_test.html

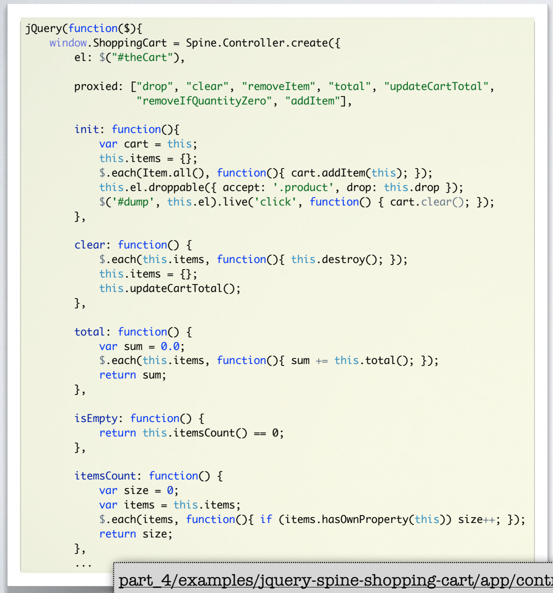


Cart Item Test

Product 1 - $100 (3) + -

SPINE.JS

MVC FOR JAVASCRIPT



```
jQuery(function($){
    window.ShoppingCart = Spine.Controller.create({
        el: $("#theCart"),

        proxied: ["drop", "clear", "removeItem", "total", "updateCartTotal",
                  "removeIfQuantityZero", "addItem"],

        init: function(){
            var cart = this;
            this.items = {};
            $.each(Item.all(), function(){ cart.addItem(this); });
            this.el.droppable({ accept: '.product', drop: this.drop });
            $('#dump', this.el).live('click', function() { cart.clear(); });
        },

        clear: function() {
            $.each(this.items, function(){ this.destroy(); });
            this.items = {};
            this.updateCartTotal();
        },

        total: function() {
            var sum = 0.0;
            $.each(this.items, function(){ sum += this.total(); });
            return sum;
        },

        isEmpty: function() {
            return this.itemsCount() == 0;
        },

        itemsCount: function() {
            var size = 0;
            var items = this.items;
            $.each(items, function(){ if (items.hasOwnProperty(this)) size++; });
            return size;
        },
        ...
```

part_4/examples/jquery-spine-shopping-cart/app/controllers/shopping_cart.js

- The **ShoppingCart** controller which will manage a collection of Item models. Internally the **ShoppingCart** keeps the items dropped in the items property
- The **clear**, **total**, **isEmpty** and **itemsCount** methods fulfill the business functionality of the cart

SPINE.JS

MVC FOR JAVASCRIPT

- The **drop** method handles drop events; it either creates a new item or increases the quantity of an existing item. It creates an Item model that is then rendered using the **CartItem** controller

```
...
drop: function(ev, ui){
    var item_dropped = ui.draggable;
    var pid = item_dropped.attr('id');
    var price = item_dropped.attr('price');
    var name = item_dropped.attr('name');

    if (this.items.hasOwnProperty(pid)) {
        this.items[pid].increase();
    }
    else {
        var item = Item.create({name: name, pid: pid, price: price, quantity: 1});
        this.addItem(item);
        $(".items").append(CartItem.init({item: item}).render().el);
    }
},
```

part_4/examples/jquery-spine-shopping-cart/app/controllers/shopping_cart.js

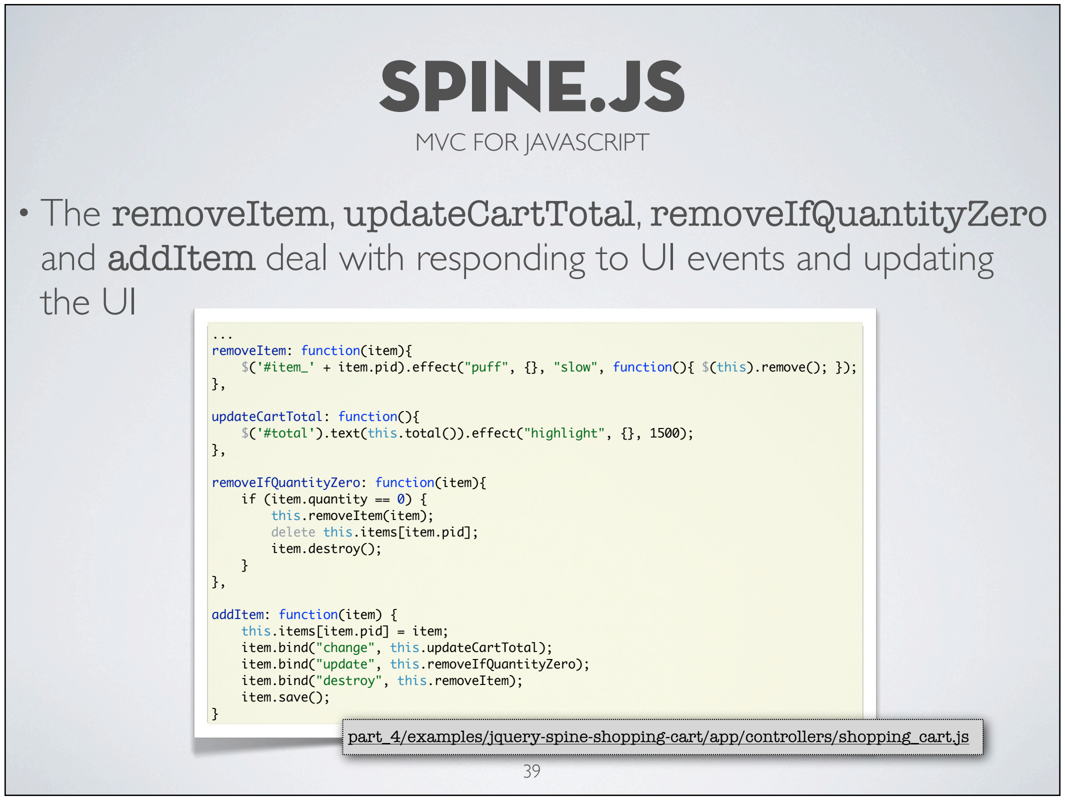
SPINE.JS

MVC FOR JAVASCRIPT

- The **render** method render the **#shoppingCart** template, decorates the **#dump** button and loops through the contained **Item**s creating a **CartItem** for each and rendering them in the **.items**

```
...
render: function(ev, ui){
    this.el.html($("#shoppingCart").tmpl());

    $('#dump').button();

    $.each(this.items, function(){
        $(".items").append(CartItem.init({item: this}).render().el);
    });

    this.updateCartTotal();
},
```

part_4/examples/jquery-spine-shopping-cart/app/controllers/shopping_cart.js



SPINE.JS

MVC FOR JAVASCRIPT

- The **removeItem**, **updateCartTotal**, **removeIfQuantityZero** and **addItem** deal with responding to UI events and updating the UI

```
...
removeItem: function(item){
    $('#item_' + item.pid).effect("puff", {}, "slow", function(){ $(this).remove(); });
},

updateCartTotal: function(){
    $('#total').text(this.total()).effect("highlight", {}, 1500);
},

removeIfQuantityZero: function(item){
    if (item.quantity == 0) {
        this.removeItem(item);
        delete this.items[item.pid];
        item.destroy();
    }
},

addItem: function(item) {
    this.items[item.pid] = item;
    item.bind("change", this.updateCartTotal);
    item.bind("update", this.removeIfQuantityZero);
    item.bind("destroy", this.removeItem);
    item.save();
}
```

part_4/examples/jquery-spine-shopping-cart/app/controllers/shopping_cart.js

Note that in addItem we save the created models so that we can later retrieve them in case of a page refresh.

