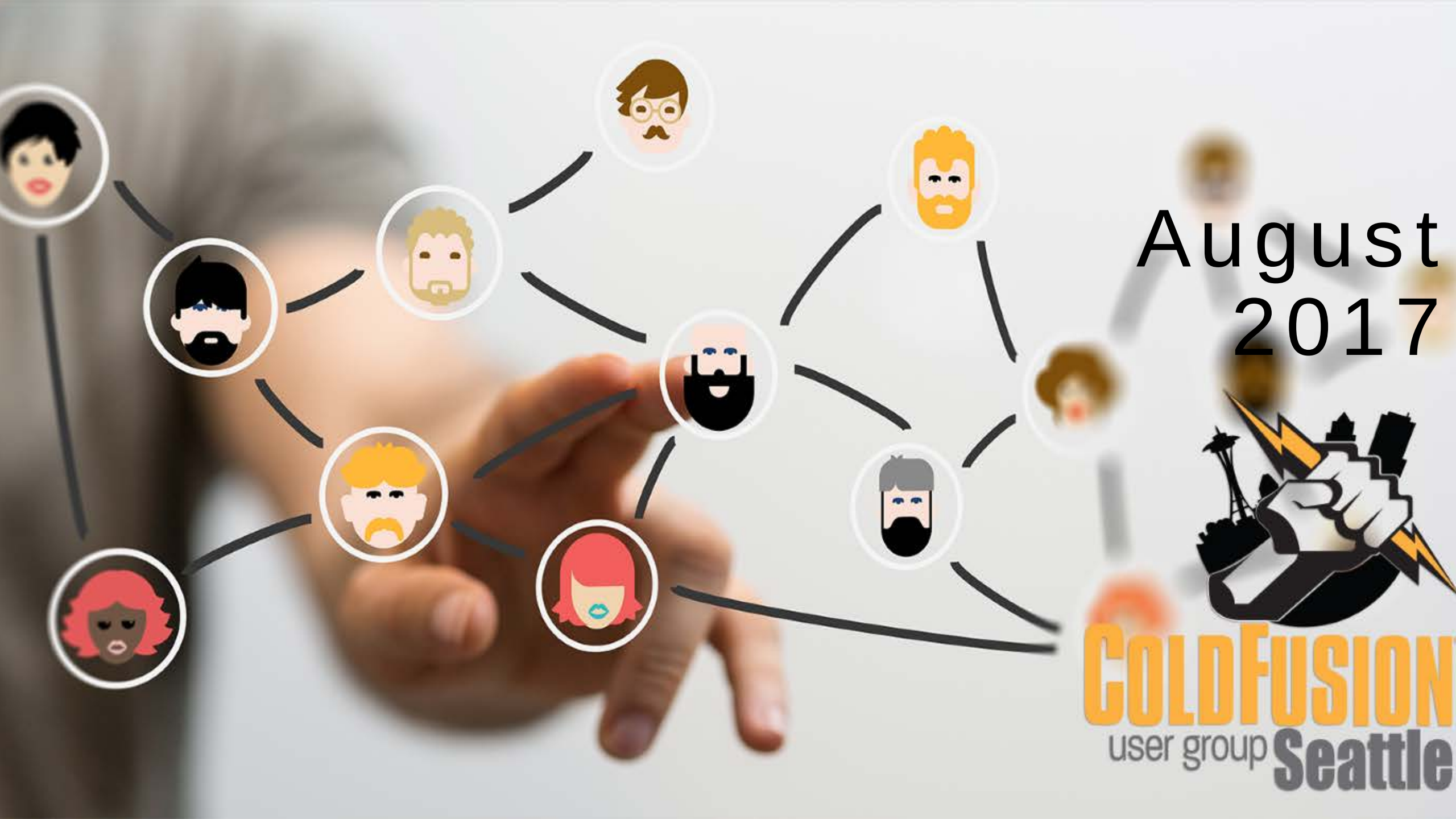

August
2017
ColdFusion
user group Seattle

Agenda

- Welcome to the Seattle ColdFusion User Group
- Introductions
- Goals
- Adobe ColdFusion Developer Week
- ColdFusion Frameworks
- ColdFusion Resources & Upcoming Events
- Social Media/On-line Resources
- Web Site Coming Soon
- Volunteer Opportunities
- Next Month’s Meeting

