Using Javascript in your Apex Applications

Josh Millinger, President
Niantic Systems, LLC
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
- 11+ 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
Niantic Systems

- Oracle Consulting with a Focus on Application Express
- Application Express Training
- Oracle Forms/Reports
- Discoverer
- Mentoring
- Customers in the Federal, Commercial, Healthcare, Higher Education, Construction verticals
What is Javascript?

- Interpreted language with Object Oriented capabilities
- Not JAVA!
- Quite robust
- In most cases, it runs in a browser (client side)
  - However, it can be run on a Web Server such as IIS, Netscape
Javascript Basics

- Case Sensitivity
  - JavaScript is very case sensitive
  - Functions, keywords, variables
  - For example:
    - Amount, AMOUNT, & AmounT are all treated differently

- Comments
  - Anything on a line after //
  - Anything between /* and */
Javascript Basics

- **Variables**
  - Must begin with a letter, _, or $
  - Following characters can be letter, digit, _, $
- **Reserved words**
  - break, case, catch, continue, default, delete, do, else, false, finally, for, function, if, in, instanceof, new, null, return, switch, this, throw, true, try, typeof, var, void, while, with
Javascript Basics

• Strings
  • Can use single or double quotes
    • ‘Hello NYOUG Attendees’
    • “Hello NYOUG Attendees”
  • ‘\n’ is the newline character
  • ‘\’ is the escape character
    • ‘ I won’t use .NET’
  • Concatenate with +
    • var t = ‘This is an ‘+ ‘example.’’
• if (expression)
  {statement1}
else
  {statement2}

• if (name == null)  // == is the equality operator
• {name = ‘Larry’;}  // = is the assignment operator
• switch

```javascript
switch (n) {
    case 'a':
        return 1;
    case 'b':
        return 2;
    default:
        return 0;
}
```
• For
  
  For ( var n=0 ; n < 10 ; n++)
  {
      document.write(n + "<br>");
  }
Functions

- Functions
  - Block of defined code that can be called more than once.
  - Can return value
  - Common convention: start with lowercase and then uppercase all subsequent words
  - Has zero or more args passed in (arg1, arg2,…argn)

```javascript
function writeText (txt)
{
    document.write(txt);
}
```
Basic Functions

- Alert
  ```javascript
  alert(msg);
  alert("You must supply a value for Name");
  ```

- Confirm
  ```javascript
  confirm('Are you sure you want to delete this');
  ```

- Array Operators
  Join, Reverse, Sort, Concat

- String Operators
  toLowerCase, concat, indexOf

- Math
  abs, sin, min, pow, ceil
JavaScript Events

- Triggered when a particular action happens
- Very useful for capturing the timing of action and running appropriate script
- Examples:
  - When button is clicked
  - When select list value is changed
  - When focus leaves an item
# Events

<table>
<thead>
<tr>
<th>Attribute</th>
<th>The event occurs when...</th>
</tr>
</thead>
<tbody>
<tr>
<td>onabort</td>
<td>Loading of an image is interrupted</td>
</tr>
<tr>
<td>onblur</td>
<td>An element loses focus</td>
</tr>
<tr>
<td>onchange</td>
<td>The user changes the content of a field</td>
</tr>
<tr>
<td>onclick</td>
<td>Mouse clicks an object</td>
</tr>
<tr>
<td>ondblclick</td>
<td>Mouse double-clicks an object</td>
</tr>
<tr>
<td>onerror</td>
<td>An error occurs when loading a document or an image</td>
</tr>
<tr>
<td>onfocus</td>
<td>An element gets focus</td>
</tr>
<tr>
<td>onkeydown</td>
<td>A keyboard key is pressed</td>
</tr>
<tr>
<td>onkeypress</td>
<td>A keyboard key is pressed or held down</td>
</tr>
<tr>
<td>onkeyup</td>
<td>A keyboard key is released</td>
</tr>
<tr>
<td>onload</td>
<td>A page or an image is finished loading</td>
</tr>
<tr>
<td>onmousedown</td>
<td>A mouse button is pressed</td>
</tr>
<tr>
<td>onmousemove</td>
<td>The mouse is moved</td>
</tr>
<tr>
<td>onmouseout</td>
<td>The mouse is moved off an element</td>
</tr>
<tr>
<td>onmouseover</td>
<td>The mouse is moved over an element</td>
</tr>
<tr>
<td>onmouseup</td>
<td>A mouse button is released</td>
</tr>
<tr>
<td>onreset</td>
<td>The reset button is clicked</td>
</tr>
<tr>
<td>onresize</td>
<td>A window or frame is resized</td>
</tr>
<tr>
<td>onselect</td>
<td>Text is selected</td>
</tr>
<tr>
<td>onsubmit</td>
<td>The submit button is clicked</td>
</tr>
<tr>
<td>onunload</td>
<td>The user exits the page</td>
</tr>
</tbody>
</table>
• Common syntax you will find in Apex applications

```html
<script type="text/javascript">
  function confirmDelete()
  {
    var x = confirm("Are you sure you wish to delete this record?");
    if (x)
    {
      doSubmit('DELETE');
    }
    else
    {
      return false;
    }
  }
</script>
```
Apex and Javascript

