Application Express

Dynamic Duo

Josh Millinger

Niantic Systems

June 7, 2011

info@nianticsystems.com

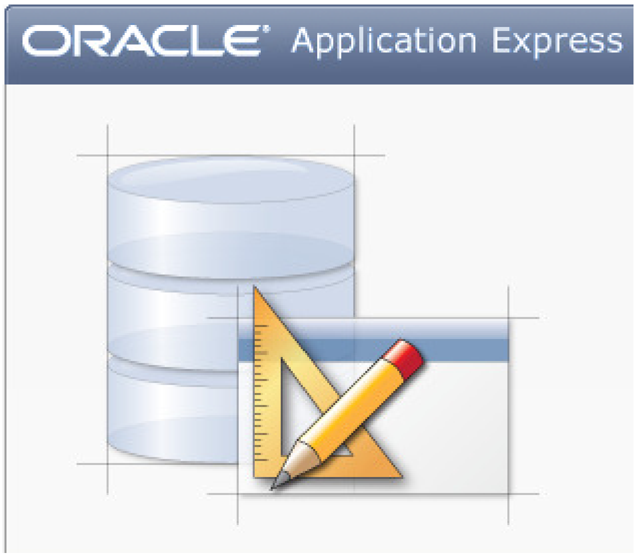
Speaker Qualifications

- Josh Millinger, President, Niantic Systems, LLC
- CS degrees from UW-Madison, Johns Hopkins
- Former Oracle Sales Consultant and Founder of the Oracle Partner Technology Center
- 15+ Years of Oracle Web Development Experience
- Have Been Developing with and Teaching ApEx Since Well Before It Was Even Released as a Product!
- Started with Excel Migration as first project
- Presenter at NYOUG, IOUG, ODTUG, Oracle OpenWorld

- Oracle Consulting with a Focus on Application Express
- Application Express Training
- Oracle Forms/Reports
- Discoverer
- Mentoring
- Forms/Reports to Apex Migration
- Customers in the Federal, Commercial, Healthcare, Higher Education, Financial, and Construction verticals

Agenda

- Define Dynamic SQL
- When to use Dynamic SQL
- Using bind variables to secure SQL
- Using with Interactive Reports



Agenda

- Javascript Overview
- Dynamic Action Overview
- Demonstration

What is Dynamic SQL?

What is Dynamic SQL?

Dynamic SQL is a programming technique that enables you to build SQL statements dynamically at runtime. You can create more general purpose, **flexible** applications by using dynamic SQL because the full text of a SQL statement may be **unknown at compilation**. For example, dynamic SQL lets you create a procedure that operates on a table whose name is not known until runtime.

You can use dynamic SQL to create applications that execute dynamic queries, whose full text is not known until runtime. Many types of applications need to use dynamic queries, including:

•*Applications that allow users to input or choose query search or sorting criteria at runtime*
•*Applications that allow users to input or choose optimizer hints at run time*
•*Applications that query a database where the data definitions of tables are constantly changing*
•*Applications that query a database where new tables are created often*

Reference: Oracle Application Developers Guide - Fundamentals

Static vs. Dynamic

Static:

```
select patient, service_date, exam from exams
```

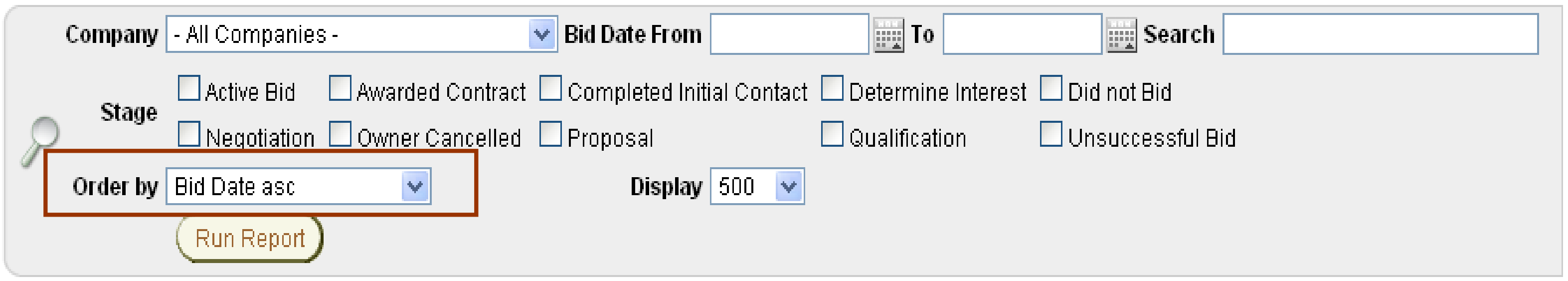
In this case the SQL statement is well known as design time

Dynamic:

```
declare
 l_sql varchar2(1000);
begin
  l_sql := 'select ' || :PX_COLNAME||' from exams';
    return l_sql;
end;
```

In this case, the column to select is user defined at runtime

When do I need to use Dynamic SQL


Company - All Companies - Bid Date From To Search

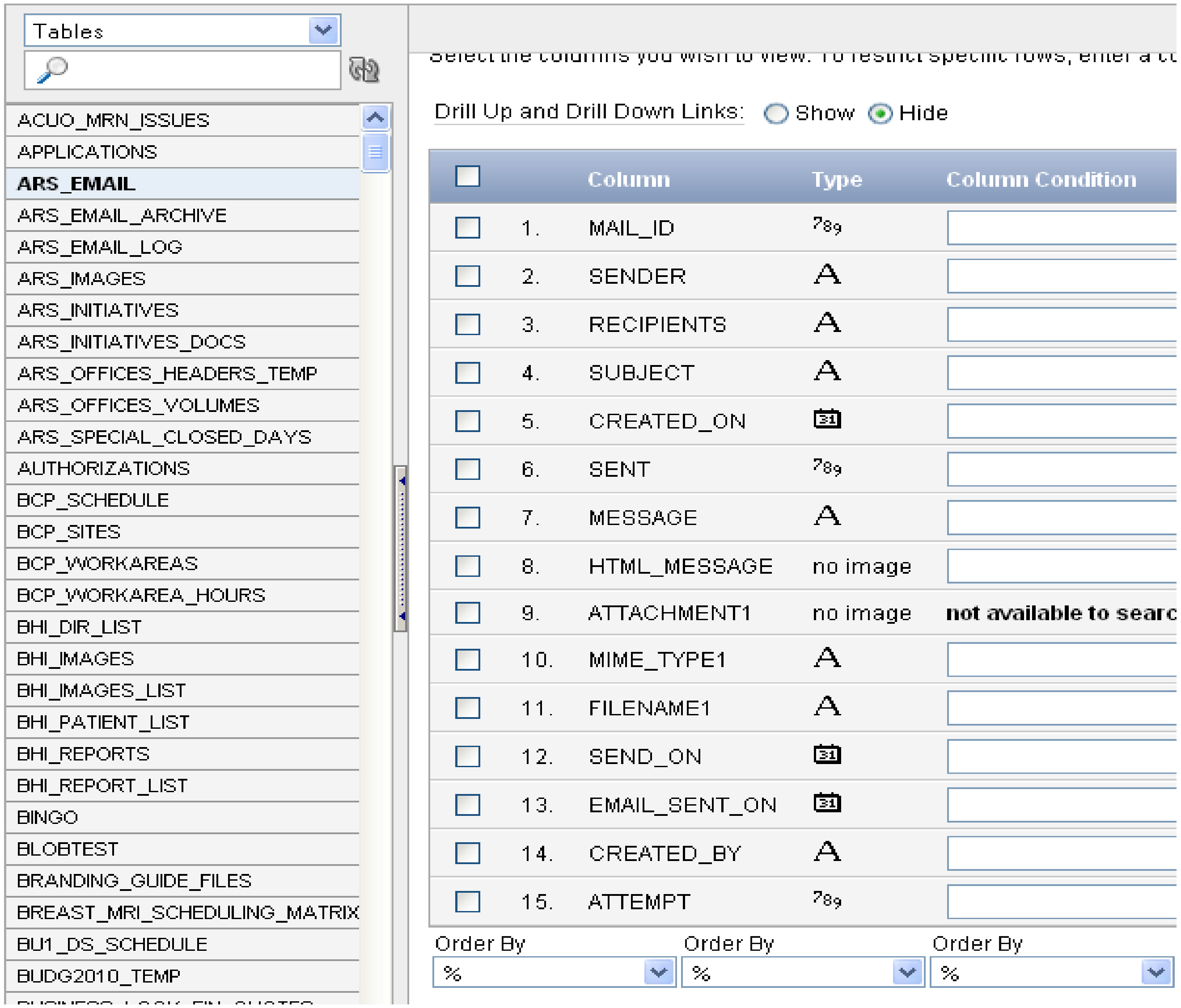
Stage ☐ Active Bid ☐ Awarded Contract ☐ Completed Initial Contact ☐ Determine Interest ☐ Did not Bid ☐ Negotiation ☐ Owner Cancelled ☐ Proposal ☐ Qualification ☐ Unsuccessful Bid