SPINE.JS

MVC FOR JAVASCRIPT

- Finally to kick everything in motion the file **application.js** file begins by fetching any previously stored **Item** records, making the sample products draggable, creating a cart and rendering it.

```
jQuery(function($){
   Item.fetch();
   $(".product").draggable({ helper: 'clone', opacity: "0.5" });
   var cart = ShoppingCart.init();
   cart.render();
});
```

part_4/examples/jquery-spine-shopping-cart/app/application.js

JQUERY UTILITIES

MISCELLANEOUS UTILITIES

JQUERY UTILITIES

MISCELLANEOUS UTILITIES

- jQuery provides a variety of utility methods available directly from the jQuery function
- For full documentation see http://api.jquery.com/category/utilities/

clearQueue	globalEval	isPlainObject	noop	removeData
contains	grep	isWindow	now	support
data	inArray	isXMlDoc	parseJSON	trim
dequeue	isArray	makeArray	parseXML	type
each	isEmptyObject	map	proxy	unique
extend	isFunction	merge	queue	

THEMES AND THEMING

JQUERY UI THEME FRAMEWORK

THEMES AND THEMING

JQUERY UI THEME ROLLER

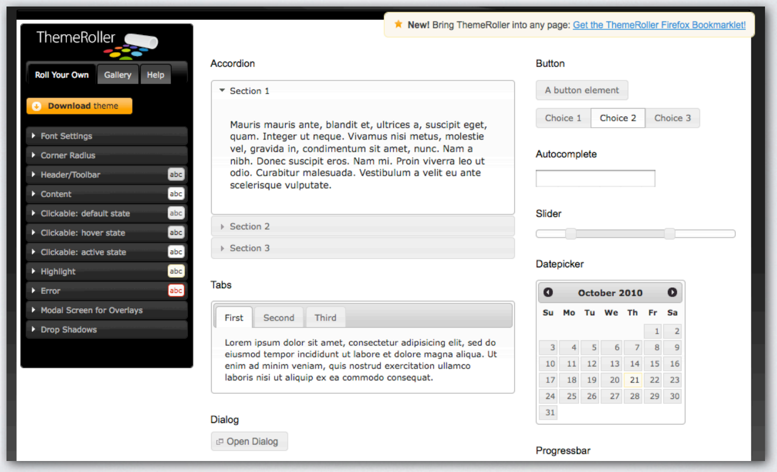
- jQuery UI provides a web based tool called ThemeRoller
- ThemeRoller allows you to design a custom jQuery UI theme
- jQuery UI also provides a gallery of pre-design themes (and there are also themes available elsewhere)
- All official jQuery UI widgets make use of the CSS classes and ids configured by a theme

THEMES AND THEMING

FOLLOW ALONG

JQUERY UI THEME ROLLER

- ThemeRoller is found at http://jqueryui.com/themeroller/


★ New! Bring ThemeRoller into any page: Get the ThemeRoller Firefox Bookmarklet!

Mauris mauris ante, blandit et, ultrices a, suscipit eget, quam. Integer ut neque. Vivamus nisi metus, molestie vel, gravida in, condimentum sit amet, nunc. Nam a nibh. Donec suscipit eros. Nam mi. Proin viverra leo ut odio. Curabitur malesuada. Vestibulum a velit eu ante scelerisque vulputate.

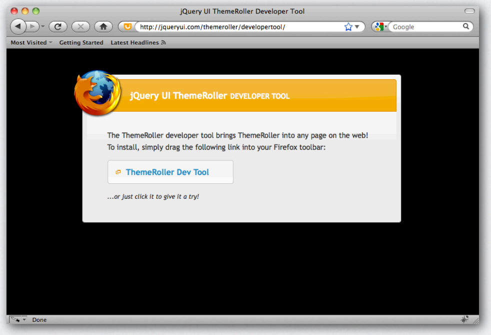
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

THEMES AND THEMING

Follow Along

JQUERY UI THEME ROLLER

- The ThemeRoller Developer Tool is also available as a bookmarklet found at http://jqueryui.com/themeroller/developertool/
- The ThemeRoller Developer Tools is useful to use on an existing page for which you want to mimic the look and feel in a jQuery UI Theme


jQuery UI ThemeRoller Developer Tool
http://jqueryui.com/themeroller/developertool/
Google
Most Visited
Getting Started
Latest Headlines
Done

jQuery UI ThemeRoller DEVELOPER TOOL

The ThemeRoller developer tool brings ThemeRoller into any page on the web!
To install, simply drag the following link into your Firefox toolbar:

ThemeRoller Dev Tool

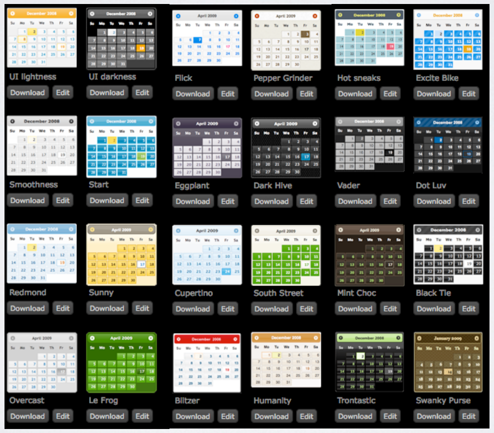
...or just click it to give it a try!

THEMES AND THEMING

Follow Along

JQUERY UI THEME ROLLER

- Available theme's can be used by name by downloading the selected theme or using a CDN
  - http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.13/themes/{THEME}/jquery-ui.css

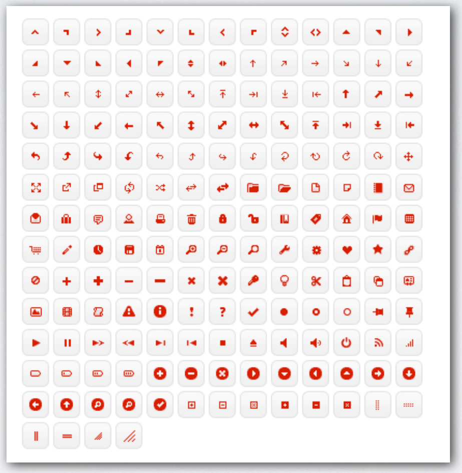


THEMES AND THEMING

Follow Along

JQUERY UI THEME ROLLER

- jQuery UI gallery themes provide icon collections themed accordingly:



THEMES AND THEMING

JQUERY UI THEME ROLLER

- jQuery UI CSS follow some standards
  - Classes and ID prefixed with ui-
  - Widget-specific settings
  - Framework classes (corners, states)
  - Theme icon collections

THEMES AND THEMING

JQUERY UI THEME ROLLER

- jQuery UI provides a Theme Switcher widget that allows you to change jQuery UI themes in your applications:

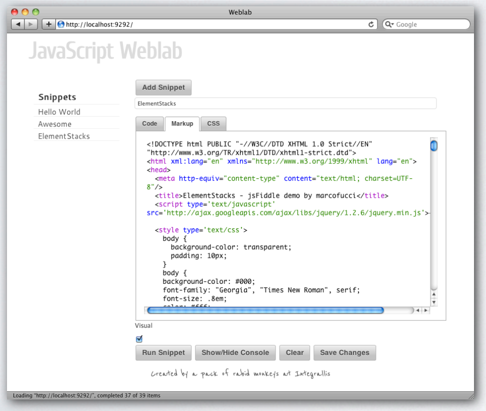
```
$(document).ready(function() {
    // jquery ui theme switcher
    $('#switcher').themeswitcher();
}
```

```
<div id="switcher"></div>
```

