Sfdc Telugu

LWC- @wire

Beginner's Guide

Using Wire Property vs Wire Function in Lightning Web Components to Search Contacts

Lightning Web Components (LWC) provide a powerful and reactive way to fetch data from Salesforce Apex classes using the @wire decorator. In this post, we will explore two common patterns to use @wire:

- Wire as a Property
- Wire as a Function

We will build a simple Contact Search app that fetches contacts dynamically based on the user input.

What We Are Building

A search box to find contacts by their first or last name, with two sections showing results using:

1. Wire Property
2. Wire Function

Apex Class: Server-side Logic

First, the Apex class searchContact exposes a method to search contacts based on a search string:

```
public with sharing class searchContact {
    @AuraEnabled(cacheable=true)
    public static List<Contact> findContacts(String searchText) {
        if (String.isBlank(searchText)) {
            return new List<Contact>();
        }

        String sk = '%' + searchText + '%';
        return [
            SELECT Id, Name, Phone, Email
            FROM Contact
            WHERE FirstName LIKE :sk OR LastName LIKE :sk
            LIMIT 50
        ];
    }
}
```

Key points

- @AuraEnabled(cacheable=true) allows LWC to call this method reactively and cache results.
- Search uses SQL LIKE with % wildcards for partial matching.

LWC JavaScript: Fetching Data with Wire

```
import { LightningElement, wire } from 'lwc';
import findContacts from
'@salesforce/apex/searchContact.findContacts';

export default class ContactFilters extends
LightningElement {
    searchkey = '';

    // Wire Property
    contactsProperty;

    // Wire Function
    contactsFunction = [];
    errorFunction;
```

```
// Wire as Property: auto manages data and error in contactsProperty
@wire(findContacts, { searchText: '$searchkey' })
wiredContactsProperty(value) {
this.contactsProperty = value; // value contains { data, error }
}

// Wire as Function: manually handle data and error
@wire(findContacts, { searchText: '$searchkey' })
wiredContactsFunction({ data, error }) {
if (data) {
this.contactsFunction = data;
this.errorFunction = undefined;
} else if (error) {
this.errorFunction = error;
this.contactsFunction = [];
}
}

// Update search key when user types
handleSearch(event) {
this.searchkey = event.target.value;
}
}
```

Explanation

- The reactive parameter $searchkey triggers Apex calls on every change.
- Wire as Property returns a value object with .data and .error.
- Wire as Function destructures data and error to manually control component variables.

LWC HTML Template

```
<template>
   <lightning-card title="Search sss Contact - Wire Property
& Wire Function">


      <lightning-input
         type="search"
         label="Enter Name"
         value={searchkey}
         onchange={handleSearch}>
      </lightning-input>

      <template if:true={searchkey}>
```

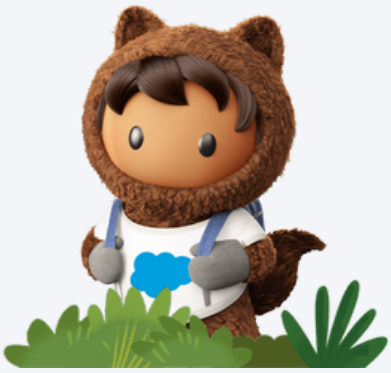
```
<h3>Wire as Property</h3>
<template if:true={contactsProperty.data}>
<template if:true={contactsProperty.data.length}>
<template for:each={contactsProperty.data}
for:item="con">
<p key={con.Id}>
<b>Name:</b> {con.Name} | <b>Email:</b> {con.Email}
</p>
</template>
</template>
<template if:false={contactsProperty.data.length}>
<p>No contacts found.</p>
</template>
</template>
<template if:true={contactsProperty.error}>
<p style="color:red;">Error loading contacts.</p>
</template>

<hr/>

<h3>Wire as Function</h3>
<template if:true={contactsFunction.length}>
<template for:each={contactsFunction} for:item="con">
<p key={con.Id}>
<b>Name:</b> {con.Name} | <b>Email:</b> {con.Email}
</p>
```

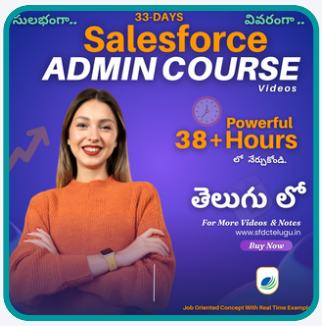
```
</template>
</template>
<template if:false={contactsFunction.length}>
<p>No contacts found.</p>
</template>
<template if:true={errorFunction}>
<p style="color:red;">Error loading contacts.</p>
</template>

</template>
</lightning-card>
</template>
```

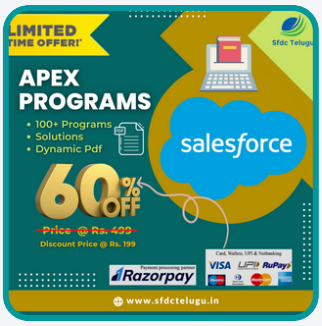
Summary

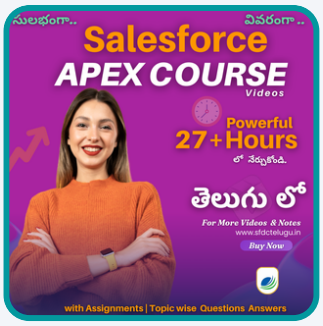
- Use Wire Property for simple, declarative data fetching.
- Use Wire Function when you need to process or transform data, or handle errors differently.
- Both approaches rely on reactive parameters to refresh data dynamically.