Order by Bid Date asc

Display 500

```
declare
 l_sql varchar2(10000);
begin
  l_sql := ‘select col1, col2 from jobs where 1=1 ‘;
    ..filters here…
   l_sql := l_sql || ‘ order by ‘|| v(‘P1_ORDERBY’) ;
  return l_sql;
end;
```

When do I need to use Dynamic SQL - Example



Tables

Select the columns you wish to view. To restrict specific rows, enter a c

Drill Up and Drill Down Links: ( ) Show [✓ Hide]

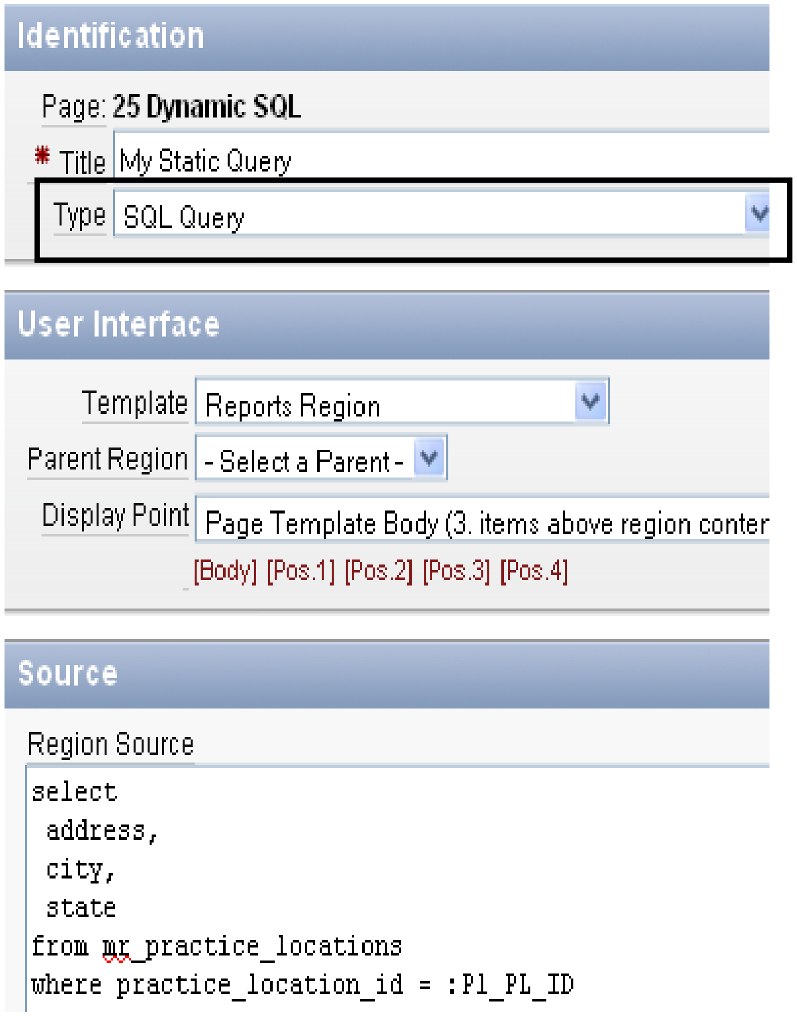
ACUO_MRN_ISSUES
APPLICATIONS
ARS_EMAIL
ARS_EMAIL_ARCHIVE
ARS_EMAIL_LOG
ARS_IMAGES
ARS_INITIATIVES
ARS_INITIATIVES_DOCS
ARS_OFFICES_HEADERS_TEMP
ARS_OFFICES_VOLUMES
ARS_SPECIAL_CLOSED_DAYS
AUTHORIZATIONS
BCP_SCHEDULE
BCP_SITES
BCP_WORKAREAS
BCP_WORKAREA_HOURS
BHI_DIR_LIST
BHI_IMAGES
BHI_IMAGES_LIST
BHI_PATIENT_LIST
BHI_REPORTS
BHI_REPORT_LIST
BINGO
BLOBTEST
BRANDING_GUIDE_FILES
BREAST_MRI_SCHEDULING_MATRIX
BU1_DS_SCHEDULE
BUDG2010_TEMP

☐		Column	Type	Column Condition
☐	1.	MAIL_ID	789	
☐	2.	SENDER	A	
☐	3.	RECIPIENTS	A	
☐	4.	SUBJECT	A	
☐	5.	CREATED_ON	31	
☐	6.	SENT	789	
☐	7.	MESSAGE	A	
☐	8.	HTML_MESSAGE	no image	
☐	9.	ATTACHMENT1	no image	**not available to searc**
☐	10.	MIME_TYPE1	A	
☐	11.	FILENAME1	A	
☐	12.	SEND_ON	31	
☐	13.	EMAIL_SENT_ON	31	
☐	14.	CREATED_BY	A	
☐	15.	ATTEMPT	789	

