61. Component Communications, Parent to Child Communication - 29 June 2022

Today we will learn:

1] Component Communications

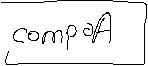
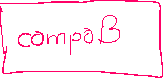
2] Types of Component Communications

3] Parent to Child Communication

4] Decorators (@api, @track, @wire)



1] Component Communications



2] Types of Component Communications

**A] Parent to Child Communication**

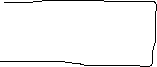
**B] Child to Parent Communication**

**C] Independent / Sibling Component Communication (No Relation)**

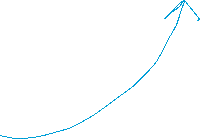
* **Pub-Sub**
* **Lightning Messaging Service**

**D] Communication with Visualforce Pages**

**A] Parent to Child Communication**



<c-child-compo p={x}> </c-child-compo>



Decorator

* **@api**:

Apni Samaz : It is used to expose/public the properties of a component. It is usable when we want to call one component property from another component.

OR

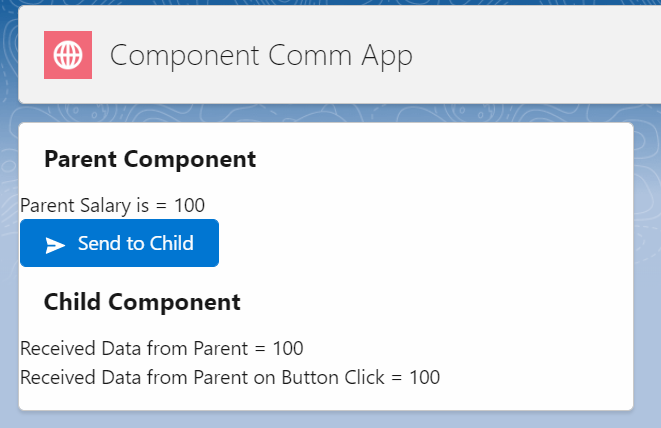
To expose a public property, decorate it with @api. Public properties define the API for a component. An owner component that uses the component in its markup can access the component’s public properties. Public properties are reactive. If the value of reactive property changes, the component’s template rerenders any content that references the property.

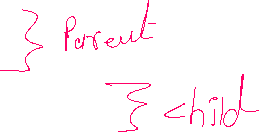
* **@track**:

Apni Samaz: It used to make a property reactive. That mean used to make some changes in the property.

OR

* To track a private property’s value and rerender a component when it changes, decorate the property with @track. Tracked properties are also called private reactive properties.
* **@wire**: To read Salesforce data, Lightning web components use a reactive wire service. When the wire service provisions data, the component rerenders. Components use @wire in their JavaScript class to specify a wire adaptor or an Apex method.





Parent Compo:

<template>

    <lightning-card title="Parent Component">

        Parent Salary is = {parentSalary} <br/>

        <lightning-button variant="brand" icon-name="utility:send" label="Send to Child" onclick={sendToChildHandler} > </lightning-button>

       <!-- Calling Child Compo -->

        <c-child-compo received-from-parent={parentSalary} received-on-button-click={sendData}></c-child-compo>

    </lightning-card>

</template>

import { LightningElement } from 'lwc';

export default class ParentCompo extends LightningElement {

    parentSalary=100;

    sendData;

    sendToChildHandler(){

        this.sendData = this.parentSalary;

    }

}

Child Compo:

<template>

    <lightning-card title="Child Component">

         Received Data from Parent = {receivedFromParent}   <br />

         Received Data from Parent on Button Click = {receivedOnButtonClick}

    </lightning-card>

</template>

import { LightningElement,api } from 'lwc';

