Salesforce Lightning

Lesson 04: Getting started with Lightning Component Framework

Capgemini

Lesson Objectives

Getting started with Lightning Experience

- Setting up Custom Domain
- Salesforce Lightning Component Framework
- What is lightning Component
- Working with attributes
- Expressions
- Value Provider
- Global Value Provider

Setting up custom Domain

- Have your own URL
- Can be used to login thro social account – like google, facebook
- Can use different orgs at the same time

To create custom domain

- Go to setup
- Type “My Domain” in quick find and then provide (unique) domain name
- Rules to create domain name:
  - Don’t use reserve words www, salesforce and Heroku
  - Don’t use keywords root, status and hyphen

Salesforce Lightning Component framework

- Provides lot of out-of-box components
- Event driven architecture
- Framework is optimized for performance
- Performance efficient
- Built out of Aura framework

What is a Lightning Component?

- Lightning framework is made up of components
- Components are self-contained and reusable units of app
- Eg. Like assembling different parts of car

Create lightning component in Developer console

- Creates two file myComp and myComp.cmp
- Use these components in Lightning App
- You can attach styles to the component

Demo

Create a helloComp lightning component

Use it in Lightning App

Adding styles to our Lightning Applications

- We can add styles to our lightning components
  - By adding style sheets
  - By adding external style sheets
- Add style in “STYLE” part of component
- To add default style

```
.THIS {
        background-color: blue;
}

H1.THIS{                        //for h1 tag
        font-size:60px;
}

.THIS .red{                     //for class red
        background-color:red;
}
```

Add external style

Add external resource file into static resources

E.g. bootstrap.css should be uploaded in static resources

To use the classes from this .css file use following format:

```
<ltng:require style="/resource/bootstrap"/>
<div class="class from bootstrap.css">
        HELLO
</div>
```

Attributes in Lightning

- Attribute is a member variable

- Declare Attribute in Lightning component

```
<aura:attribute name="yourName" type="String" default="Sachin Tendulkar" description="Variable to store the name"/>
```

- Attributes are always declare with aura: namespace.

Attribute has 2 mandatory properties:

- attribute Name and
- attribute Type

Attributes in Lightning

- Some other attributes:
  - default,
  - access,
  - required,
  - Description

- Attribute can have a type corresponding to a standard or custom object:

```
<aura:attribute name="yourName" type="String" default=“Sachin Tendulkar" description=“Variable to store the name"/>
```

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

Access attribute in Lightning Component/Lightning application

- we have to use value providers to access attribute
- value provider is just a syntax to access 'aura:attributes' to lightning component code
- use the expression **{! v.attributeName}** here 'v.' gives you a 'control' to access the component attributes

```
<aura:component>
    <aura:attribute name="FirstName" type="String"
            default="Marc Benioff" />
    <aura:attribute name="Age" type="Integer" default="52"/>
    <aura:attribute name="isStud" type="Boolean" default="false"/>
   <p>Name is {!v.FirstName}</p>
   <p>{!v.FirstName} is {!v.Age} Years Old.</p>
    <p>{!v.FirstName} is student? = {!v.isStud}</p>
</aura:component>
```

Component attributes

- Component attributes can be created in lightning application
- They are to be created in the same way, using <aura:attribute> tag
- These attributes can have default values or you can pass them as string parameter in URL

```
<aura:application >
  <aura:attribute name="param" type="string" default="world"/>
   <h1> Hello {!v.param}</h1>
</aura:application>
```

Component Composition

Component Composition:

- We can create a new component using existing components

- Create a helloHTML component as first component

- Create another helloAttributes component with attributes

- Create nested component which is composed of the above to components

- Create lightning application to add these components

Expression

- Expression syntax {!expression}
- Expressions allow you to make calculations and access property values
- Expressions are evaluated and dynamically replaced when the component is rendered

- Rules for using the expression
- Only use the {!} syntax in markup in .app or .cmp resources
- In Javascript use string syntax to evaluate an expression
- eg: var str = cmp.get("v.label");

Value provider

- Value providers are the way to access data
- Value providers encapsulate related values together, just list properties and methods are encapsulated in an object
- Value provider for the component are v(view) and c (controller)
- Components view refers to attribute set
- Components controller enables you to wire up event handlers and actions for the components – this is the place to control components logic
- All components have v value provider, but they aren’t required to have a controller
- Both value providers are created automatically when defined for a component

Global Value Provider

- Global value providers are global values and methods that a component can use in expression

- globalID – returns global ID of the component
- Every component has a unique global id, generated at runtime

- $Browser – returns information about the hardware and the operating system of the browser

- $label – enables you to access label stored outside your code

- $local – returns information about current user local

Demo

Create component

Create attributes

Create composition component

Create styles

Summary

Create Lightning Components

Attributes can be added to components and application

Component values can be accessed

Global value provides - $local, $Browser

People matter, results count.

About Capgemini

With more than 190,000 people, Capgemini is present in over 40 countries and celebrates its 50th Anniversary year in 2017. A global leader in consulting, technology and outsourcing services, the Group reported 2016 global revenues of EUR 12.5 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Learn more about us at

www.capgemini.com