Order By % | Order By % | Order By %

info@nianticsystems.com

Region Type Definition



Identification

Page: 25 Dynamic SQL

* Title	My Static Query
Type	SQL Query

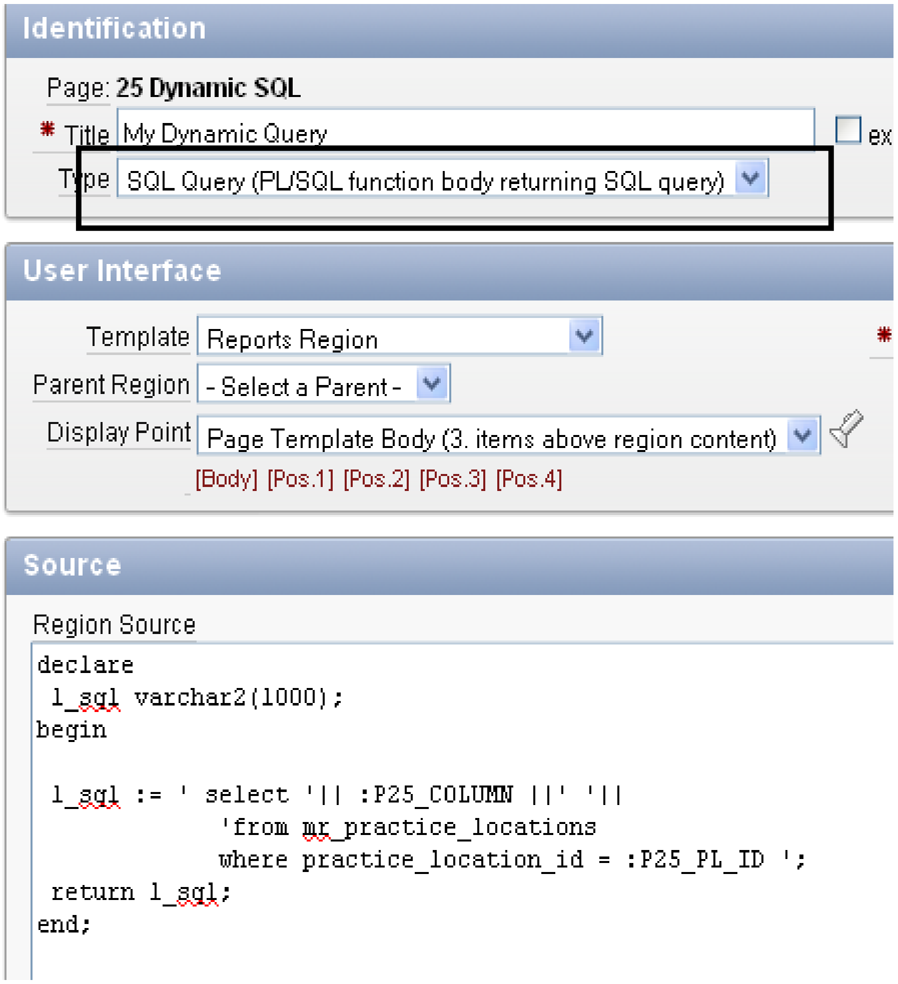
User Interface

Template	Reports Region
Parent Region	- Select a Parent -
Display Point	Page Template Body (3. items above region conter
	[Body] [Pos.1] [Pos.2] [Pos.3] [Pos.4]

Source

Region Source

```
select
 address,
 city,
 state
from mr_practice_locations
where practice_location_id = :P1_PL_ID
```



Identification

Page: 25 Dynamic SQL

* Title	My Dynamic Query
Type	SQL Query (PL/SQL function body returning SQL query)

User Interface

Template	Reports Region
Parent Region	- Select a Parent -
Display Point	Page Template Body (3. items above region content)
	[Body] [Pos.1] [Pos.2] [Pos.3] [Pos.4]

Source

Region Source

```
declare
 l_sql varchar2(1000);
begin

 l_sql := ' select '|| :P25_COLUMN ||' '||
             'from mr_practice_locations
             where practice_location_id = :P25_PL_ID ';
 return l_sql;
end;
```

info@nianticsystems.com

Where do I create Dynamic SQL?

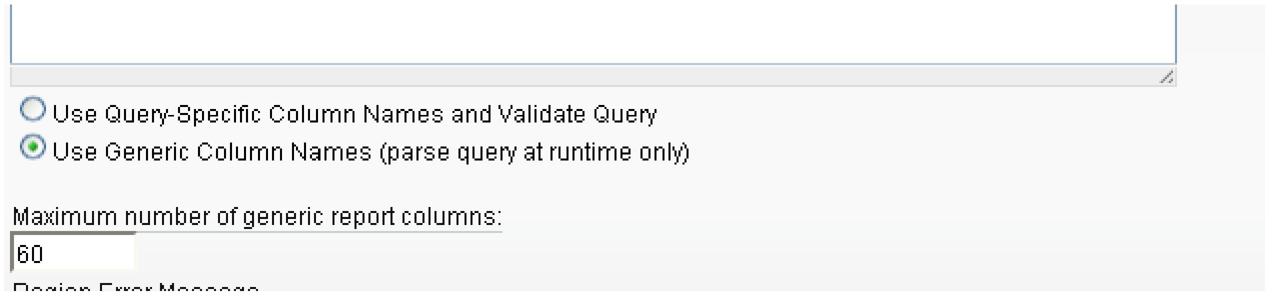
- When creating a complex function to generate dynamic SQL, it is best to place function in database either standalone or in package
  - Easier to edit
  - Creates ability to reuse it on other pages
- Call function through normal “return” syntax

```
return my_pkg.get_page25_query(:P25_ITEMNAME);
```

Apex Parsing

- If the builder cannot parse the sql statement you will have to select:

Use Generic Column Names (parse query at runtime only)



Use Query-Specific Column Names and Validate Query [✓ Use Generic Column Names (parse query at runtime only)]

Maximum number of generic report columns:
60

Best Practice: For performance and productivity purposes, change the maximum number of colums to something equal or slightly higher than maximum number of possible columns

Warning: You might get error if column number greater than the number of columns in query is higher on Report Attributes Page

Dynamic SQL can also be used in:

- Charts
- List of Values

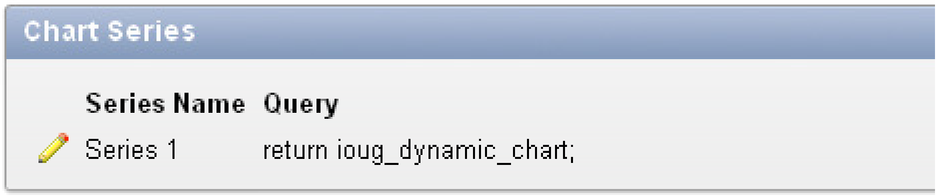
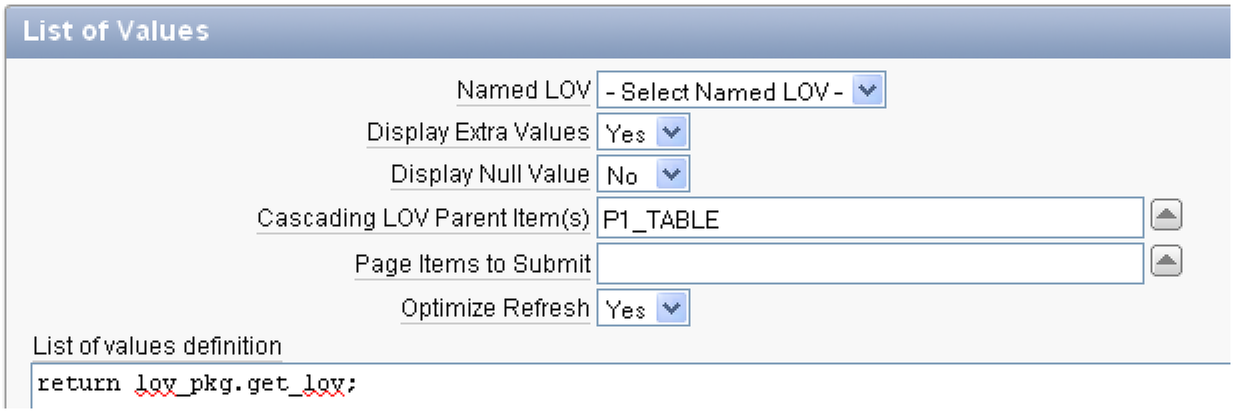


Chart Series

	Series Name	Query
	Series 1	return ioug_dynamic_chart;