THEMES AND THEMING

JQUERY UI THEME ROLLER

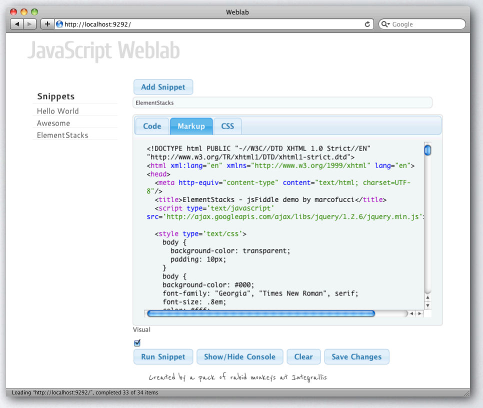
- We'll **demo** a simple Ruby on Rails application that uses jQuery and jQuery UI and customizable themes:



ElementStacks

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xml:lang="en" xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-
8"/>
  <title>ElementStacks - jsFiddle demo by marcofucci</title>
  <script type='text/javascript'
src='http://ajax.googleapis.com/ajax/libs/jquery/1.2.6/jquery.min.js'>

  <style type='text/css'>
    body {
      background-color: transparent;
      padding: 10px;
    }
    body {
    background-color: #000;
    font-family: "Georgia", "Times New Roman", serif;
    font-size: .8em;
```

Loading "http://localhost:9292/", completed 37 of 39 items



ElementStacks

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xml:lang="en" xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-
8"/>
  <title>ElementStacks - jsFiddle demo by marcofucci</title>
  <script type='text/javascript'
src='http://ajax.googleapis.com/ajax/libs/jquery/1.2.6/jquery.min.js'

  <style type='text/css'>
    body {
      background-color: transparent;
      padding: 10px;
    }
    body {
    background-color: #000;
    font-family: "Georgia", "Times New Roman", serif;
    font-size: .8em;
```

Loading "http://localhost:9292/", completed 33 of 34 items

LAB 4.2

THEMING

- Modify Lab 3.1 to
  - add the JQuery UI Theme Switcher tool
  - OR
  - Apply one of the available themes in jQuery UI

JQUERY NOCONFLICT

AVOIDING NAMESPACE CLASHES

JQUERY NO CONFLICT

AVOIDING NAMESPACE CONFLICTS

- We learned that $ is an alias for the jQuery object/function that allows us to be brief in our jQuery endeavors
- The use of $ as a namespace is not unique to jQuery, when working with other libraries jQuery provides the noConflict function
- noConflict de-associates the $ from the jQuery object therefore relinquishing it to be used by others

```
// give $ back
jQuery.noConflict();

// use jQuery
jQuery("p").hide();

// this wouldn't work
$("p").show();
```

part_4/examples/no_conflict.js

JQUERY NO CONFLICT

AVOIDING NAMESPACE CONFLICTS

- Of course you could use functional scope to re-associate $ with jQuery in the context of a snippet of code:

```
jQuery.noConflict();

(function($) {
    $(function() {
        // this now works work
        $("p").show();
    });
})(jQuery);
```

part_4/examples/no_conflict_regain.js

JASMINE

BDD WITH JAVASCRIPT

BDD

BEHAVIOR-DRIVEN DEVELOPMENT

- BDD focuses on getting the language right, the resulting specifications become a runnable/self-verifying form of documentation
- BDD specifications follow a format that makes them easy to be driven by your system's User Stories

*As a [**role**], I want [**goal**] so that [**benefit**]*

User Story

*Given [**role and its state**]*
*When [**an event/action occurs**]*
*Then [**the benefit**]*

Specification

JASMINE

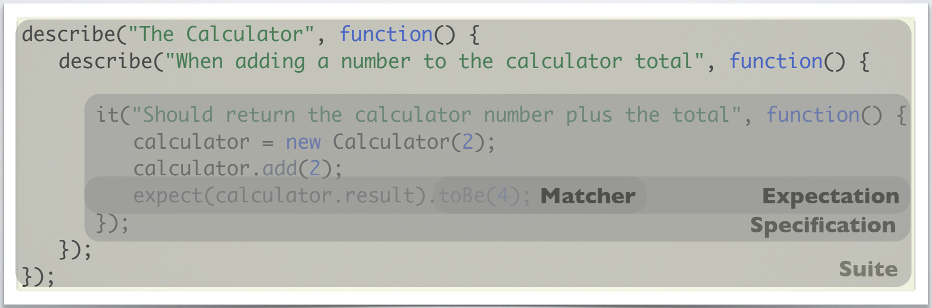
BDD FOR JAVASCRIPT

- Jasmine is a JavaScript testing framework
- Jasmine is specially useful if used to apply Behavior-Driven Development (BDD) techniques
- Heavily inspired by Ruby's RSpec
- No dependencies on other frameworks
- Doesn't need a DOM present to work
- You can get Jasmine from http://pivotal.github.com/jasmine/
- Available as a standalone JS file, a Ruby Gem

JASMINE

BDD FOR JAVASCRIPT

- A Jasmine Specification is part of a ***Suite***
- A Specification verifies behavior using ***Expectations***
- Jasmine provides several ***Matchers*** to express your Expectations



```
describe("The Calculator", function() {
   describe("When adding a number to the calculator total", function() {

      it("Should return the calculator number plus the total", function() {
         calculator = new Calculator(2);
         calculator.add(2);
         expect(calculator.result).toBe(4);
      });
   });
});
```

JASMINE

BDD FOR JAVASCRIPT

- Jasmine provides several matchers:

toBe(expected)	*toHaveBeenCalled()*
toBeCloseTo(expected, precision)	*toHaveBeenCalledWith()*
toBeDefined()	*toMatch(expected)*
toBeFalsy()	*toNotBe(expected)*
toBeGreaterThan(expected)	*toNotContain(expected)*
toBeLessThan(expected)	*toNotEqual(expected)*
toBeNull()	*toNotMatch(expected)*
toBeTruthy()	*toThrow(expected)*
toBeUndefined()	*wasNotCalled()*
toContain(expected)	*wasNotCalledWith()*
toEqual(expected)	

JASMINE

BDD FOR JAVASCRIPT

- Specifications read like sentences with a clean and simple structured and more importantly; focused!
- The suites organized specifications around a set of know initial conditions/states
- The closer to the outside the test is the simpler it becomes to describe the "benefit"
- You'll notice that to describe inner "benefits" you require more and more technical terms and to implement the spec them you'll require mocks, stubs and spies

JASMINE

BDD FOR JAVASCRIPT

- Jasmine is a Behavior-Driven Development (BDD) Testing Framework for JavaScript inspired by Ruby's RSpec
- Let's look at a test for the **Stack** created previously in the course:

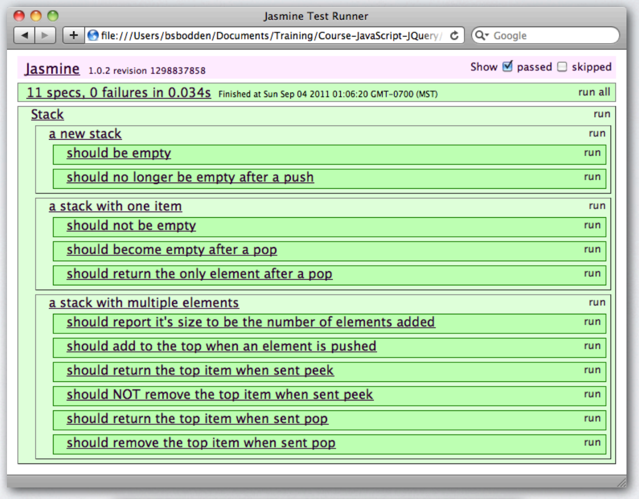
```
describe("Stack", function() {
    var stack;

    beforeEach(function() {
        stack = new INTEGRALLIS.structures.Stack();
    });

    describe("a new stack", function() {

        it("should be empty", function() {
            expect(stack.isEmpty()).toBeTruthy();
        });

        it("should no longer be empty after a push", function() {
            stack.push("anything");
            expect(stack.isEmpty()).toBeFalsy();
        });

    });
...
```

part_4/examples/jasmine/spec/StackSpec.js

JASMINE

BDD FOR JAVASCRIPT

- Running the specs:



```
Jasmine Test Runner
file:///Users/bsbodden/Documents/Training/Course-JavaScript-JQuery/

Jasmine 1.0.2 revision 1298837858                    Show [✓ passed] skipped
11 specs, 0 failures in 0.034s  Finished at Sun Sep 04 2011 01:06:20 GMT-0700 (MST)    run all
Stack                                                                                   run
  a new stack                                                                           run
    should be empty                                                                     run
    should no longer be empty after a push                                              run
  a stack with one item                                                                 run
    should not be empty                                                                 run
    should become empty after a pop                                                     run
    should return the only element after a pop                                          run
  a stack with multiple elements                                                        run
    should report it's size to be the number of elements added                          run
    should add to the top when an element is pushed                                    run
    should return the top item when sent peek                                           run
    should NOT remove the top item when sent peek                                       run
    should return the top item when sent pop                                            run
    should remove the top item when sent pop                                            run
```

part_4/examples/jasmine/SpecRunner.html

TDD WITH BDD

DRIVING YOUR JAVASCRIPT DESIGNS WITH JASMINE

TDD WITH BDD

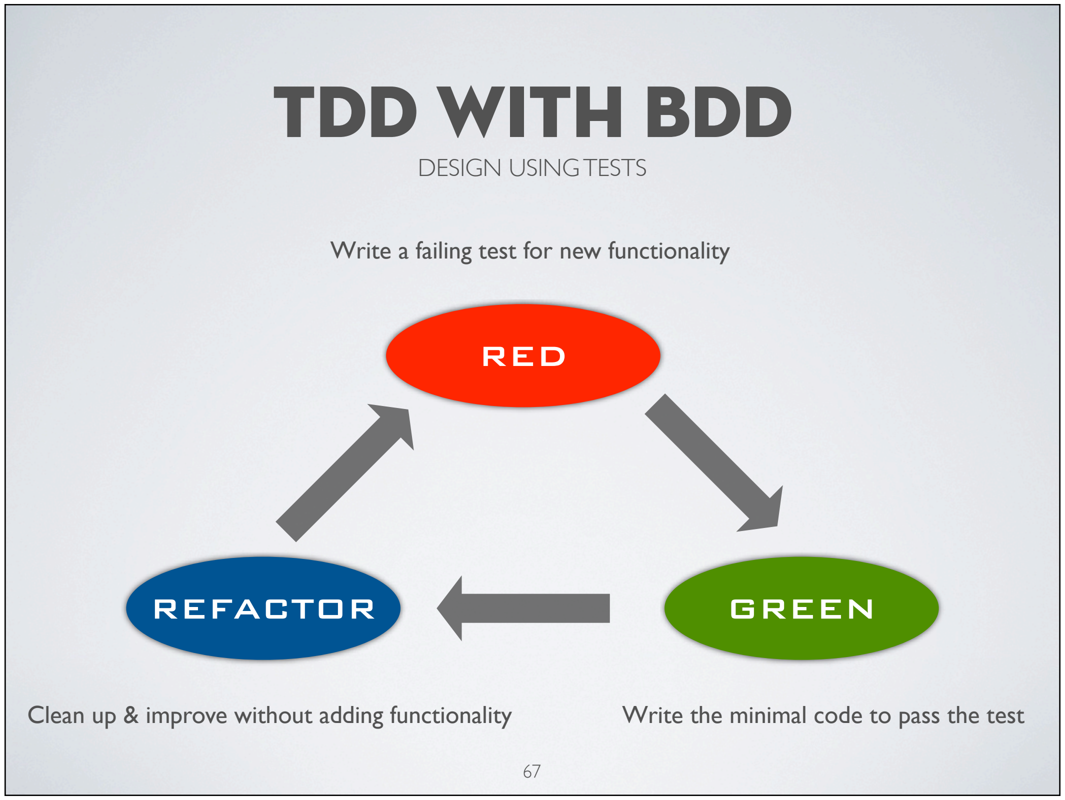
DESIGN USING TESTS

- TDD is not *really* about testing
- TDD is a design technique
- TDD leads to cleaner code with separation of concerns
- Cleaner code is more reliable and easier to maintain

TDD WITH BDD

DESIGN USING TESTS

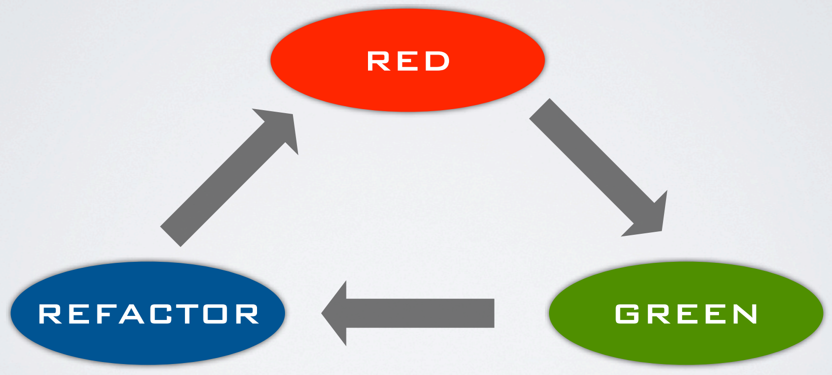
- TDD creates a tight loop of development that cognitively engages us
- TDD gives us lightweight rigor by making development, goal-oriented with a clear goal setting, goal reaching and improvement stages
- The stages of TDD are commonly known as the Red-Green-Refactor loop



TDD WITH BDD

DESIGN USING TESTS

Write a failing test for new functionality



Clean up & improve without adding functionality

Write the minimal code to pass the test

TDD WITH BDD

DESIGN USING TESTS

“As a User of the Calculator I should be able to add two numbers”

TDD WITH BDD

DESIGN USING TESTS

- Let's start with a simple test case for our non-existing calculator:

```
describe("The Calculator", function() {
   describe("When adding two numbers", function() {

      it("Should return the total of adding the numbers", function() {
         calculator = new Calculator();
         calculator.add(2, 2);
         expect(calculator.result).toBe(4);
      });
   });
});
```

