APEX Plugins Unplugged

Christian Rokitta
rokit
Netherlands

Keywords:

Oracle Application Express, Plug-ins, JavaScript, CSS, HTML, API’s

Introduction

Oracle Application Express enables developers to build data-centric web applications declaratively, meaning, you can build feature rich applications, running in web browsers. But APEX declarative functionality is limited, not providing the full capabilities of what is possible with web technologies these days. And, let’s admit, not all developers are able (yet) to easily integrate these capabilities in APEX.

Fortunately APEX offers a mechanism to those who do seem to understand advanced HTML/JavaScript, to expand it’s functionality and expose it in a declarative way to other developers: Plugins.

Unplug yourself from APEX’s declarative limits

Plug-in’s are where it all comes together: PL/SQL, HTML, JavaScript, CSS and APEX framework. At first glance, Plug-in’s seem to be complex. But this isn’t true. Once you have a closer look, you will find that creating Plugin’s isn’t magic. With some basic knowledge and understanding of the APEX Plugin API one can build amazing Plug-in’s. All you need is an idea or inspiration. It will not surprise you, that most (well-known and community-wide used) APEX Plug-ins are made from components prior published as jQuery Plug-in’s or JavaScript Libraries. But it’s not just cut-n-paste.
This session will start with some fundamental Plugin basics to start with and accelerate to more advanced techniques, which will help you to kick-start your own Plugin development.

When to use Plug-ins

Use or create plug-ins in Oracle Application Express whenever you need a component that is not part of the standard components, you want to reuse this component in one or more applications, want to hide the (JavaScript, CSS, HTML) complexity from other developers using it and want to keep the code combined in one container in a maintainable maner.

Skills and Inspiration

Writing plug-ins in Oracle Application Express does ask for, at least, basic knowledge of JavaScript, CSS and HTML technology. Fortunately, lots of problems you might encounter are already solved by developers in the online (jQuery) community. So, writing plug-ins sometimes can be reduced to cut-n-paste coding, if you know the APEX structures to follow.

Structure

Plug-in’s in APEX follow the same Page State methodology as any APEX page:

Render phase: HTML generation, based on templates, definitions and resources

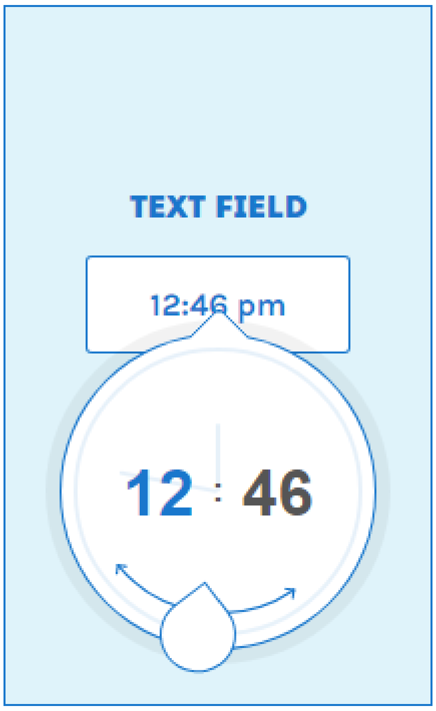
Interaction: Dynamic Actions and AJAX calls

Processing: Submitting the page, validating and processing data

Different plug-in types are relevant in one or more of this page states.

Basic example

Using a basic example, I will convert a free jQuery plug-in (Timedropper) into an APEX Item type plug-in using these small code snippets found in the plug-in’s documentation:



Include library files:

```
<script src="js/timedropper.js"></script>
<link rel="stylesheet" type="text/css"
href="js/timedropper.css">
```

HTML code:

```
<input type="text" id="alarm" />
```

JavaScript initialization:

```
<script>$( "#alarm" ).timeDropper();</script>
```

APEX Plug-in Code

Converting this example into an APEX plug-in, following the APEX API function signature will result in a simple PLSQL function:

```
function render_timedropper(p_item                  in apex_plugin.t_page_item
                          , p_plugin                in apex_plugin.t_plugin
                          , p_value                 in varchar2
                          , p_is_readonly           in boolean
                          , p_is_printer_friendly   in boolean)
   return apex_plugin.t_page_item_render_result
is
   v_result   apex_plugin.t_page_item_render_result;
begin
   apex_javascript.add_library(p_name => 'timedropper'
                             , p_directory => p_plugin.file_prefix
                             , p_check_to_add_minified => true);
   apex_css.add_file(p_name => 'timefropper', p_directory => p_plugin.file_prefix);
   sys.htp.p('<input type="text" id="' || p_page_item.id ||
             '" name="' || p_page_item || '" />');
   apex_javascript.add_onload_code(
        p_code => '$( "#' || p_page_item.id || '" ).timeDropper();', p_key => null);
   return v_result;
end;
```

Using and following the predefined structure and API’s, plug-in development is no magic.

Contact address:

Christian Rokitta
rokit
Agnietenhove 5
3834 XA, Leusden (NL)

Phone:	+31(0)6-41754763
Fax:	-
Email	christian@rokitta.nl
Internet:	www.rokit.nl