How many types of event in salesforce?

in salesforce their is a two types of event(Application Evevnt, Component Event)

how to register a event in component.cmp?

How to fire an event in controller.js?

How to handle an event in component.cmp?

Explain an event life cycle?

What is lightning?

How to create a namespace?

\*\*What is the Lightning Experience?\*\*

Lightning is a component based framework which is inbuilt aura framework and aura is a open source and open source meins where things are available for free,

\*\*What is an Attribute in Lightning ? \*\*

Attributes are used to store information to be referenced and used in our lightning component code.

\*\*How to Add or Declare attributes in lightning component or lightning application ? \*\*

Below is the syntax of using attribute in lightning component or application

<aura:component>

<aura:attribute name="yourName" type="String" default="Marc Benioff"/>

</aura:component>

So basically in the above sample syntax we have a \*\*‘aura:attribute’\*\* In which, attribute Name is \*\*‘yourName‘\*\* and its Type is String. and we are give it default value as \*\*‘Marc Benioff\*\*’.

Attributes are always declare with \*\*aura: namespace\*\*.

In the attribute creation, attribute Name and attribute Type is \*\*mandatory properties\*\*.

An attribute can have a type corresponding to a standard or custom object. Such as :

<aura:component>

<aura:attribute name="objAccount" type="Account" />

<aura:attribute name="objCustomObject" type="customObject\_\_c" />

</aura:component>

\*\*How to Use or Access aura:attribute In Lightning Component or Lightning Application?\*\*

To access attribute in our Lightning Component or Lightning Application, we have to use \*\*value providers\*\*. value provider is a syntax to access \*\*‘aura:attributes’\*\* to lightning component code.

To access a attribute in component markup using the expression \*\*{! v.attributeName}\*\* syntax. here ‘v.‘ gives you a ‘control’ to access the component attributes.

\*\*Example:\*\*

Open Developer Console and Create a lightning component (File > New > Lightning Component)

Enter name “testAttribute“. and write following code.

testAttribute.cmp

<aura:component>

<aura:attribute name="FirstName" type="String" default="Marc Benioff" />

<aura:attribute name="Age" type="Integer" default="52"/>

<aura:attribute name="isMale" type="Boolean" default="true"/>

<p>Name of Salesforce CEO is {!v.FirstName}</p>

<p>{!v.FirstName} is {!v.Age} Years Old.</p>

<p>{!v.FirstName} is male? = {!v.isMale}</p>

</aura:component>

Create an app to see component output:

Open Developer Console and Create a lightning Application (File > New > Lightning Application)

testAttribute.app

<aura:application>

<c:testAttribute/>

</aura:application>

Click on the preview button in developer console sidebar.

Output:

Output of Using Attributes in lightning component

In the above “testAttribute” lightning component code, first we have create 3 \*\*‘aura:attribute’,\*\* with specific attribute type such as : string, Integer and Boolean.

For use this attributes value in lightning component we are using the expression syntax \*\*"{! v.attributeName}"\*\* with value provider “v.”

<aura:component>

<aura:attribute name="FirstName" type="String" default="Marc Benioff" />

<aura:attribute name="Age" type="Integer" default="52"/>

<aura:attribute name="isMale" type="Boolean" default="true"/>

<p>{!'Name of Salesforce CEO is ' + v.FirstName}</p>

<p>{!v.FirstName + ' is ' + v.Age + ' Years Old.'}</p>

<p>{!v.FirstName + ' is male? = ' + v.isMale}</p>

</aura:component>

Notice that we’ve used the “+” operator to concatenate the two strings together inside expression.

Output is same as Example 1 output.

\*\*What is Event-driven architecture?\*\*

\*\*What is Event?\*\*

The framework uses events to communicate data between components. Events are usually triggered by a user action.

suppose when a user click on any button. like, onClick, Press, onmouseover, etc

Events are fired from JavaScript controller actions that are typically triggered by a user interacting with the user interface.