Ajax, Cold Fusion & Google Maps

- Presenter (who is this guy?)
  - Jim Schell
    - Principal of My Fire Rules
      - aka the Rules Evangelist
    - Fire Fighter for City of Tukwila
    - Self taught no formal training

Development Environment

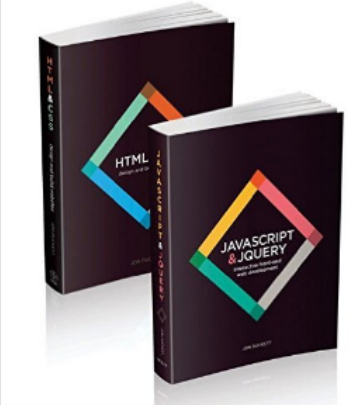
- Eclipse 4.5.2 (mars version 2) with the following libraries
  - Cf 2016
  - Bootstrap 3.3.7
  - Jquery 3.2.1
  - Jqwidgets 5.1
  - Googlemaps api 3.0

Books Used

Web Design with HTML, CSS, JavaScript and jQuery Set 1st Edition

by Jon Duckett (Author)

388 customer reviews



ISBN-13: 978-1118907443
ISBN-10: 1118907442
Why is ISBN important?

Sell yours for a Gift Card
We'll buy it for **$6.97**
Learn More
Trade in now

Have one to sell? Sell on Amazon

Hardcover	Paperback	Other Sellers
$41.97 - $47.62	$32.59 - $35.48	from $26.50

Buy used $32.59

Buy new **$35.48**

In Stock.
Ships from and sold by Amazon.com. Gift-wrap available.

Want it tomorrow, Sept. 7? Order within **15 hrs 59 mins** and choose **One-Day Shipping** at checkout.
Details

List Price: ~~$58.00~~ Save: $22.52 (39%)
41 New from $31.92

prime
FREE Shipping—or get **FREE Two-Day Shipping** with Amazon Prime

Qty: 1

Add to Cart

Turn on 1-Click ordering

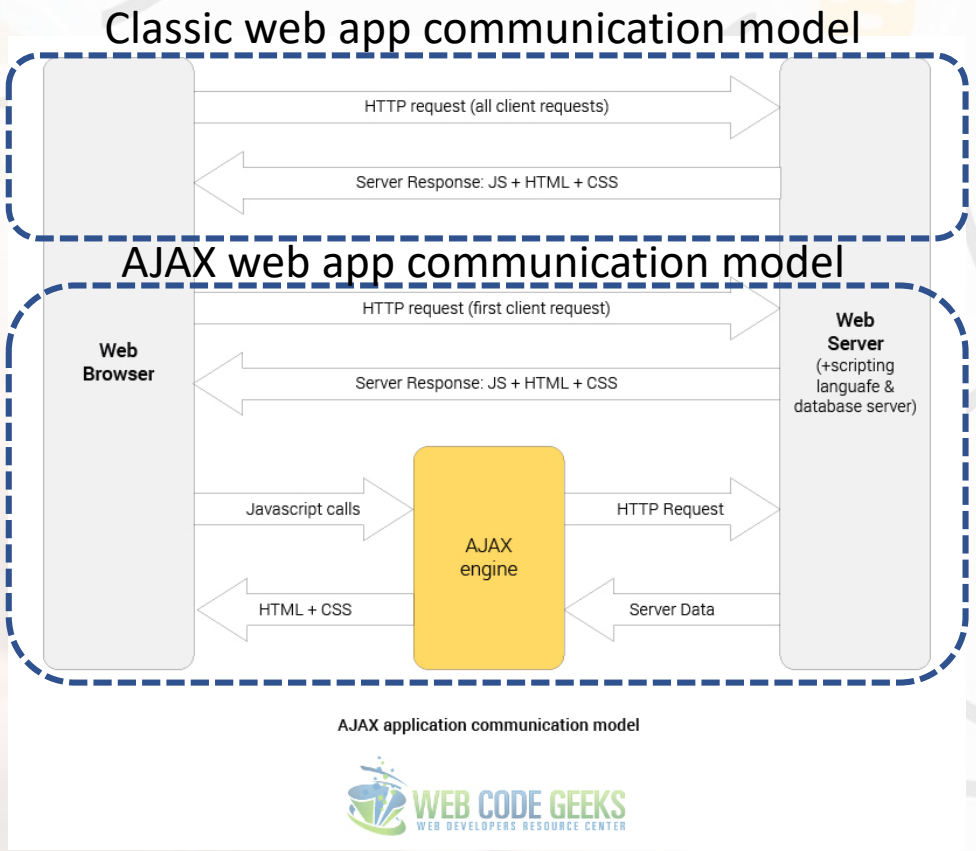
Ship to:
seattle, 98101

More Buying Choices
63 used & new from $26.50
41 New from $31.92 | 22 Used from $26.50
See All Buying Options

What is AJAX....