CalculatorSpec.js

TDD WITH BDD

DESIGN USING TESTS

- To run the test we can use the TrivialReporter

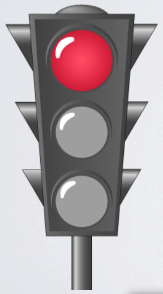
```
<html>
<head>
  <title>Calculator Specs</title>
  <link rel="stylesheet" type="text/css" href="lib/jasmine-1.1.0/jasmine.css">
  <script type="text/javascript" src="lib/jasmine-1.1.0/jasmine.js"></script>
  <script type="text/javascript" src="lib/jasmine-1.1.0/jasmine-html.js"></script>

  <!-- include source files here... -->
  <script type="text/javascript" src="src/calculator.js"></script>

  <!-- include spec files here... -->
  <script type="text/javascript" src="spec/CalculatorSpec.js"></script>

</head>
<body>
<script type="text/javascript">
jasmine.getEnv().addReporter(new jasmine.TrivialReporter());
jasmine.getEnv().execute();
</script>
</body>
</html>
```

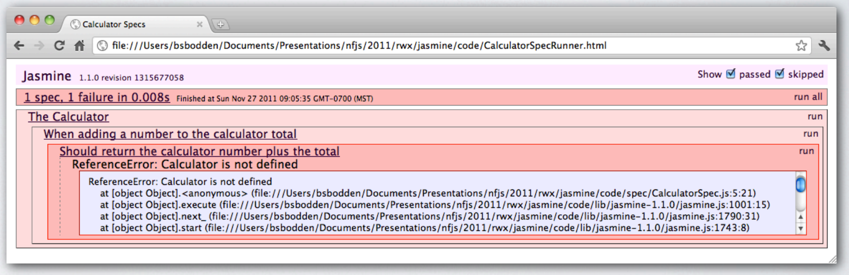
CalculatorSpecRunner.html

TDD WITH BDD

DESIGN USING TESTS

- Start with a simple test case for the non-existing Calculator
- We have reached the RED state in our red-green-refactor cycle





Calculator Specs

file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/CalculatorSpecRunner.html

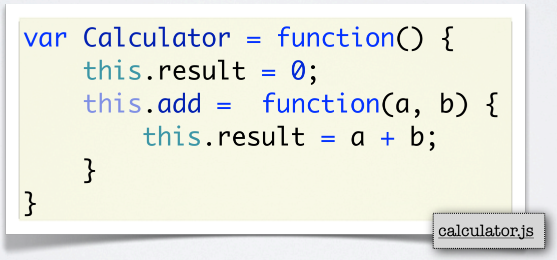
```
Jasmine 1.1.0 revision 1315677058                                   Show ☑ passed ☑ skipped
1 spec, 1 failure in 0.008s Finished at Sun Nov 27 2011 09:05:35 GMT-0700 (MST)     run all
The Calculator                                                                       run
  When adding a number to the calculator total                                       run
    Should return the calculator number plus the total                               run
      ReferenceError: Calculator is not defined
        ReferenceError: Calculator is not defined
          at [object Object].<anonymous> (file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/spec/CalculatorSpec.js:5:21)
          at [object Object].execute (file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/lib/jasmine-1.1.0/jasmine.js:1001:15)
          at [object Object].next_ (file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/lib/jasmine-1.1.0/jasmine.js:1790:31)
          at [object Object].start (file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/lib/jasmine-1.1.0/jasmine.js:1743:8)
```

We have a failing test at that is a good thing!

TDD WITH BDD

DESIGN USING TESTS

- The simple implementation of the **`Calculator`** shown below will serve our purposes
- To get the GREEN state it provides a simple implementation of the **`add`** method and **`result`** properties



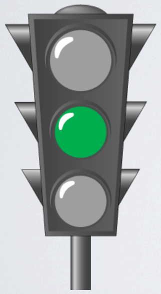
```
var Calculator = function() {
    this.result = 0;
    this.add =  function(a, b) {
        this.result = a + b;
    }
}
```

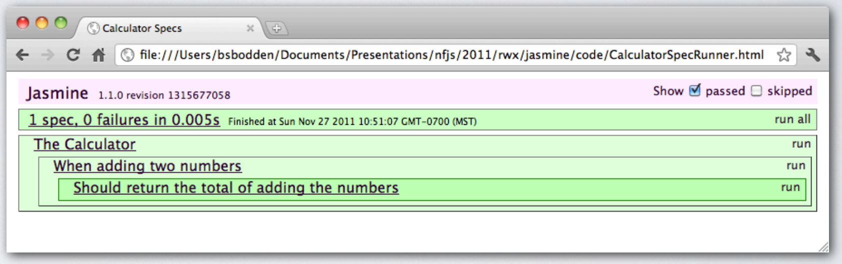
calculator.js

TDD WITH BDD

DESIGN USING TESTS

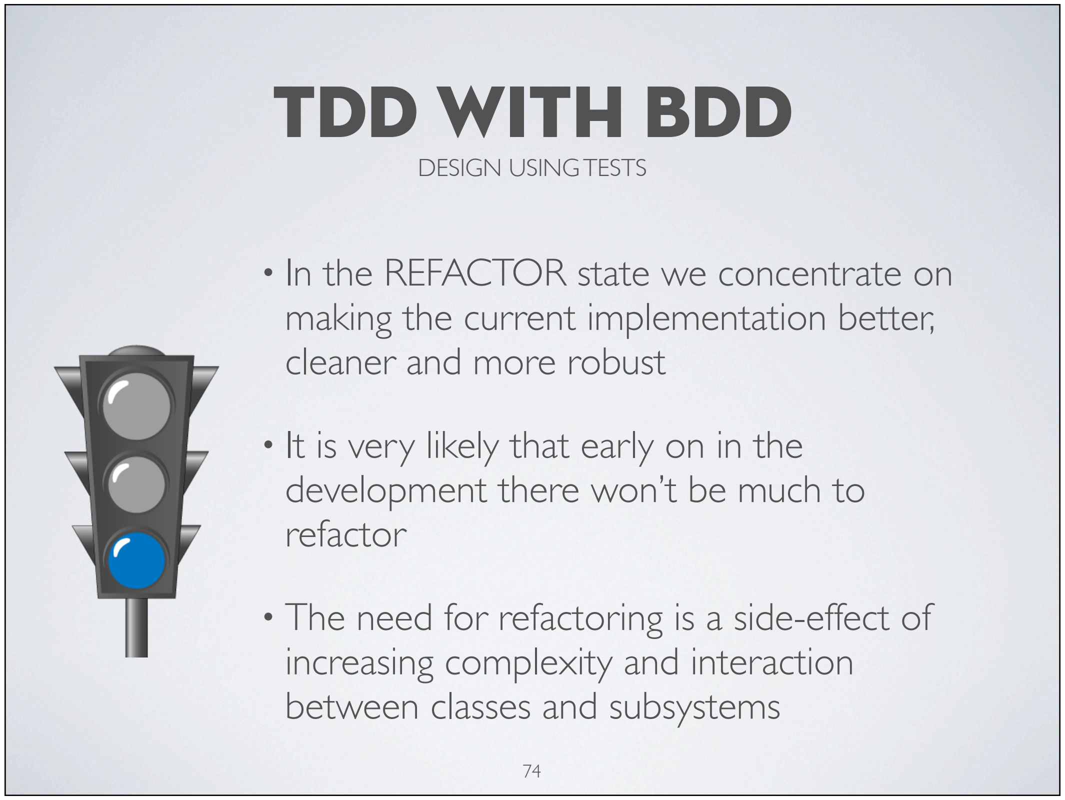
- Let's run the test again:




Calculator Specs
file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/CalculatorSpecRunner.html
Show ✓ passed ☐ skipped
run all
run