- APEX wouldn’t be APEX without Javascript
- Delete Confirmation
- Menus
- Query Builder
- Drag and Drop Layout
JavaScript Uses

- To accomplish tasks before page processing
  - Validation data
  - Compute new values
  - Confirmations/Alerts
  - Dynamically change look and feel of application through DHTML
  - Ajax for database communication
  - Overall, make the application more like a client/server application
- Set values as page is loading
Validations

- Very useful to increase the interactive nature of applications
- Must be a number or number < 100%
- Should always be used in conjunction with APEX validations.
- Javascript is easy to disable
- Example is Firefox->Tools->Options->Content
Computations

- Javascript can compute values for items on the page
- Assign values through various methods
  - Static Text
    - $x(‘P1_SEARCH’).value=‘Jones’;
    - Document.getElementById(‘REGION_ID’).innerHTML=‘Jones’;
  - Arithmetic/String Functions
    - Document.getElementById(‘P1_SEARCH’).value=someString.toUpperCase();
    - $x(‘P1_SAL’).value=someNumber+1.1;
APEX JavaScript Files

- /i/javascript
  - apex_3_1.js
  - apex_get_3_1.js
  - plugins.js
  - apex_ns_3_1.js
  - htmldb_validate.js
function $x(pNd) {
    try {
        var node;
        switch (typeof (pNd)) {
            case 'string':
                node = document.getElementById(pNd);
                break;
            case 'object':
                node = pNd;
                break;
            default:
                node = false;
                break;
        }
        if (node.nodeType === 1) {
            return node;
        } else {
            return false;
        }
    } catch (e) {
        return false;
    }
}

- * nodeType 1 is Element*
Custom JavaScript Functions

- JavaScript Functions are collections of commands which execute when triggered to do so by an event.
- Can be included in your APEX application in five places:
  - .js File
  - PL/SQL Region on Page Zero
  - Page Header
  - Page Region
  - Page Template
- Best approach is usually a combination of the above methods.
You can put all or some of your JavaScript functions in a static text file and then include that file as part of your page template:

```html
<script src="/c/myFunctions.js" type="text/javascript"></script>
```
.js File

• Benefits:
  • Easier to manage, as you can use a text editor with syntax highlighting
  • Browser will cache the JS file

• Drawbacks
  • Cannot dynamically build JavaScript code
  • Browser will cache the JS file
  • You have to remember to include the .js file when you promote your application to production
• Can call a procedure or be an anonymous PL/SQL block

• Allows you to refer to session state items and use them in your JavaScript

• Most flexible way to implement JavaScript in ApEx

```javascript
var pThis = 'P'+$v('pFlowStepId')+'_CAR_HAVE';
var pThat = 'P'+$v('pFlowStepId')+'_CAR_RIDERS';
$f_DisableOnValue(pThis,'N',pThat);
if ($v(pThis)=='N' || $v(pThis)=='')
{
  $s(pThat,'');
  $x(pThat).disabled=true;
}
```
Benefits
• You can dynamically control what code is generated and include session state variables
  • Good example is APP_ID

Drawbacks
• Each page view will execute the PL/SQL procedure
  • Which isn’t really a bad thing, as it will be pinned in the SGA before long and take almost no time to execute
• If you refer to a named PL/SQL procedure, you must include that procedure when you promote your application to production
• write_javascript procedure
Page