- Ajax allows content on Web pages to update immediately when a user performs an action...
  http://searchwindevelopment.techtarget.com/definition/Ajax

AJAX Communication...



- https://www.webcodegeeks.com/javascript/jquery/jquery-ajax-example/

- Re: https://www.slideshare.net/shadedecho/extending-ajax-events-for-all-mankind

JQuery Ajax Events

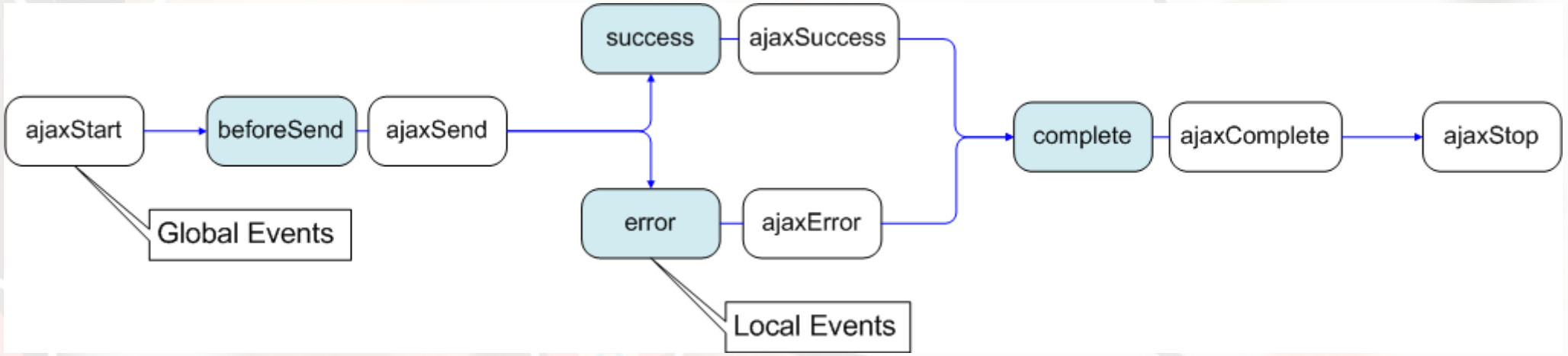
- **ajaxStart** (Global Event)
  This event is triggered if an Ajax request is started and no other Ajax requests are currently running.
  - **beforeSend** (Local Event)
    This event, which is triggered before an Ajax request is started, allows you to modify the XMLHttpRequest object (setting additional headers, if need be.)
  - **ajaxSend** (Global Event)
    This global event is also triggered before the request is run.
  - **success** (Local Event)
    This event is only called if the request was successful (no errors from the server, no errors with the data).
  - **ajaxSuccess** (Global Event)
    This event is also only called if the request was successful.

JQuery Ajax Events part 2

- **error** (Local Event)
  This event is only called if an error occurred with the request (you can never have both an error and a success callback with a request).
- **ajaxError** (Global Event)
  This global event behaves the same as the local error event.
- **complete** (Local Event)
  This event is called regardless of if the request was successful, or not. You will always receive a complete callback, even for synchronous requests.
- **ajaxComplete** (Global Event)
  This event behaves the same as the complete event and will be triggered every time an Ajax request finishes.

- **ajaxStop** (Global Event)
  This global event is triggered if there are no more Ajax requests being processed
- Re: http://api.jquery.com/Ajax_Events/

AJAX Events Explained...