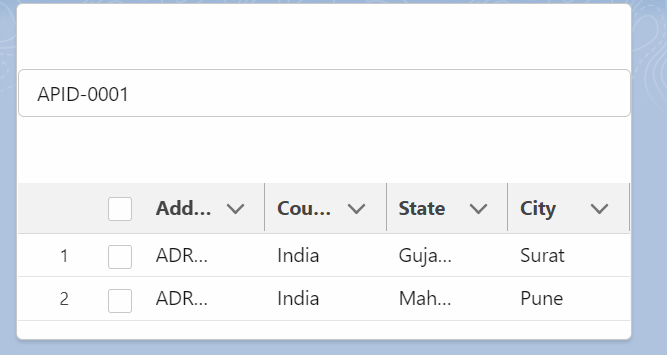
export default class ChildCompo extends LightningElement {

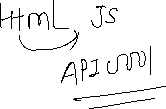
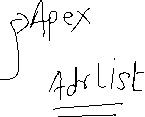
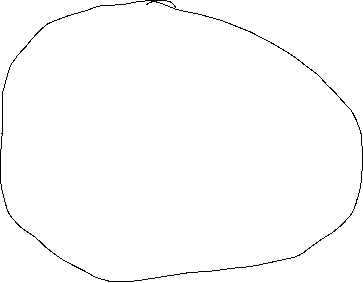
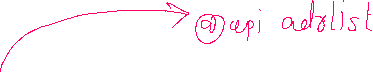
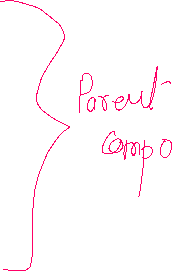
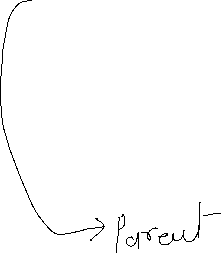
   @api receivedFromParent=0;

   @api receivedOnButtonClick=0;

}

===================================================================================





applicantCompo 🡨 🡪 addressCompo

<template>

    <lightning-card title="Applicant Information">

        <lightning-input type="search" data-formfield="applicantId" label="Enter Applicant ID" placeholder="example API-0001" onblur={searchAddressesHandler}></lightning-input> <br /><br />

        <!-- Calling Address Table Compo-->

            <c-address-table-compo received-address-list={addressList}> </c-address-table-compo>

    </lightning-card>

</template>

import { LightningElement } from 'lwc';

import searchAddresses from '@salesforce/apex/AddressProvider.searchAddresses';

export default class ApplicantCompo extends LightningElement {

    objApplicant = {'sObjectType' : 'Applicant\_\_c'}

    addressList;

    searchAddressesHandler(){

        this.objApplicant.Name = this.template.querySelector('lightning-input[data-formfield="applicantId"]').value;

        console.log(this.objApplicant.Name);

        searchAddresses({objApp : this.objApplicant})

        .then((result)=>{

            console.log(JSON.stringify(result));

            this.addressList = result;

        })

        .catch((error)=>{

            console.log(JSON.stringify(error));

        });

    }

}

Address Compo

<template>

    <lightning-card title="Address Table">

<lightning-datatable

        key-field="Id"

        data={receivedAddressList}

        columns={columns}

        draft-values={draftValues}

         >

</lightning-datatable>

    </lightning-card>

</template>

import { LightningElement,api } from 'lwc';

const  columns = [

    { label: 'Name', fieldName: 'Name', editable: true },

    { label: 'Country', fieldName: 'Country\_\_c', editable: true },

    { label: 'State', fieldName: 'State\_\_c', editable: true },

    { label: 'City', fieldName: 'City\_\_c', editable: true }

  ];

export default class AddressTableCompo extends LightningElement {

    @api receivedAddressList;

    columns = columns;

}

public with sharing class AddressProvider {

    @AuraEnabled

    public static List<Address\_\_c> searchAddresses(Applicant\_\_c objApp){

        try {

            Id appId  = [select id, First\_Name\_\_c from Applicant\_\_c where Name =:objApp.Name LIMIT 1].Id;

            if( objApp !=null ){

                return [select Id, Name, Country\_\_c, State\_\_c, City\_\_c from Address\_\_c where Applicant\_\_c  =: appId];

            }else{

                return null;

