JavaScript & Salesforce Visualforce – jQuery Partial ID Selectors vs $Component

JavaScript & Salesforce Visualforce – jQuery Partial ID Selectors vs $Component

JavaScript with Salesforce Visualforce: using the jQuery Partial ID Selector instead of $Component.

When we build Visualforce pages on Salesforce's Force.com platform, we often use JavaScript to add functionality. How we refer to page elements by ID in our JavaScript code, however, is not straightforward.

Here we discuss three ways to refer to Visualforce page elements by ID and show how jQuery partial ID selectors provide a good solution. Using jQuery partial ID selectors allows us to (1) consolidate our JavaScript code and (2) handle changes to our page structure.

1. Hierarchy ID Structure

When Salesforce compiles Visualforce code into HTML, it uses a hierarchy to build ID values for page elements. For example, if we want the ID of the input field in the following Visualforce code, we find that a document.getElementById(**'input'**) will return no results.

```
<apex:page id="page">;
    <apex:form id="form">
        <apex:inputText id="input" />
    </apex:form>
</apex:page>
```

We get no results because when Salesforce compiles the Visualforce code into HTML, the ID is NOT set to **'input'**. Instead, Salesforce has generated an ID for the element which follows the page structure: **'page:form:input'**.

One option is to replace the ID in the JavaScript like this: document.getElementById(**'page:form:input'**). This is not best practice, however, since **adding new elements will change the hierarchy of the page, and hence the selector will break.**

For example, in the code below, the ID of the input has changed to **'page:form:panel:input'**.

```
<apex:page id="page">
    <apex:form id="form">
        <apex:outputPanel id="panel">
            <apex:inputText id="input" />
        </apex:outputPanel>
    </apex:form>
</apex:page>
```

2. $Component selectors

Salesforce best practice is to use the $Component selector – document.getElementById(**'{!$Component.input}'**) – which traverses the hierarchy and gives the ID. **This selection, however, must take place at the same hierarchical level as the element.**

Below, we illustrate making the selection at the same level as the element (which will work) and then making the selection at a different level (which will NOT work):

```
<!-- Will work -->
<apex:page id="page">
    <apex:form id="form">
        <apex:outputPanel id="panel">
            <script>
                function foo() {
                    <!-- This selection is at SAME level as element -->
                    var obj = document.getElementById('{!$Component.input}');
                    obj.value = 'Foo';
                }
            </script>
            <apex:inputText id="input" onfocus="foo();" />
        </apex:outputPanel>
    </apex:form>
</apex:page>


<!-- Will NOT work -->
<apex:page id="page">
    <apex:form id="form">
        <apex:outputPanel id="panel">
            <apex:inputText id="input" onfocus="foo();" />
        </apex:outputPanel>
    </apex:form>
    <script>
        function foo() {
            <!-- This selection is at DIFFERENT level than element -->
            var obj = document.getElementById('{!$Component.input}');
            obj.value = 'Foo';
        }
    </script>
</apex:page>
```

But here's the drawback to this approach: if we need to make selections at various hierarchical levels to properly grab the ID values of the elements we want, **we end up with fragmented JavaScript and script tags spread throughout our code**.

3. jQuery partial ID selectors

A great alternative is to use the jQuery library. jQuery provides **partial ID selectors** that can select IDs by **'Ends With'.**

To work with jQuery in Visualforce, we need to add the jQuery library. We can do this by either (1) adding the jQuery library as a static resource and adding a script include tag on the page, or by (2) linking to a hosted copy of the jQuery library and adding a script include tag. See the jQuery site for more information on downloading and using jQuery.

Here's what a partial ID selector in jQuery looks like:

```
jQuery( 'input[id$=input]' ) //finds all elements with ID ending in input
```

This selector will always find our input field, regardless of how we change the page hierarchy. It works because the ID will always end in **'input'** once the Visualforce code is compiled into HTML.

Notice that we preface the ID selector with **'input'**. This helps jQuery be more efficient because it only needs to check for the ID on HTML tags of type input.

Tip: If you know the type of html tag for the ID you want, preface the ID selector with the tag type to get better performance on large pages.

So let's see the partial ID selector in action:

```
<apex:page id="page">
    <apex:form id="form">
        <apex:outputPanel id="panel">
            <apex:inputText id="input" onfocus="foo();" />
        </apex:outputPanel>
    </apex:form>
    <!-- Uploaded jQuery as a static resource -->
    <apex:includeScript value="{!URLFOR($Resource.JQueryUI_1_8_20, 'js/jquery-1.7.2.min.js')}" />
    <script>
        function foo() {
            jQuery( 'input[id$=input]' ).val('Foo');
        }
    </script>
</apex:page>
```

By using jQuery's partial ID selector, we can consolidate our JavaScript code into one script block. And we never have to worry about changes to our page structure. In addition, we can use the excellent additional features the jQuery library provides.