- Re: https://stackoverflow.com/questions/1021062/use-success-or-complete-in-ajax-call

AJAX Function Sample

```
1.      var loadLB = function () {
2.       $.ajax({
3.          type: "post",
4.          url: "assets/cfc/inspRotation.cfc",
5.          dataType: "json",
6.          data:{
7.               method: "getInspDates",
8.               SS: $("#inspAssignment").val()
9.          },
10.         error: function (json, textStatus, errorThrown) {
11.              alert(' Error :' + errorThrown);
12.         },
13.         success: function (data){
14.         // prepare the data
15.         var lbData=data;
16.         var sourceLB =
17.         {
18.           localdata: lbData,
19.           datatype: "json",
20.           dataFields: [
21.             { name: 'ASG_MONTHYEAR', type: "string" },
22.             { name: 'INSPDATE', type: "string" }
23.           ],
24.           id:'ASG_MONTHYEAR'
25.         };
26.         // create data adapter
27.         var dataAdapterLB = new $.jqx.dataAdapter(sourceLB);
28.         // create list box
29.         $("#inspDateLB").jqxListBox({
30.         source: dataAdapterLB,
31.         theme: 'darkblue',
32.         displayMember: "INSPDATE",
33.         valueMember: "ASG_MONTHYEAR",
34.         width: 200,
35.         height: 125
36.         });
37.     }
38.   });
39. }
```

AJAX Function Sample

```
var getInspections = function () {
   $.ajax({
      type: "post",
      url: "assets/cfc/inspRotation.cfc",
      dataType: "json",
      data: {
         method: "getInspList",
         SS: $("#inspAssignment").val(),
         ASG: $("#inspDateLB").val()
      },
      error: function (json,
           textStatus, errorThrown) {
           alert(' Error :' + errorThrown);
      },
      success: function (data){
      // begin google map code
      // default google map options
      var myOptions = {
        center: new google.maps.
          LatLng(47.455863, -122.254503),
        zoom: 14,
        mapTypeId:
          google.maps.MapTypeId.ROADMAP
      };
      map = new google.maps.Map(document.
        getElementById("map"), myOptions);
      var ltlng = [];
      var latlngbounds = new
        google.maps.LatLngBounds();
if (data){
for (var i = 0, length = data.length; i <
  length; i++) {
  var jsn = data[i],
    latLng = new google.maps.LatLng(
      jsn.LATITUDE, jsn.LONGTITUDE);
  latlngbounds.extend(latLng);

// Creating a marker and put it on the map
  var marker = new google.maps.Marker({
    position: latLng,
    map: map,
    title: jsn.NAME+': '+jsn.LOCATION
  });
}};
// adjust google map bounds to fit all markers
          map.fitBounds(latlngbounds);
      // end google map code

          // prepare the data
          var latData=data;
          var sourceLatGrid =
          {
          localdata: latData,
              datatype: "json",
              dataFields: [
                  { name: 'OCCID',
                    type: "string" },
                  { name: 'OCCUPANCY',
                    type: "string" }
              ],
              id:'OCCID'
          };
    // create data adapter
    var latDataAdapterGrid = new
    $.jqx.dataAdapter(sourceLatGrid);
    // create data grid
    $("#latGrid").jqxGrid({
        source: latDataAdapterGrid,
        theme: 'darkblue',
        width: 350,
        height: 350,
        columnsresize: true,
        columns: [
            { text: 'Occid',
              datafield: 'OCCID',
              width: 50 },
            { text:'Name/Address',
              datafield: 'OCCUPANCY',
              width: 350 }
        ]
    });
    // get data without latitude value
    getNoLat();
    }
});
}
```

Cold Fusion CFC Anatomy 101

- CFC Layout
  - Header comments
  - Component
  - Functions
    - Use a consistent formatting of your functions so that you can debug easier

CFC Anatomy

- Header Comments

```
<!---
<Date>07/30/2017</Date>
<WhoCreated>JS</WhoCreated>
<FileName>inspection.cfc</FileName>
<Version>1.0</Version>
<RevisionDate>07/30/2017</RevisionDate>
<Purpose>get look up / mapping data for the inspection rotation</Purpose>
<Comments>
    Requires the following sql stored procedures:
        None
    Requires the following sql user functions:
        None
    Functions:
        1) getAssignments() -- gets a distinct list of inspection assignements comprised of shift and station
        2) getInspDates(SS) -- gets a distinct list of inspection dates related to the assignment selected
        3) getInspList(SS,ASG) -- gets a distinct list of inspections from the occ_schedule table based on assignment and date selected
        4) getNoLatInspList(SS,ASG) -- gets a distinct list of inspections without a latitude value from the occ_schedule table based on assignment and
</Comments>
 --->
```