List of Values

Named LOV	- Select Named LOV -
Display Extra Values	Yes
Display Null Value	No
Cascading LOV Parent Item(s)	P1_TABLE
Page Items to Submit	
Optimize Refresh	Yes

List of values definition

```
return lov_pkg.get_lov;
```

How to debug

- When using Dynamic SQL it is important to be able to see what query is being generated
- Use “DEBUG” to help you determine the query

```
declare
 l_sql varchar2(1000);
begin
  l_sql := ‘select ‘ || :PX_COLNAME||’ from exams’;
  apex_application.debug(‘My query is : ‘||l_sql);
  return l_sql;
end;
```

Using bind variables in Dynamic SQL

- When generating in database still use the :BINDVAR syntax
  - This will allow the optimizer to reuse execution plan

```
declare
  l_sql varchar2(1000);
 l_col  varchar2(100);
begin
   if v(‘P1_TEST’)  = 1 then  l_col := ‘mycol1’ ; else l_col := ‘mycol2’; end if;
  l_sql := ‘select ‘ || v(‘PX_COLNAME’)||’ from exams where id = :P1_ID ’;
  apex_application.debug(‘My query is : ‘||l_sql);
  return l_sql;
end;
```

Using bind variables in Dynamic SQL

- Prevent SQL Injection Attacks
- Take a block of code that generates a query

```
declare
    q varchar2(4000);
begin
    q := ‘select  *
           from tasks
          where assigned = :APP_USER ';

    if :P1_SEARCH is not NULL THEN
        q := q || ' AND category
          = ' ||:P1_SEARCH ;
    end if;
    return q;
end;
```

Using bind variables in Dynamic SQL

When a user provides “email” for P1_SEARCH our query will be:

```
select *
  from tasks
 where assigned=:app_user
    and category = ‘email’
```

Using bind variables in Dynamic SQL

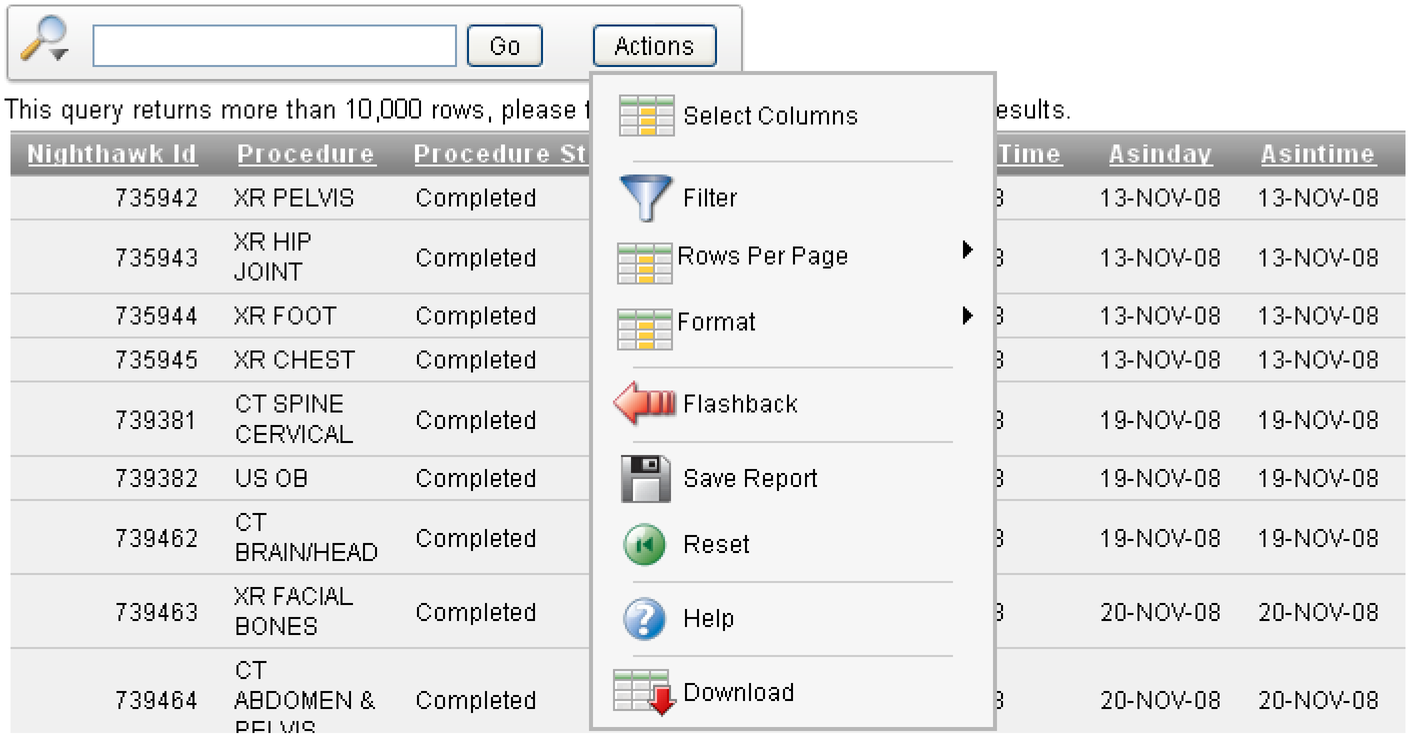
…but when a user provides “email’ or ‘a’=‘a” for P1_SEARCH our query becomes

```
select *
  from tasks
 where ….
   and category = ‘email’ or
    ‘a’ = ‘a’
```

…So ***never*** arbitrarily append user input into your application queries.

Interactive Reports

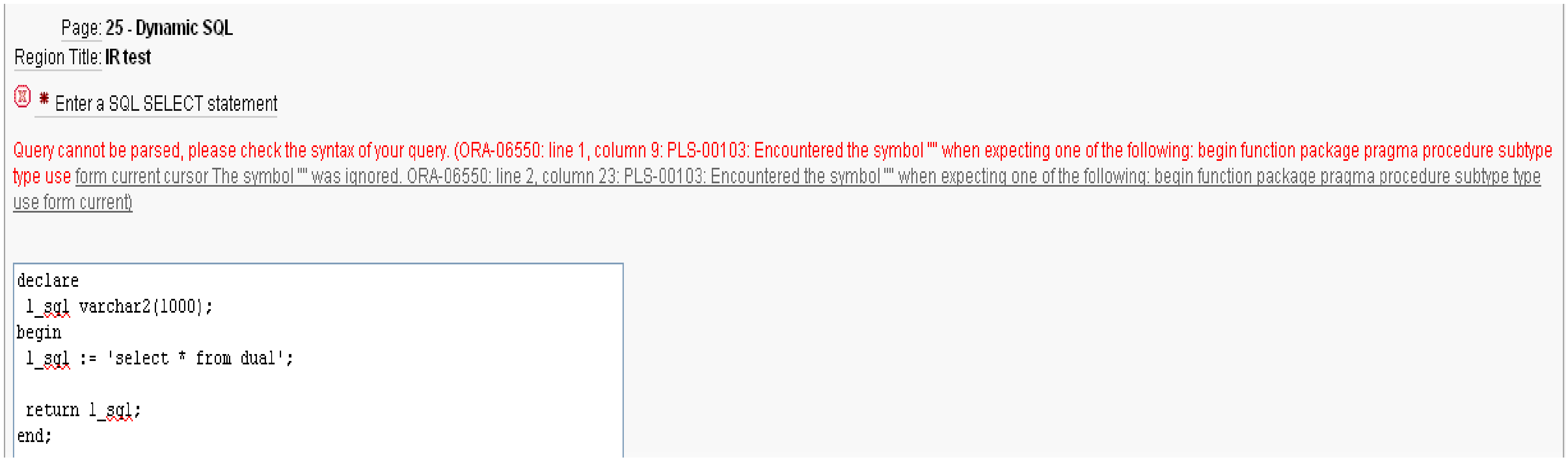
- Interactive Reports were introduced in version 3.0
- They provide
  - End users the ability to customize the data to their liking using controls such as column filters, aggregates, computations, groupings, etc.
- IR's are based off a SQL query
- Limitation is SQL query HAS to be static



This query returns more than 10,000 rows, please f esults.