- Can call a JavaScript function when the page loads
- Contents are substituted in the Page Template #ONLOAD# token
Item

- Can associate to an item
- Elements > HTML Form Element Attributes
- Once the event occurs on the Item, it will execute
Button & Column Links

- When a button or Link is clicked, it can either submit the page or redirect to a URL

```javascript
javascript:confirmDelete(htmldb_delete_message,'MULTI_ROW_DELETE');
```
You can also include JavaScript on the Page Header

Only use for JavaScript which will be page-specific
You can also put JavaScript in a Page Region of type HTML

Best Practices:

• Put at the top of your page (Page Position 1, Sequence 1)

• Set Template to **No Template**

• To share among more than one page, consider putting the Region on Page 0
Page Template

• Used when you have some static JavaScript that you want on every single page, and you don’t want to include a .js file

• Drawbacks
  • Browser will never cache like it would a .js file, as each page is different
  • Hard to edit, as edit window in the web page is quite small
doSubmit()

- Part of the Apex JavaScript libraries
- Causes the page to be submitted
- Can also pass in the REQUEST parameter
  - `doSubmit('SAVE');`
Ajax in APEX
Ajax in Apex

• Ajax = Asynchronous JavaScript & XML
• Not a language, but rather an implementation of a suite of technologies which have been around for some time
• “Extends Client-Server like behavior to web applications”
• Apex 2.0 and later makes use of Ajax in the Application Builder & SQL Workshop
• You can take advantage of some of these Ajax libraries in your own applications
Ajax in Apex

- Web 1.0 Applications
  - Navigate to a Page with a Form
  - Fill out the Form
  - Click Submit
  - Page will process some server-side program and return the results back
- Still the safest way to build a web site!
- Apex validations work this way
Ajax in Apex

- Web 2.0 Applications
  - Navigate to a Page with a Form
  - Fill out the Form
  - Get instantaneous feedback on validation errors, dynamically data-driven select lists, etc.
- Submit the form only when data is correct
- While it may be cooler, this method introduces more “moving parts” which have to be adequately secured & managed
- Web 2.0 forms should always have server-side validations
Firebug is your Friend

- Firebug

Employees

<table>
<thead>
<tr>
<th>Emp#</th>
<th>Name</th>
<th>Job</th>
<th>Manager</th>
<th>Hiredate</th>
<th>Salary</th>
<th>Commission</th>
<th>Dept</th>
<th>Deleted</th>
<th>Add Field</th>
<th>Seqfield</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SMITH</td>
<td>CLERK</td>
<td>FORD</td>
<td>12/17/1980</td>
<td>$800.00</td>
<td></td>
<td>RESEARCH</td>
<td></td>
<td>add field</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ALLEN</td>
<td>SALESMAN</td>
<td>BLAKE</td>
<td>02/20/1981</td>
<td>$1,600.00</td>
<td>$300.00</td>
<td>SALES</td>
<td></td>
<td>add field</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>WARD</td>
<td>SALESMAN</td>
<td>BLAKE</td>
<td>02/22/1981</td>
<td>$1,250.00</td>
<td>$500.00</td>
<td>SALES</td>
<td></td>
<td>add field</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>JONES</td>
<td>MANAGER</td>
<td>KING</td>
<td>04/02/1981</td>
<td>$2,975.00</td>
<td></td>
<td>RESEARCH</td>
<td></td>
<td>add field</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MARTIN</td>
<td>SALESMAN</td>
<td>BLAKE</td>
<td>09/29/1981</td>
<td>$1,250.00</td>
<td>$1,400.00</td>
<td>SALES</td>
<td></td>
<td>add field</td>
<td></td>
</tr>
</tbody>
</table>
Coming in Apex 4.0 is Dynamic Action
JQuery
Shown at ODTUG 2009
Declaratively create javascript functionality from builder.
  – Hide Items
  – Change styles
  – Change values
JavaScript: Resources

• Using JavaScript in ApEx Applications:

Thank You!

• If you’re so inclined, send me questions & comments directly:
  – Josh Millinger, Niantic Systems, LLC
  – Phone: 609.945.3151
  – Email: jmillinger@nianticsystems.com