- Component
  - only one per cfc but can have multiple functions within

CFC Anatomy In-line SQL

- Sample cffunction

```
<cffunction name="getInspDates" displayname="Get Inspection Dates"
    description="gets a distinct list of inspection dates related to the assignment selected"
    access="remote" output="true" returntype="any" returnformat="json" >

    <!--- Define arguments --->
    <cfargument name="SS" required="true" type="string" />

    <!--- Define variables --->
    <cfset var q="">

    <!--- Get data --->
    <cfquery name="q" datasource="inspList" >
        select distinct CONVERT(VARCHAR(10),asg_monthyear, 101) as inspDate, asg_monthyear,year(asg_monthyear) as YOI
        from occ_schedule
        where asstocode = <cfqueryparam cfsqltype="cf_sql_char" value="#arguments.SS#" />
        order by YOI ,inspDate
    </cfquery>

    <!--- And return it --->
    <cfset q = serializeJSON(q,"struct")>
    <cfreturn q>

</cffunction>
```

CFC Anatomy Stored Proc

- Stored procedures….

```
<cffunction name="getData" displayname="Get Occupancy Data"
    access="remote" output="true" returntype="Any" returnformat="JSON" >

    <!--- Define arguments --->
    <cfargument name="dbType" required="true" type="string" />
    <cfargument name="searchParam" required="false" type="string" />
    <cfargument name="qry" required="true" type="numeric" />

    <!--- Define variables --->
    <cfset var q="">

    <!--- Get data --->
    <cfstoredproc procedure="usp_MyFireTouch_Get_Occupancy_Public_Records" datasource="inspList">
        <cfprocresult name="q" />
        <cfprocparam type="in" cfsqltype="CF_SQL_CHAR" value="#arguments.dbtype#" maxlength="1"/>
        <cfprocparam type="in" cfsqltype="CF_SQL_CHAR" value="#arguments.searchParam#"/>
        <cfprocparam type="in" cfsqltype="CF_SQL_INTEGER" value="#arguments.qry#" maxlength="1"/>
    </cfstoredproc>

    <!--- And return it --->
    <cfset q = serializeJSON(q,"struct")>
    <cfreturn q>
</cffunction>
```

SQL Injection Safety Net

- In Line SQL
  - cfqueryparam
- Stored Procedures
  - cfprocparam
- USE THEM THEY WILL HELP PREVENT SQL INJECTION ATTACKS!!!!!!!!!!!
- Consider not allowing free form text search fields, if you do then test the text first before you send it against your live tables