Nighthawk Id	Procedure	Procedure St	Time	Asinday	Asintime
735942	XR PELVIS	Completed	3	13-NOV-08	13-NOV-08
735943	XR HIP JOINT	Completed	3	13-NOV-08	13-NOV-08
735944	XR FOOT	Completed	3	13-NOV-08	13-NOV-08
735945	XR CHEST	Completed	3	13-NOV-08	13-NOV-08
739381	CT SPINE CERVICAL	Completed	3	19-NOV-08	19-NOV-08
739382	US OB	Completed	3	19-NOV-08	19-NOV-08
739462	CT BRAIN/HEAD	Completed	3	19-NOV-08	19-NOV-08
739463	XR FACIAL BONES	Completed	3	20-NOV-08	20-NOV-08
739464	CT ABDOMEN & PELVIS	Completed	3	20-NOV-08	20-NOV-08

- To overcome this limitation we use *Collections*
- Collections defined:
  - Collections enable you to temporarily capture one or more nonscalar values. You can use collections to store rows and columns currently in session state so they can be accessed, manipulated, or processed during a user's specific session. You can think of a collection as a bucket in which you temporarily store and name rows of information (Apex Documentation)
  - Useful when data is needed across page views as temporary tables won't work



Page: 25 - Dynamic SQL
Region Title: IR test

* Enter a SQL SELECT statement

Query cannot be parsed, please check the syntax of your query. (ORA-06550: line 1, column 9: PLS-00103: Encountered the symbol "" when expecting one of the following: begin function package pragma procedure subtype type use form current cursor The symbol "" was ignored. ORA-06550: line 2, column 23: PLS-00103: Encountered the symbol "" when expecting one of the following: begin function package pragma procedure subtype type use form current)

```
declare
 l_sql varchar2(1000);
begin
 l_sql := 'select * from dual';

 return l_sql;
end;
```

Interactive Reports and Collections

1. Create collection when the page renders

```
declare
 l_sql varchar2(1000);
begin
  if apex_collection.collection_exists(‘P25_ROWS’)
   then
    apex_collection.delete_collection(‘P25_ROWS’);
  end if;

   l_sql :=  my_pkg.get_my_query (:P1_VAR);  -- get the dynamic sql query here

  apex_collection.create_collection_from_query_b(‘P25_ROWS’,l_sql);  -- create the collection
  end;
```

2. Create Interactive Report from Collection

```
 select col1, col2
   from my_table m,
          apex_collections a
where m.id = a.c001
   and a.collection_name = ‘P25_ROWS’
```

Dynamic Sql - Conclusion

- Dynamic SQL
  - Is useful when a query not known at develop time
    - Unknown table
    - Unknown columns
    - Unknowns sorting
  - Can be used with Interactive Reports
    - By using collection or other row collecting mechanism
  - Can be used with Reports, Charts, and LOV's

Apex and Javascript

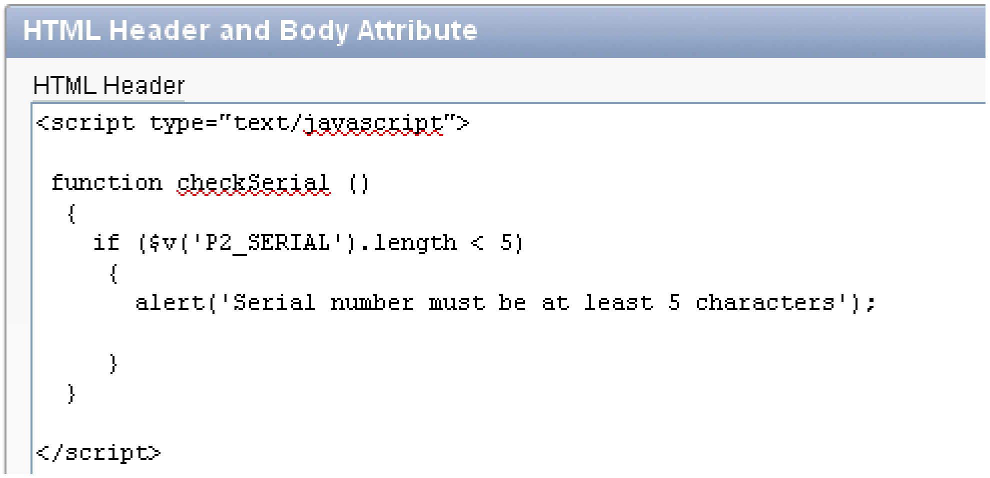
- Apex is reliant on Javascript
  - Object Browser
  - Builder - Drag/Drop, Delete Confirmation
  - apex.submit
  - Hide/Show of relevant fields in Builder
  - Region Selector
- Javascript can also be used by developers
  - Allow for custom interactive actions on page
  - Should not be confused with Java
  - See previous presentation by Niantic for NYOUG

Apex and Javascript

- Developers use Javascript for
  - Validations
  - Computations and Calculations
  - Dynamic Control of the GUI
  - Alerts
  - Confirm Boxes
  - Region Selectors
  - Interactive Reports
  - AJAX

Apex and Javascript

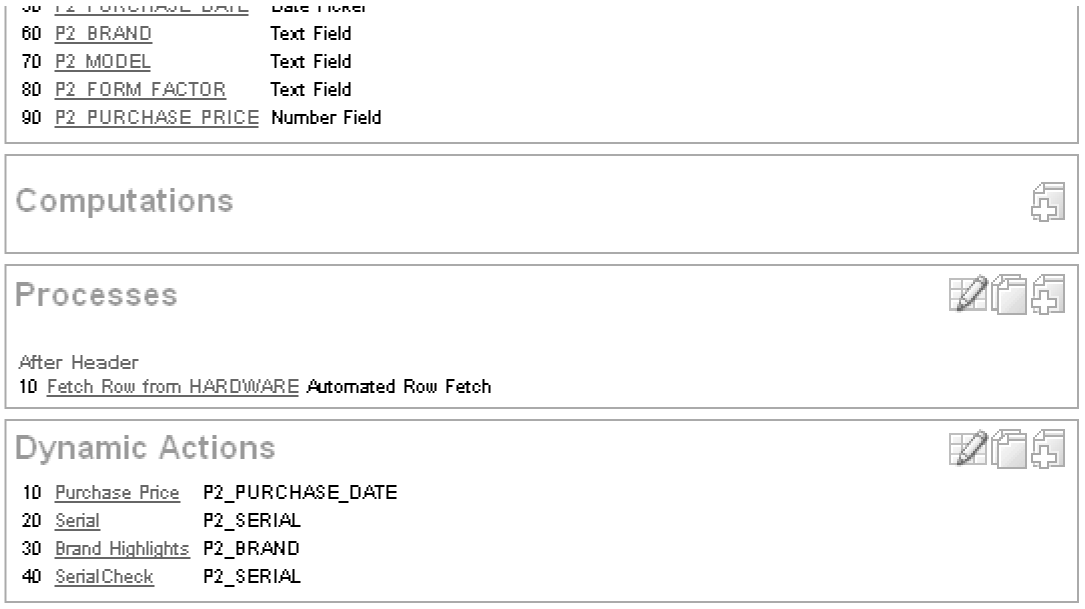
- Before Apex 4.0 Javascript would be either
  - Placed in .js file on filesystem
  - Placed in HTML Header
  - Placed in Page Template
  - Placed on Page Zero
  - Placed in Region on Page where needed
- Would require manual creation of code
- Needed knowledge of how to code Javascript



```
<script type="text/javascript">

 function checkSerial ()
  {
    if ($v('P2_SERIAL').length < 5)
     {
       alert('Serial number must be at least 5 characters');

     }
  }

</script>
```