Events are declared by the \*\*aura:event\*\* tag in a \*\*.evt\*\* resource, and they can have one of two types: component or application

There are two types of event (Application Event, Component Event)

\*\*Application Event:-\*\*

\*\*Component Event:-\*\*

Component events are parent-child relationships, in which we are registered an event in the child component and also it will fire in the child component controller and that would be handle only by the parent component and not other sibling components

\*\*1. Create Custom Component Event:-\*\*

\*\*2. Fire Component Events:-\*\*

\*\*3. Handling Component Events:-\*\*

Example:-

\*\*productAddItem.evt\*\*

<aura:event type="COMPONENT">

<aura:attribute name="products" type="Products\_\_c" />

</aura:event>

Events are defined as a separate entity from components, controllers and helper, and The event definition also determines the event type: COMPONENT or APPLICATION.

\*\*productForm.cmp (Child Component)\*\*

<aura:component>

<aura:attribute name="products" type="Products\_\_c" />

<lightning:button label="Add Product" press="{!c.clickAddProduct}" />

<aura:registerEvent name="addProduct" type="c:productAddItem" />

</aura:component>

this form component has a button which has attached to the event. The component registers an event (\*\*registerEvent\*\* is your sending component) named \*\*addProduct\*\*.

Keep this name in mind, because the receiving component will call this same name. It’s binds the sender and receiver together.

The button references the \*\*clickAddProduct\*\* function in its controller, which begins the event process.

\*\*productFormController.js\*\*

({

clickAddProduct: function(component, event, helper) {

var products = component.get(“v.products");

var createEvent = component.getEvent(“addProduct");

createEvent.setParams({“products" : products});

createEvent.fire();

}

})

\*\*products.cmp (Parent Component)\*\*

<aura:component>

<aura:attribute name=“products" type=“Products\_\_c"/>

<aura:handler name="addProduct" event="c:productAddItem" action="{!c.handleAddProduct}" />

<c:productForm />

</aura:component>

Just like we registered the event in the \*\*child component\*\*, now we \*\*handle\*\* it in the \*\*parent component\*\* with a \*\*handler\*\*.

Note that the name, \*\*addProduct\*\*, is the same as when we registered the event. The action attribute defines the next step in the loop, which calls a function in the controller called \*\*handleAddProduct\*\*.

\*\*productsController.js\*\*

({

handleAddProduct: function(component, event, helper) {

var newProduct = event.getParam(“products");

var action = component.get("c.saveProduct");

action.setCallback(this, function(response) {

});

$A.enqueueAction(action);

}

})

This is the receiving controller that handles the incoming event and communicates with the apex controller. The function, saveProduct, is a reference to the apex controller. The final line of code queues up the action (the call to the server), the event cycle is complete.

\*\*1. Create Custom Application Event:-\*\*

\*\*2. Fire Application Events:-\*\*

\*\*3. Handling Appication Events:-\*\*

\*\*<!--c:appEvent-->\*\*

<aura:event type="APPLICATION" description="Event template">

<aura:attribute name="message" type="String"/>

</aura:event>

The component that fires an event can set the event’s data. To set the attribute values, call event.setParam() or event.setParams(). A parameter name set in the event must match the name attribute of an <aura:attribute> in the

event. For example, if you fire c:appEvent

event.setParam("message", "event message here");

The component that handles an event can retrieve the event data. To retrieve the attribute in this event, call event.getParam("message") in the handler’s client-side controller.

A component registers that it may fire an application event by using <aura:registerEvent> in its markup. The name attribute is required but not used for application events. The name attribute is only relevant for component events. This example uses

name="appEvent" but the value isn’t used anywhere.

<aura:registerEvent name="appEvent" type="c:appEvent"/>

Use $A.get("e.myNamespace:myAppEvent") in JavaScript to get an instance of the myAppEvent event in the myNamespace namespace.

var appEvent = $A.get("e.c:appEvent");

// A parameter’s name must match the name attribute of one of the event’s <aura:attribute> tags

//appEvent.setParams({ "myParam" : myValue });

appEvent.fire();

Use <aura:handler> in the markup of the handler component.