Stored Procedure Anatomy

- Header comments
- Structured and clean

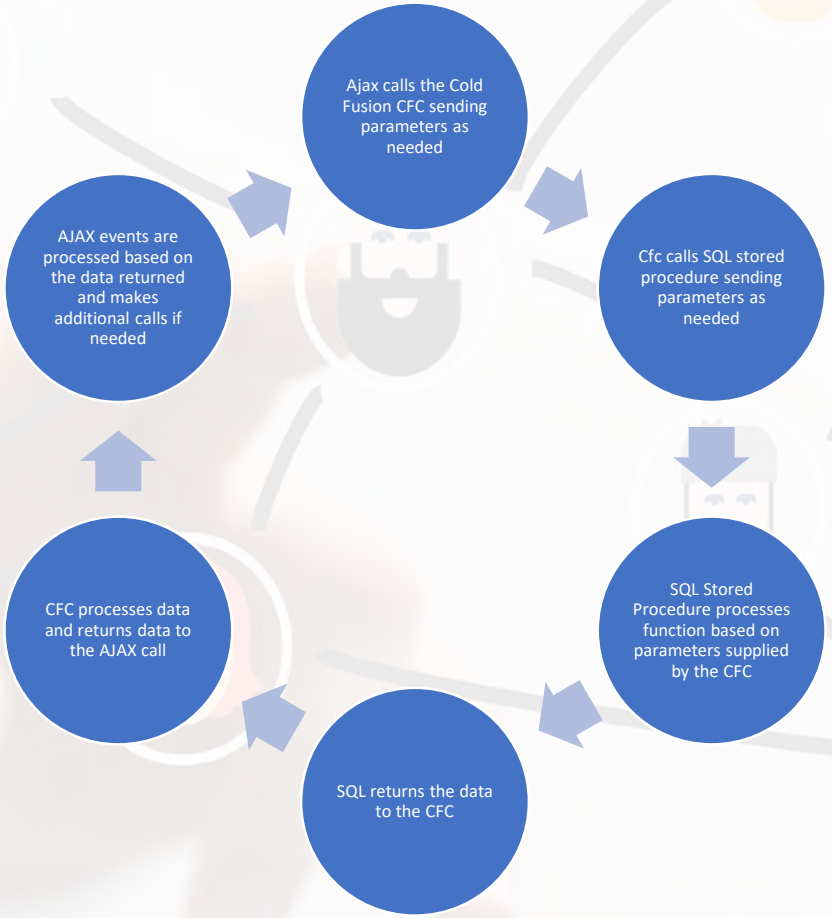
```
/*
<Date>07/15/17</Date>
<WhoUpdated>JS</WhoUpdated>
<FileName>usp_MyFireTouch_Get_Occupancy_Public_Records.sql</FileName>
<CreatesObject>usp_MyFireTouch_Get_Occupancy_Public_Records</CreatesObject>
<Purpose>Retrieve Records matching the parameters passed</Purpose>
<Parameters>@testing, @dbtype, @searchParam, @qry</Parameters>
<Comments>Return Incident Data for records search application
  testing: @testing char(1) 0 = production (default value)
           @testing char(1) 1 = testing

  database: @dbType char(1) L = Verify user agains login table
            @dbType char(1) V = Verify search param against sql reserverd words table
            @dbType char(1) Z = Zoll (default value is Z for Zoll, currently only support database)

  parameter: @searchParam char(40) wildcard value to search for

  query: @qry char(1)   0 = Full Street Listing
         @qry char(1)   1 = Occs By Full Street
         @qry char(1)   2 = Occs By Street Address
         @qry char(1)   3 = Single Occ Record
         @qry char(1)   4 = Occs By Parcel Id
         @qry char(1)   5 = Occs By Activity Comment
         @qry char(1)   6 = Occs By Attachment Comment
</Comments>
<Object>StoredProcedure [dbo].usp_MyFireTouch_Get_Occupancy_Public_Records.sql</Object>
<Dependencies>User Function (NONE)</Dependencies>
<Testing>
  EXEC usp_MyFireTouch_Get_Occupancy_Public_Records 'L','','' -- login check
  EXEC usp_MyFireTouch_Get_Occupancy_Public_Records 'V','EXEC','' -- sql injection attack
  EXEC usp_MyFireTouch_Get_Occupancy_Public_Records 'Z','',0 -- full street listing
  EXEC usp_MyFireTouch_Get_Occupancy_Public_Records 'Z','65 Ave S',1 -- occs by full street
  EXEC usp_MyFireTouch_Get_Occupancy_Public_Records 'Z','17300',2 -- occs by street address
  EXEC usp_MyFireTouch_Get_Occupancy_Public_Records 'Z','_E81AFB29BBMFR',3 -- single composite occ record
  EXEC usp_MyFireTouch_Get_Occupancy_Public_Records 'Z','367',4 -- occs by parcel id
  EXEC usp_MyFireTouch_Get_Occupancy_Public_Records 'Z','adt',5 -- occs by activity comment
  EXEC usp_MyFireTouch_Get_Occupancy_Public_Records 'Z','file',6 -- occs by attachment comment
</Testing>
*/
```

SQL Stored Procedures...

- Code snippet of a parameter based SQL stored procedure:

```
IF @dbType = 'V' -- Verify SQL Injection Reserved Words
  BEGIN
    DECLARE @pos INT;
    SET @pos = 0;
    SELECT @pos += CHARINDEX (LOWER(reservedWord) , LOWER(@searchParam)) FROM SQLReservedWords
    SELECT CASE WHEN @pos>0 THEN 1 ELSE 0 END AS RWORDVALUE, CASE WHEN @pos>0 THEN 'BAD' ELSE 'GOOD' END AS RWORDTEXT
  END
```

AJAX to SQL and back...



DEMO TIME!!!!

- Sample application for Tukwila Fire Dept.
  - Shows monthly inspection locations
  - Incorporates
    - HTML
    - CFML
    - CSS
    - Javascript
    - AJAX / JQUERY
    - JQWIDGETS
    - Bootstrap
    - Google Map API