Dynamic Actions

- Introduced in Apex 4.0
- Allow for declarative creation of Javascript
- Developers no longer need to be JS coders
- Wizard based and Re-entrant
- Created at Page Level



60	P2_BRAND	Text Field
70	P2_MODEL	Text Field
80	P2_FORM_FACTOR	Text Field
90	P2_PURCHASE_PRICE	Number Field

Computations

Processes

After Header

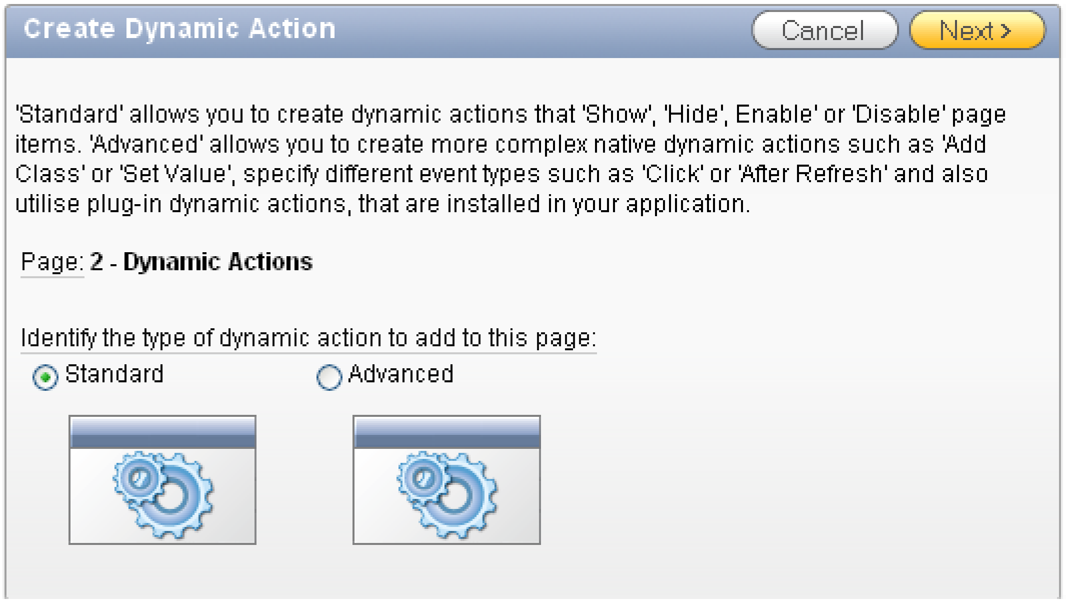
10 Fetch Row from HARDWARE Automated Row Fetch

Dynamic Actions

10	Purchase Price	P2_PURCHASE_DATE
20	Serial	P2_SERIAL
30	Brand Highlights	P2_BRAND
40	SerialCheck	P2_SERIAL

Dynamic Actions

- Two types of Dynamic Actions
    - Standard
    - Advanced



'Standard' allows you to create dynamic actions that 'Show', 'Hide', Enable' or 'Disable' page items. 'Advanced' allows you to create more complex native dynamic actions such as 'Add Class' or 'Set Value', specify different event types such as 'Click' or 'After Refresh' and also utilise plug-in dynamic actions, that are installed in your application.

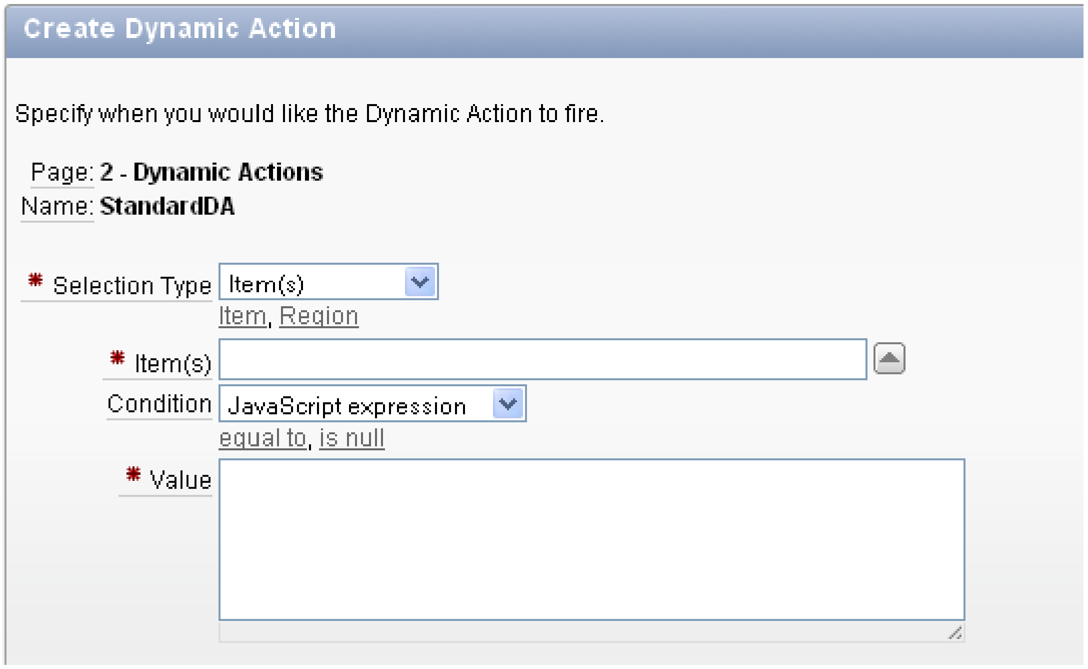
Page: **2 - Dynamic Actions**

Identify the type of dynamic action to add to this page:

[✓ Standard] Advanced

Dynamic Actions - Standard

- Selection Type: Item, Region, jQuery or DOM Object
- Can be conditionally executed



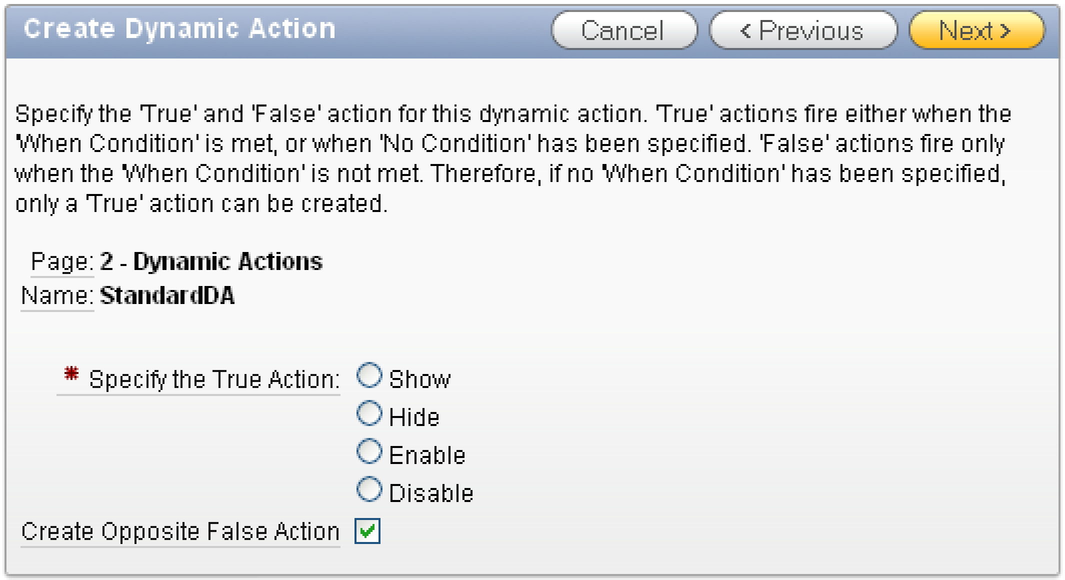
Specify when you would like the Dynamic Action to fire.