            }

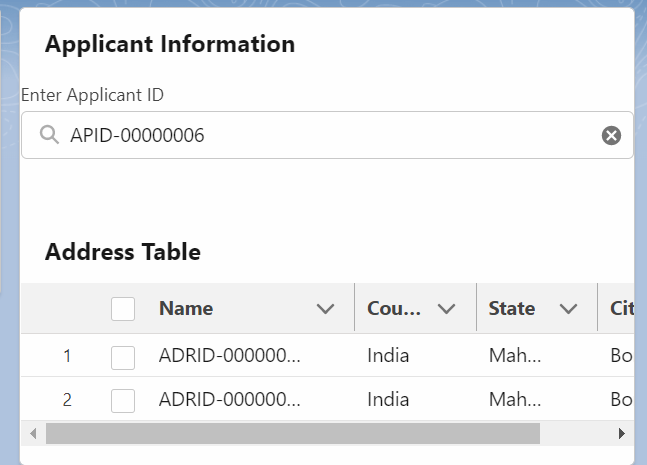
        } catch (Exception e) {

            throw new AuraHandledException(e.getMessage());

        }

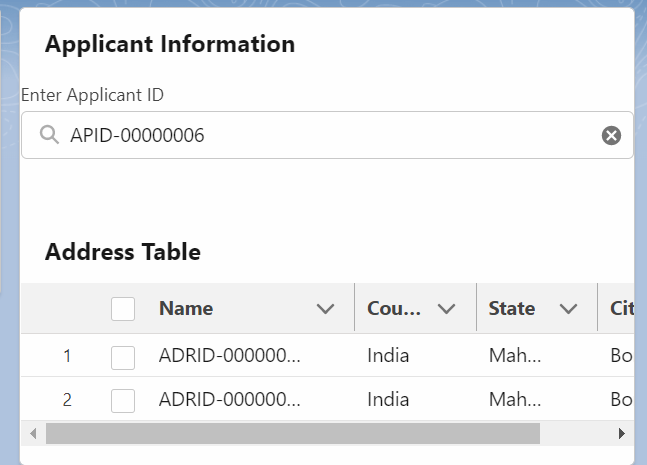
    }

}



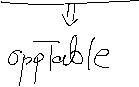
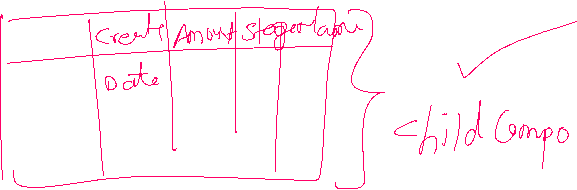
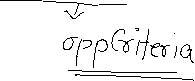
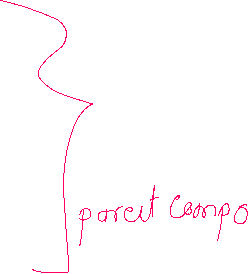
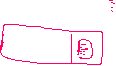
Assignment:







Assignment 2:



Aisa bhi kar sakte ho:

public with sharing class AddressProvider {

    @AuraEnabled

    public static List<Address\_\_c> searchAddresses(Applicant\_\_c objApp){

        try {

            Id appId  = [select id, First\_Name\_\_c from Applicant\_\_c where Name =:objApp.Name LIMIT 1].Id;

            if( objApp !=null ){

                return [select Id, Name, Country\_\_c, State\_\_c, City\_\_c,Applicant\_\_r.Name from Address\_\_c where Applicant\_\_r.Name =:objApp.Name];

            }else{

                return null;

            }

        } catch (Exception e) {

            throw new AuraHandledException(e.getMessage());

        }

    }

}

Comments:

|  |  |  |
| --- | --- | --- |
|  | **Coding / JS Controller** | HTML Tags |
| Single Line Comment | // | <!-- --> |
| Multi Line Comments | /\*  CODE  \*/ | <!--  HTML TAGE  --> |

Shortcut : Single Line = CTRL + /

Multi Line 🡺 Select Line and CTRL + /