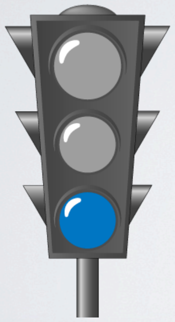
```
Jasmine 1.1.0 revision 1315677058
1 spec, 0 failures in 0.005s Finished at Sun Nov 27 2011 10:51:07 GMT-0700 (MST)
The Calculator
  When adding two numbers
    Should return the total of adding the numbers
```

We've reached the GREEN state



TDD WITH BDD

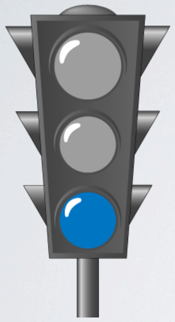
DESIGN USING TESTS



- In the REFACTOR state we concentrate on making the current implementation better, cleaner and more robust
- It is very likely that early on in the development there won't be much to refactor
- The need for refactoring is a side-effect of increasing complexity and interaction between classes and subsystems

TDD WITH BDD

DESIGN USING TESTS



- For the purposes of this exercise, let's assume that our `Calculator` stakeholders have requested that we provide the add method with the ability to add more than two digits in a single call

- **Write** a new spec will express the intent of our refactoring/enhancement

"As a User of the Calculator I should be able to add one or more numbers"

TDD WITH BDD

DESIGN USING TESTS

- Let's state the intent of our new user story with a Jasmine specification (a pending specification):

```
describe("The Calculator", function() {
   ...

   describe("When adding one or more numbers", function() {

      xit("Should return the total of adding the numbers", function() {
         calculator = new Calculator();
         calculator.add(1, 2, 3, 4, 5, 6, 7, 8, 9);
         expect(calculator.result).toBe(45);
      });
   });
});
```

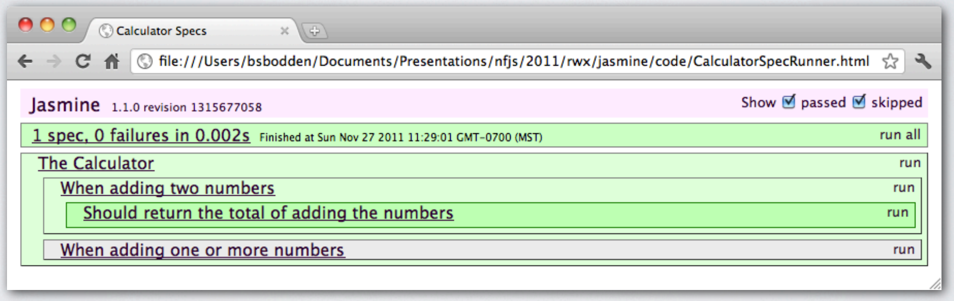
CalculatorSpec.js

The method defines a skipped or "pending" spec

TDD WITH BDD

DESIGN USING TESTS

- Running the specs again we can see that new "skipped/ pending" spec:

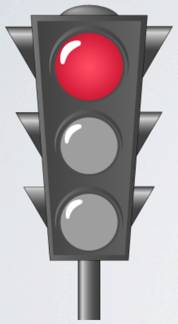

Calculator Specs
file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/CalculatorSpecRunner.html

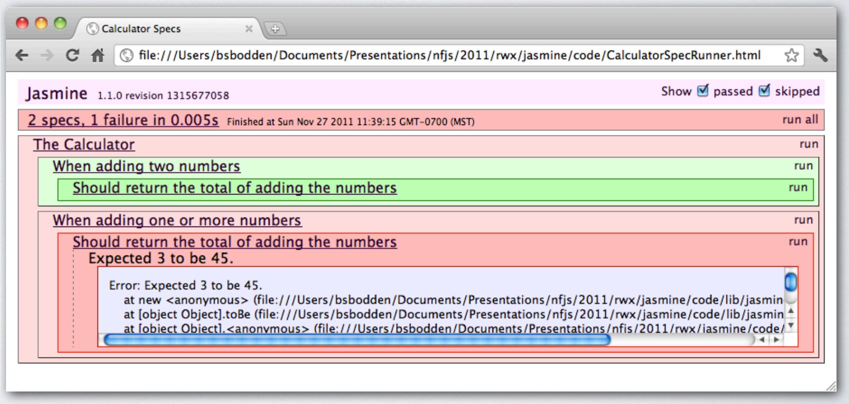
```
Jasmine 1.1.0 revision 1315677058                          Show ☑ passed ☑ skipped
1 spec, 0 failures in 0.002s Finished at Sun Nov 27 2011 11:29:01 GMT-0700 (MST)   run all
The Calculator                                                                  run
  When adding two numbers                                                       run
    Should return the total of adding the numbers                               run
  When adding one or more numbers                                               run
```

TDD WITH BDD

DESIGN USING TESTS

- **Run** the tests again reveals the missing functionality





Calculator Specs
file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/CalculatorSpecRunner.html

Jasmine 1.1.0 revision 1315677058 Show ☑ passed ☑ skipped

2 specs, 1 failure in 0.005s Finished at Sun Nov 27 2011 11:39:15 GMT-0700 (MST) run all

The Calculator run

When adding two numbers run

Should return the total of adding the numbers run

When adding one or more numbers run

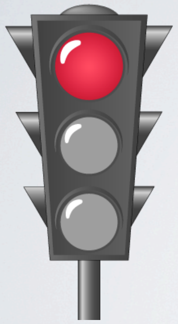
Should return the total of adding the numbers run

Expected 3 to be 45.

```
Error: Expected 3 to be 45.
    at new <anonymous> (file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/lib/jasmin
    at [object Object].toBe (file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/lib/jasmin
    at [object Object].<anonymous> (file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/
```

TDD WITH BDD

DESIGN USING TESTS



- Now we can "refactor without fear" with the previous test protecting us from breaking existing functionality
- **Change** the `add` method implementation allow for multiple parameters

TDD WITH BDD

DESIGN USING TESTS

- Let's now fearlessly refactor the add method to deal with multiple numbers being added:

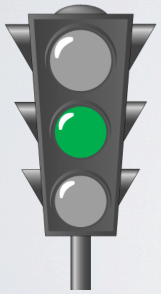
```
var Calculator = function() {
    this.result = 0;
    this.add =  function() {
        for (var i = arguments.length - 1; i >= 0; i--){
            total = total + arguments[i];
        };
        this.result = total;
    }
}
```

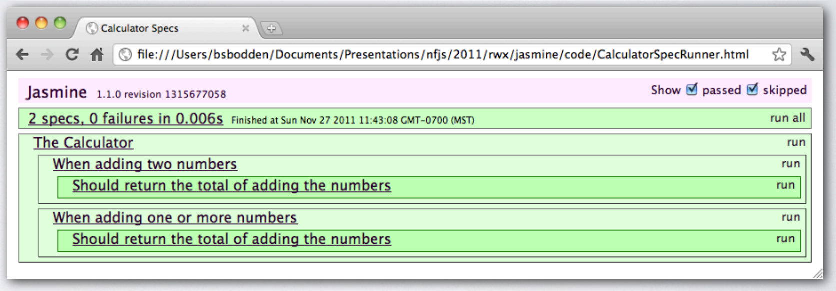
calculator.js

TDD WITH BDD

DESIGN USING TESTS

- Let's running the tests again reveals that both our existing spec and the newly added spec now pass