Page: **2 - Dynamic Actions**
Name: **StandardDA**

* Selection Type	Item(s)
	Item, Region
* Item(s)	
Condition	JavaScript expression
	equal to, is null
* Value	

Dynamic Actions - Standard

- Action can Hide/Show or Enable/Disable page elements
- Can create opposite False Action
  - If Dynamic Action shows item when the condition is TRUE, then this created DA that hides item when FALSE



Specify the 'True' and 'False' action for this dynamic action. 'True' actions fire either when the 'When Condition' is met, or when 'No Condition' has been specified. 'False' actions fire only when the 'When Condition' is not met. Therefore, if no 'When Condition' has been specified, only a 'True' action can be created.

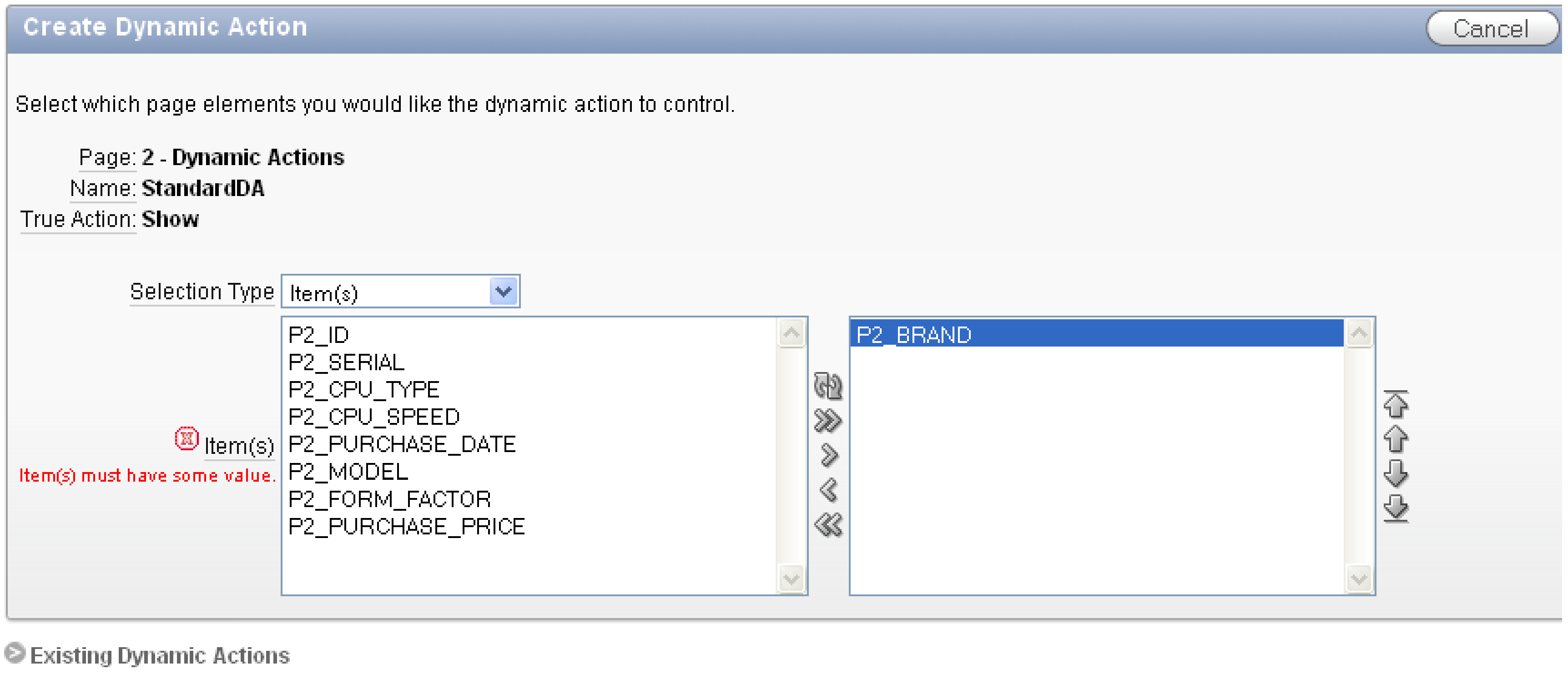
Page: **2 - Dynamic Actions**
Name: **StandardDA**

* Specify the True Action: ○ Show ○ Hide ○ Enable ○ Disable

Create Opposite False Action ✓

Dynamic Actions - Standard

Select what elements are affected by TRUE/FALSE action



Select which page elements you would like the dynamic action to control.

Page: **2 - Dynamic Actions**
Name: **StandardDA**
True Action: **Show**

Selection Type: Item(s)

Item(s)
Item(s) must have some value.

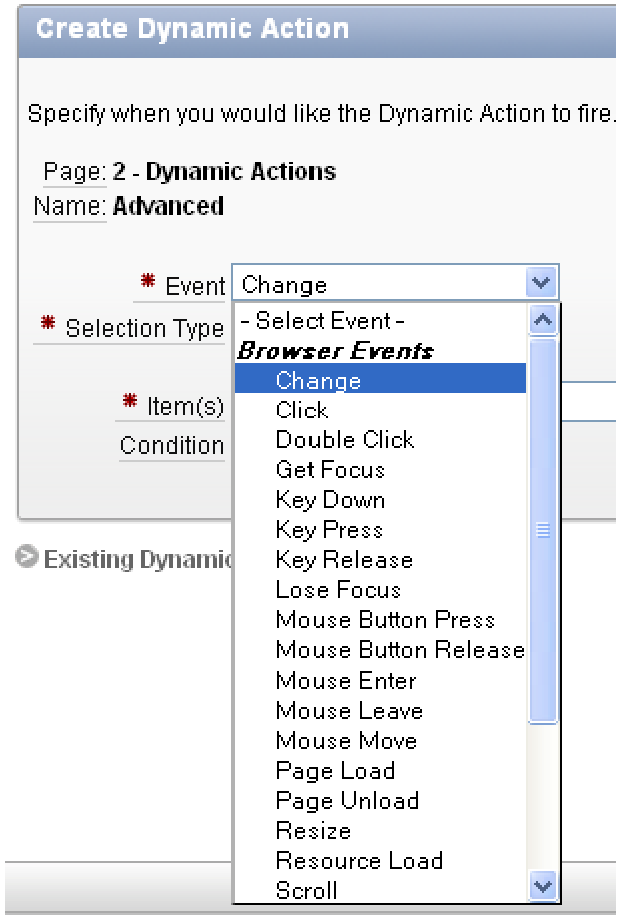
P2_ID
P2_SERIAL
P2_CPU_TYPE
P2_CPU_SPEED
P2_PURCHASE_DATE
P2_MODEL
P2_FORM_FACTOR
P2_PURCHASE_PRICE

P2_BRAND

Dynamic Actions - Advanced

Event Based

- Change of Value
- Losing Focus
- Mouse entering/leaving
- Page load/unload
- Scroll
- Double Click
- Key Up/Down



Create Dynamic Action

Specify when you would like the Dynamic Action to fire.

