Auto Complete Input Component (R1.3)

Auto Complete Input is a component built in native salesforce lightning. It can be used while you creating forms of lightning to add data in objects. It will help user to provide values for lookup fields. I will reduce the chances for human error while entering the id of a record in lookup field value. Instead of it user can use this component to add a auto complete and let the user choose records by its name field or any other field by which user can easily identify the records.

This component can be configured to work with any Standard/Custom object and any of it's fields. In addition to this many other parameters are also configurable. It will fire a lightning event on selecting an item from the component and any other component can handle that particular event and use that event's parameters to get the selected value.

The attributes of this component are listed below:

- **object** - API Name of the Object in which you want search.
- **id_field** - API Name of the field that you want to select in auto complete. This field should belong to the object you have provided in the “object” attribute.
- **name_field** - API Name of the field that you want to show in Auto Complete. This field should belong to the object you have provided in the “object” attribute.
- **order_field** - API Name of the field by which you want to order the response.This field should belong to the object you have provided in the “object” attribute.
- **limit_value** - Maximum no of suggestions.
- **min_length** - Minimum no of characters to start Auto complete search.
- **input_label** - Label for the auto complete box. by default it is 'Auto-Complete Input.
    - Note - From R1.2.2 release this attribute is dependent on **show_label** attribute.
- **standard_css** (only supported from R1.2.1 release) - This attribute type is Boolean hence it will have one of the two possible values ie. true/false and depending on the value its UI will change accordingly.
    - If **standard_css = true** then it will be displayed with its predefined css you can use this if it is perfectly situated in your app/component.
    - If **standard_css = false** then it will be displayed with its basic UI without any css and its UI will depends on the browser in which it is opened.

- **show_label** (only supported from R1.2.2 release) - By toggling its value with true/false you can show/hide the label on search box.
- **search_placeholder** (only supported from R1.2.2 release) - It is the text which will be displayed in the search box as placeholder.

This component consists a input box in it. This input will start showing suggestions according to your input. When the user clicks over any of the suggestions it will fire an lightning event , named " autoCompleteEvent ". Any other component can handle that event and use the event parameters named , " search_value " and " search_name ".

Example :
Here is a demo component, In this component we are using our component and handling its event.

AutoCompleteDemo.cmp

```
<aura:component >
        <aura:attribute name="result_id" type="String" access="global"/>
        <aura:attribute name="result_name" type="String" access="global"/>

         <!-- Handling the event of Auto Complete Input component. acinpt  is the namespace of
        component-->
        <aura:handler event="acinpt:autoCompleteEvent" action = " {!c.getValueFromApplicationEvent}"
        />
         <!-- Adding  Auto Complete Input component -->
        <acinpt:autoComplete object="Account" id_field="Id" input_label="Account Auto Complete"
        name_field="Name" order_field="Name" limit_value="10" min_length="3"  standard_css=”true”
        show_label=”true” search_placeholder=”Search here... “ />

        <div style="width:100%;float:left;">
                The result is {!v.result_id}
        </div>
        <div style="width:100%;float:left;">
                Its Name is  {!v.result_name}
        </div>
</aura:component>
```

AutoCompleteDemoController.js

```
({
```

```
        /*
        *  Method to handle the event fired by auto complete input component
        */
        getValueFromApplicationEvent : function(component, event, helper) {
                var ShowResultValue = event.getParam("search_value");
                var ShowResultName = event.getParam("search_name");
                component.set("v.result_id", ShowResultValue);
                component.set("v.result_name", ShowResultName);
        }
})
```

How to override its UI

Follow below steps to change its UI if you are using its basic version (attribute standard_css = false).

- Simply you have to define the CSS at App level, for each element to which you want to adjust in your component. Let's see how we can achieve it.

Suppose you are using AutoComplete component in your component named "AutoCompleteDemo" as shown below,

AutoCompleteDemo.cmp

```
<aura:component >

        <div class="acinpt_div">
                <acinpt:autoComplete object="Account" id_field="Id" input_label="Account Auto
        Complete" name_field="Name" order_field="Name" limit_value="10" min_length="3" />
        </div>

</aura:component>
```

Here are some common customizations with which you will be definitely going to play when you are using its basic version (means standard_css = false).

1) **Adjust margin & padding of the component** - You can simply achieve this by wrapping the component in outer div like we wrapped it in the above e.g. and then you can define the CSS for the div class as in our above case it is ".acinpt_div". This css will be defined in the style of the component in which you are using the AutoComplete component -

   AutoCompleteDemo.css

```
.THIS .acinpt_div {

        margin : 20px;
        padding : 20px;
}
```

2) Change UI of Input box or any of its parent div - Input box is having a style class named "input_box" so you can define the styles in that class at **app level** according to your requirements as shown in the below example

AutoCompleteDemoApp.app

```
<aura:application>

        <c:autoCompleteDemo>

</aura:application>
```

Syntax of the app CSS -

```
.THIS  .class_of_input_box {

        // define your styles here

}
```

For our **AutoCompleteDemoApp** we will define CSS like this -

AutoCompleteDemoApp.css

```
.THIS  .input_box {

        width:20%;
        border : 1px solid #000;

        // many styles here

}
```

Using the same way as we defined the above css for the input_box we can find the class of each div or element by inspecting the component in the browser and we can redefine its css in Your_app.css.