Calculator Specs
file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/CalculatorSpecRunner.html

```
Jasmine 1.1.0 revision 1315677058                                Show ☑ passed ☑ skipped
2 specs, 0 failures in 0.006s Finished at Sun Nov 27 2011 11:43:08 GMT-0700 (MST)   run all
The Calculator                                                                       run
  When adding two numbers                                                            run
    Should return the total of adding the numbers                                    run
  When adding one or more numbers                                                    run
    Should return the total of adding the numbers                                    run
```

TDD WITH BDD

DESIGN USING TESTS

- Long story short, now the Calculator has been enhanced using BDD/TDD techniques to fulfill the following stories:

“As a User of the Calculator I should be able to keep a running total”

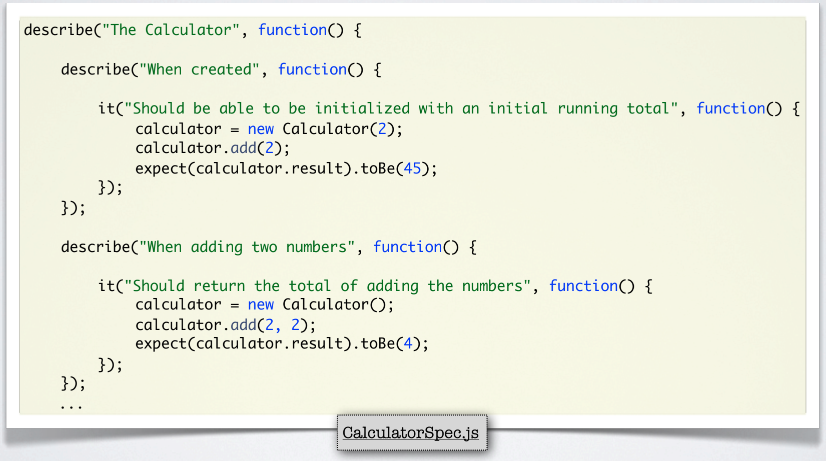
“As a User of the Calculator I should be able to reset the running total”

“As a User of the Calculator I should be able to initialize a running total”

TDD WITH BDD

DESIGN USING TESTS

- The new specs are added one by one as we refactor the code:



```
describe("The Calculator", function() {

    describe("When created", function() {

        it("Should be able to be initialized with an initial running total", function() {
            calculator = new Calculator(2);
            calculator.add(2);
            expect(calculator.result).toBe(45);
        });
    });

    describe("When adding two numbers", function() {

        it("Should return the total of adding the numbers", function() {
            calculator = new Calculator();
            calculator.add(2, 2);
            expect(calculator.result).toBe(4);
        });
    });
    ...
```

TDD WITH BDD

DESIGN USING TESTS

```
    ...
    describe("When adding one or more numbers", function() {

        it("Should return the total of adding the numbers", function() {
            calculator = new Calculator();
            calculator.add(1, 2, 3, 4, 5, 6, 7, 8, 9);
            expect(calculator.result).toBe(45);
        });
    });

    describe("When performing any operations", function() {

        it("Should keep a running total", function() {
            calculator = new Calculator();
            calculator.add(2, 2);
            calculator.add(5);
            expect(calculator.result).toBe(9);
        });

        it("Should be able to reset the running total", function() {
            calculator = new Calculator(2);
            calculator.reset();
            expect(calculator.result).toBe(0);
        });
    });
});
```

CalculatorSpec.js

TDD WITH BDD

DESIGN USING TESTS

- The version of **calculator.js** below shows the accumulated refactoring to fulfill the new user stories:

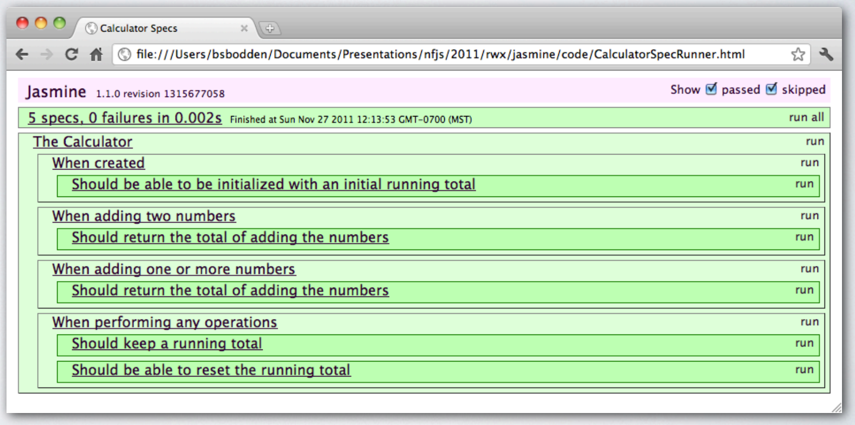
```
var Calculator = function(initial) {
    this.result = = initial || 0.0;
    this.add =  function() {
        for (var i = arguments.length - 1; i >= 0; i--){
            total = total + arguments[i];
        };
        this.result = this.result + total;
    }
}
```

calculator.js

TDD WITH BDD

DESIGN USING TESTS

- The final set of specs running:



Calculator Specs

file:///Users/bsbodden/Documents/Presentations/nfjs/2011/rwx/jasmine/code/CalculatorSpecRunner.html

```
Jasmine 1.1.0 revision 1315677058                                   Show ☑ passed ☑ skipped
5 specs, 0 failures in 0.002s  Finished at Sun Nov 27 2011 12:13:53 GMT-0700 (MST)    run all
The Calculator                                                                        run
  When created                                                                        run
    Should be able to be initialized with an initial running total                    run
  When adding two numbers                                                             run
    Should return the total of adding the numbers                                     run
  When adding one or more numbers                                                     run
    Should return the total of adding the numbers                                     run
  When performing any operations                                                      run
    Should keep a running total                                                       run
    Should be able to reset the running total                                         run
```

BEFORE AND AFTER EACH

USING PRE AND POST CONDITIONS

BEFORE EACH

ORGANIZING YOUR SPECS

- We can refactor the common initialization code in some of our specs by using the **beforeEach** function:

```
describe("The Calculator", function() {
    var calculator;

    beforeEach(function() {
        calculator = new Calculator();
    });

    describe("When adding two numbers", function() {

        it("Should return the total of adding the numbers", function() {
            calculator.add(2, 2);
            expect(calculator.result).toBe(4);
        });
    });

    describe("When adding one or more numbers", function() {

        it("Should return the total of adding the numbers", function() {
            calculator.add(1, 2, 3, 4, 5, 6, 7, 8, 9);
            expect(calculator.result).toBe(45);
        });
    });
```

BEFORE EACH

ORGANIZING YOUR SPECS

- We could also using a single constructor call and use the **`afterEach`** function to reset the calculator:

```
describe("The Calculator", function() {
    var calculator = new Calculator();

    afterEach(function() {
        calculator.reset();
    });

    describe("When adding two numbers", function() {

        it("Should return the total of adding the numbers", function() {
            calculator.add(2, 2);
            expect(calculator.result).toBe(4);
        });
    });

    describe("When adding one or more numbers", function() {

        it("Should return the total of adding the numbers", function() {
            calculator.add(1, 2, 3, 4, 5, 6, 7, 8, 9);
            expect(calculator.result).toBe(45);
        });
    });
```

JASMINE SPIES

BEHAVIOR VERSUS STATE TESTING

SPIES

TESTING BEHAVIOR AGAINST COLLABORATORS

- Supposed we have some visual component as shown below that takes a listener object

```
var MyComponent = function(listener) {
   this.listener = (typeof listener == 'undefined') ? defaultListener : listener;

   // callbacks
   var defaultListener = {
      beforeRender: function() {},
      afterRender: function() {}
   };

    this.render =  function() {
      this.listener.beforeRender();
      alert("Rendering!");
      this.listener.afterRender();
   }
}
```