<aura:handler event="c:appEvent" action="{!c.handleApplicationEvent}"/>

The event attribute specifies the event being handled. The format is namespace:eventName.

The action attribute of <aura:handler> sets the client-side controller action to handle the event.

The handler for an application event won’t work if you set the name attribute in <aura:handler>. Use the name attribute only when you’re handling component events.

In this example, when the event is fired, the handleApplicationEvent client-side controller action is called.

\*\*Application Event\*\*

The aeEvent.evt application event has one attribute. We’ll use this attribute to pass some data in the event when it’s fired.

<!--c:aeEvent-->

<aura:event type="APPLICATION">

<aura:attribute name="message" type="String"/>

</aura:event>

\*\*Notifier Component\*\*

The aeNotifier.cmp notifier component uses aura:registerEvent to declare that it may fire the application event. The name attribute is required but not used for application events. The name attribute is only relevant for component events.

The button in the component contains a onclick browser event that is wired to the fireApplicationEvent action in the client-side controller. Clicking this button invokes the action.

\*\*<!--c:aeNotifier-->\*\*

<aura:component>

<aura:registerEvent name="appEvent" type="c:aeEvent"/>

<h1>Simple Application Event Sample</h1>

<lightning:button label="Click here to fire an application event" onclick="{!c.fireApplicationEvent}" />

</aura:component>

The client-side controller gets an instance of the event by calling $A.get("e.c:aeEvent"). The controller sets the message attribute of the event and fires the event.

\*\*/\* aeNotifierController.js \*/\*\*

({

fireApplicationEvent : function(component, event, helper) {

// Get the application event by using the e.<namespace>.<event> syntax

var appEvent = $A.get("e.c:aeEvent");

appEvent.setParams({

"message" : "An application event fired me. " +

"It all happened so fast. Now, I'm everywhere!" });

appEvent.fire();

}

})

The aeHandler.cmp handler component uses the <aura:handler> tag to register that it handles the application event.

When the event is fired, the handleApplicationEvent action in the client-side controller of the handler component is invoked.

\*\*<!--c:aeHandler-->\*\*

<aura:component>

<aura:attribute name="messageFromEvent" type="String"/>

<aura:attribute name="numEvents" type="Integer" default="0"/>

<aura:handler event="c:aeEvent" action="{!c.handleApplicationEvent}"/>

<p>{!v.messageFromEvent}</p>

<p>Number of events: {!v.numEvents}</p>

</aura:component>

The controller retrieves the data sent in the event and uses it to update the messageFromEvent attribute in the handler component.

\*\*/\* aeHandlerController.js \*/\*\*

({

handleApplicationEvent : function(component, event, helper) {

var message = event.getParam("message");

// set the handler attributes based on event data

component.set("v.messageFromEvent", message);

var numEventsHandled = parseInt(component.get("v.numEvents")) + 1;

component.set("v.numEvents", numEventsHandled);

}

})

The aeContainer.cmp container component contains the notifier and handler components. This is different from the component event example where the handler contains the notifier component.

\*\*<!--c:aeContainer-->\*\*

<aura:component>

<c:aeNotifier/>

<c:aeHandler/>

</aura:component>

You can test this code by adding <c:aeContainer> to a sample aeWrapper.app application and navigating to the application.

<aura:application extends="force:slds">

<c:aeContainer/>

</aura:application>

<http://bobbuzzard.blogspot.com/2015/05/lightning-component-events.html>

<https://salesforce.stackexchange.com/questions/191979/application-event-in-lightning>

<http://reidcarlberg.github.io/lightning-newbie/events.html>

<https://webkul.com/blog/how-to-fire-lightning-event-from-visualforce-page/>

<https://developer.salesforce.com/docs/atlas.en-us.lightning.meta/lightning/events_intro.htm>

\*\*What is RegisterEvent, When to use and how to use\*\* \*\*RegisterEvent in the component\*\*\*\*?\*\*

\*\*What if fire and when to fire?\*\*

\*\*What is aura:Handler and when to use?\*\*