Page: **2 - Dynamic Actions**
Name: **Advanced**

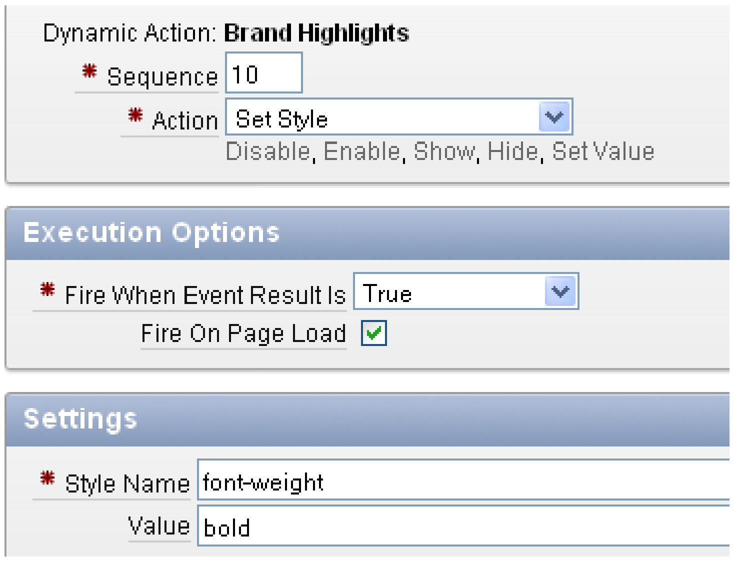
* Event: Change
* Selection Type
* Item(s)
Condition

Existing Dynamic

Dynamic Actions - Advanced

Declare what action to take

- Clear
- Hide/Show
- Set Value
- Execute JS or PL/SQL
- Add remove classes



Dynamic Action: **Brand Highlights**

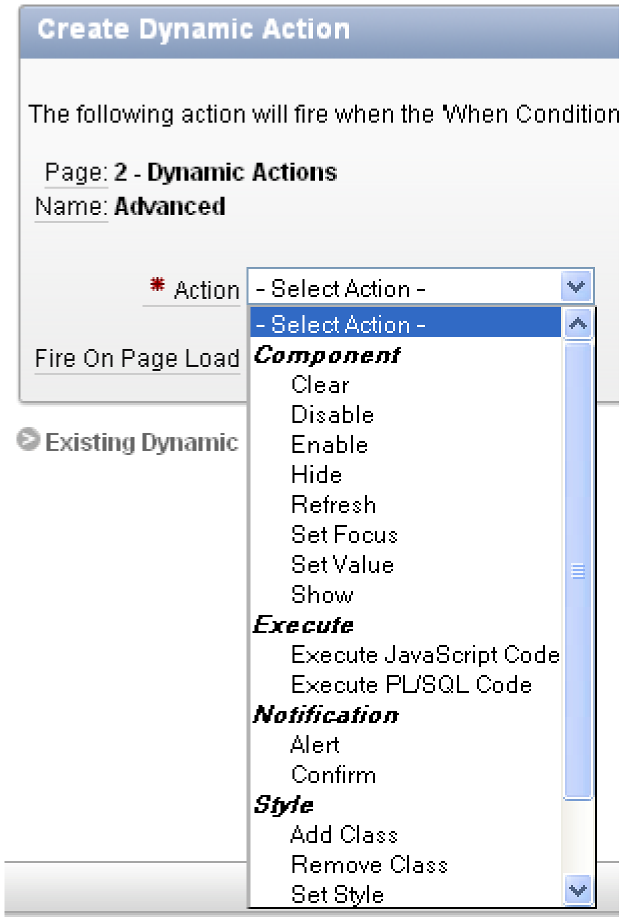
* Sequence	10
* Action	Set Style
	Disable, Enable, Show, Hide, Set Value

Execution Options

* Fire When Event Result Is	True
Fire On Page Load	✓

Settings

* Style Name	font-weight
Value	bold



Create Dynamic Action

The following action will fire when the 'When Condition

Page: **2 - Dynamic Actions**
Name: **Advanced**

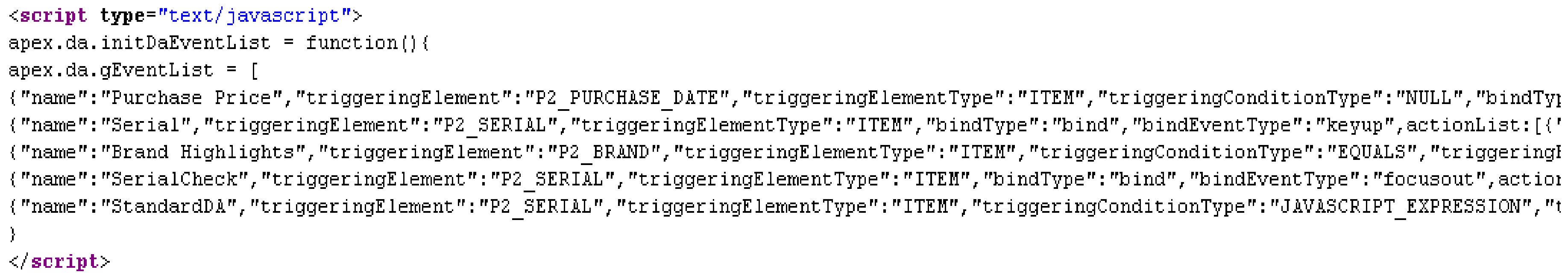
* Action: - Select Action -

Fire On Page Load

Existing Dynamic

Dynamic Actions – Code Generation

- Code is automatically generated for the page
- Can be seen when looking at page source
- Located at bottom of page



```
<script type="text/javascript">
apex.da.initDaEventList = function(){
apex.da.gEventList = [
{"name":"Purchase Price","triggeringElement":"P2_PURCHASE_DATE","triggeringElementType":"ITEM","triggeringConditionType":"NULL","bindTy
{"name":"Serial","triggeringElement":"P2_SERIAL","triggeringElementType":"ITEM","bindType":"bind","bindEventType":"keyup",actionList:[{'
{"name":"Brand Highlights","triggeringElement":"P2_BRAND","triggeringElementType":"ITEM","triggeringConditionType":"EQUALS","triggering
{"name":"SerialCheck","triggeringElement":"P2_SERIAL","triggeringElementType":"ITEM","bindType":"bind","bindEventType":"focusout",actio
{"name":"StandardDA","triggeringElement":"P2_SERIAL","triggeringElementType":"ITEM","triggeringConditionType":"JAVASCRIPT_EXPRESSION","t
}
</script>
```

Dynamic Actions – Conclusion

- Javascript is integral to Apex
- Previous to 4.0, manual coding was necessary
- Dynamic Actions allow for easy generation without coding
- Get opposite action for "free"
- Can extend with custom Javascript, PL/SQL, jQuery
- Makes everyone a Javascript Developer!

Questions?

Topics for Next Time?

Thank You!

Josh Millinger

jmillinger@nianticsystems.com

202.642.6845