SPIES

TESTING BEHAVIOR AGAINST COLLABORATORS

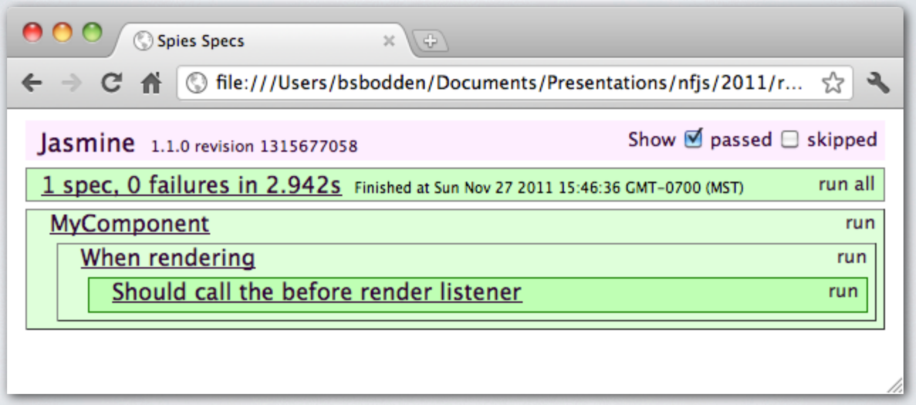
- We can write a spec to assert that the correct listener method is invoked:

```
describe("MyComponent", function() {

    describe("When rendering", function() {
        it("Should call the before render listener", function() {
            listener = {
                beforeRender: function() { /* do something */ },
                afterRender: function() { /* do something */ }
            };

            component = new MyComponent(listener);

            spyOn(listener, "beforeRender");
            component.render();
            expect(listener.beforeRender).toHaveBeenCalled();
        });
    });
});
```

SPIES

TESTING BEHAVIOR AGAINST COLLABORATORS

- The Spies Specs running:



Spies Specs
file:///Users/bsbodden/Documents/Presentations/nfjs/2011/r...

Jasmine 1.1.0 revision 1315677058

Show [✓ passed] skipped

1 spec, 0 failures in 2.942s Finished at Sun Nov 27 2011 15:46:36 GMT-0700 (MST) run all

MyComponent run

When rendering run

Should call the before render listener run

JQUERY MOBILE

CROSS-BROWSER MOBILE WEB DEVELOPMENT

JQUERY MOBILE

CROSS-BROWSER MOBILE WEB DEVELOPMENT

- There is a multitude of devices to target all with their own set of capabilities (and quirks)
- Apple's iPhone, iPad and iPod, Android, Blackberry, Windows Phone, Symbian, Palm, Meego, Kindle
- Building native apps (although ideally suited to take advantages of the underlying platform) is costly, skilled developers are hard to find and you'll need multiple code bases in multiple languages to be maintained and synchronized

JQUERY MOBILE

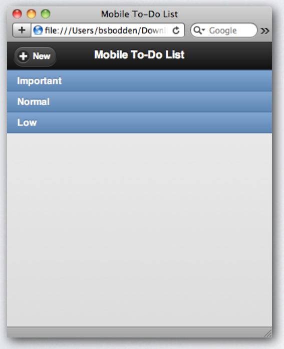
CROSS-BROWSER MOBILE WEB DEVELOPMENT

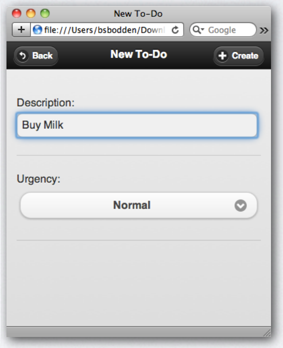
- jQuery Mobile...
    - Is a Multi-Platform Mobile Web Framework built on top of jQuery
    - Is Touch-optimized and Theme-able
    - Provides a unified user interface, widget library and touch events
    - Lightweight (< 15k), Accessible (WAI-ARIA), Progressive Enhancement and Degradable based on device capabilities

JQUERY MOBILE

CROSS-BROWSER MOBILE WEB DEVELOPMENT

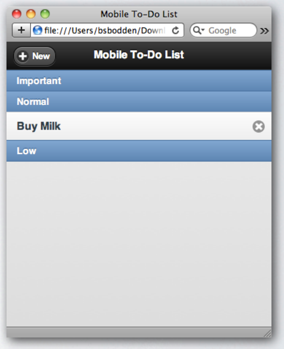
- Let's take a look at a few examples, To-Do List:





Description: Buy Milk

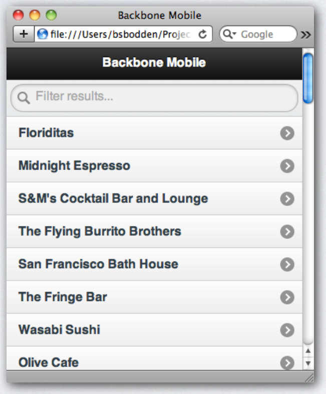
Urgency: Normal



JQUERY MOBILE

CROSS-BROWSER MOBILE WEB DEVELOPMENT

- Backbone + jQuery Mobile:



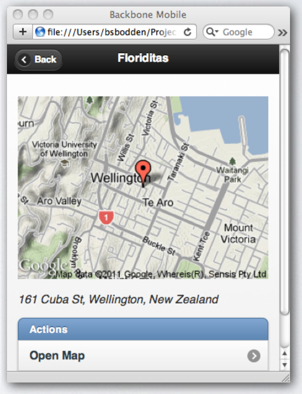
Backbone Mobile

file:///Users/bsbodden/Projec

Backbone Mobile

Filter results...

- Floriditas
- Midnight Espresso
- S&M's Cocktail Bar and Lounge
- The Flying Burrito Brothers
- San Francisco Bath House
- The Fringe Bar
- Wasabi Sushi
- Olive Cafe



Backbone Mobile

file:///Users/bsbodden/Projec

Back

Floriditas

Victoria University of Wellington
Wellington
Aro Valley
Te Aro
Waitangi Park
Mount Victoria
Map data ©2011 Google, Whereis(R), Sensis Pty Ltd

161 Cuba St, Wellington, New Zealand

Actions

Open Map

RESOURCES

BOOKS, TOOLS AND SITES OF INTEREST

TOOLS

JAVASCRIPT TOOLS

Tool	Description	URL
JSLint	quality	http://www.jslint.com/
JSHint	quality	http://jshint.com/
YUI Compressor	compression	http://developer.yahoo.com/yui/compressor/
JSLitmus	benchmarks	http://www.broofa.com/Tools/JSLitmus/
JSFiddle	prototyping	http://jsfiffle.net
Google compiler	compression / performance	http://code.google.com/closure/compiler/

RESOURCES

BOOKS AND SITES OF INTEREST

- Web Sites of interest:

JavaScript™ Developer Center	http://developer.yahoo.com/javascript/
The World's Most Misunderstood Language	http://javascript.crockford.com/javascript.html
Behavioral Separation	http://www.alistapart.com/articles/behavioralseparation
JS Libs Deconstructed	http://www.keyframesandcode.com/resources/javascript/deconstructed/
Learning JQuery	http://www.learningjquery.com/
JQuery Plugins	http://plugins.jquery.com/