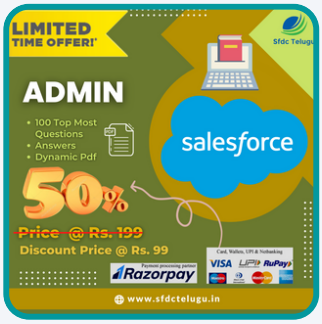
SFDC STORE

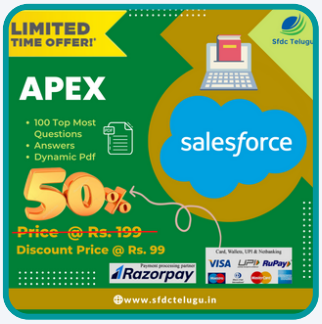


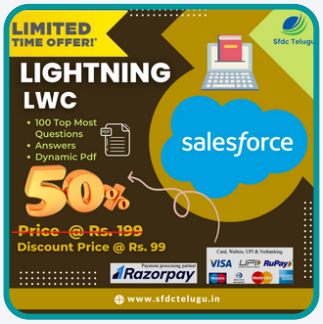


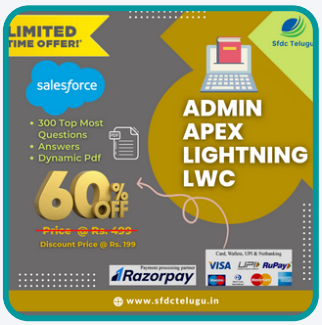










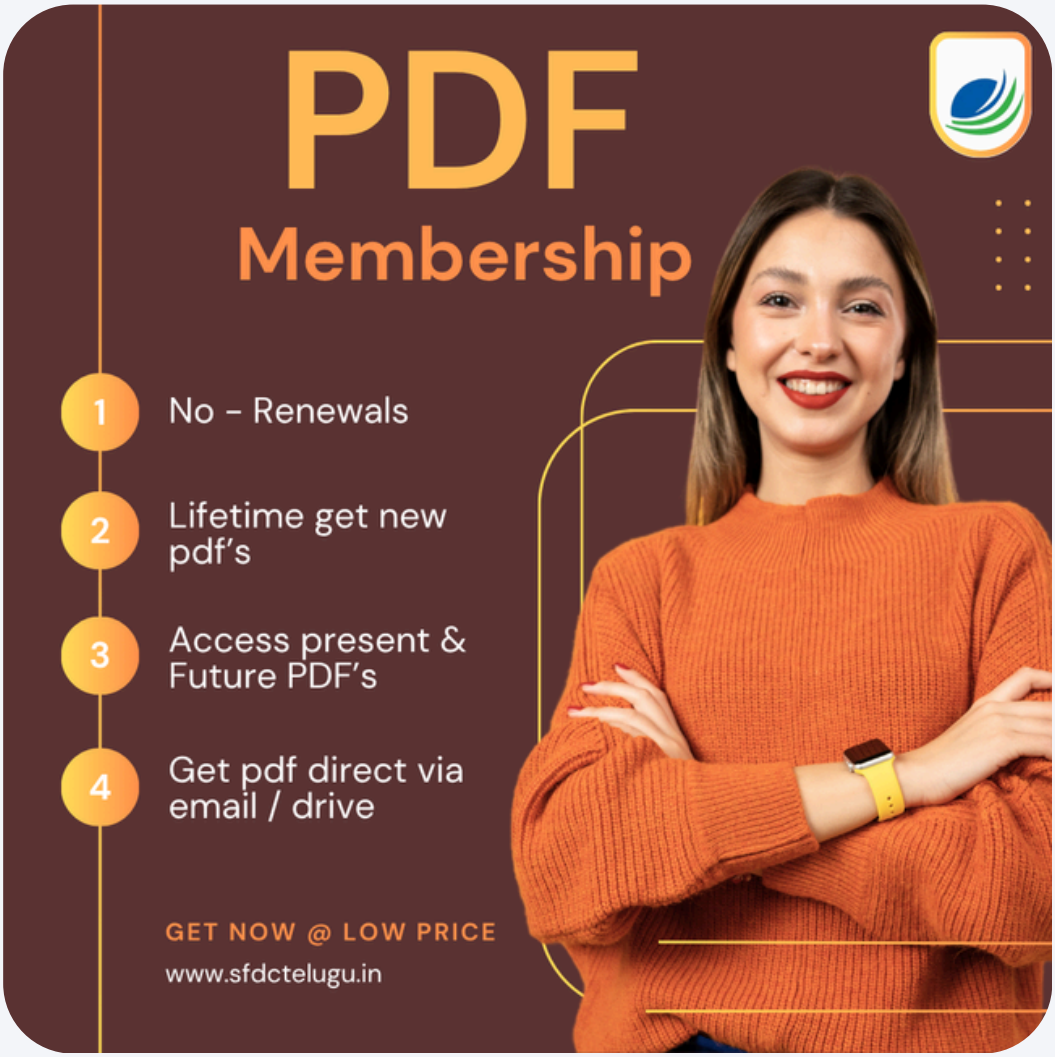


www.sfdctelugu.in/shop

VISA MasterCard PayPal RuPay G Pay paytm PhonePe BHIM
OLAMONEY JioMoney freecharge MobiKwik NEFT IMPS NET BANKING BANK TRANSFER

https://www.youtube.com/@sfdctelugu

SFDC STORE



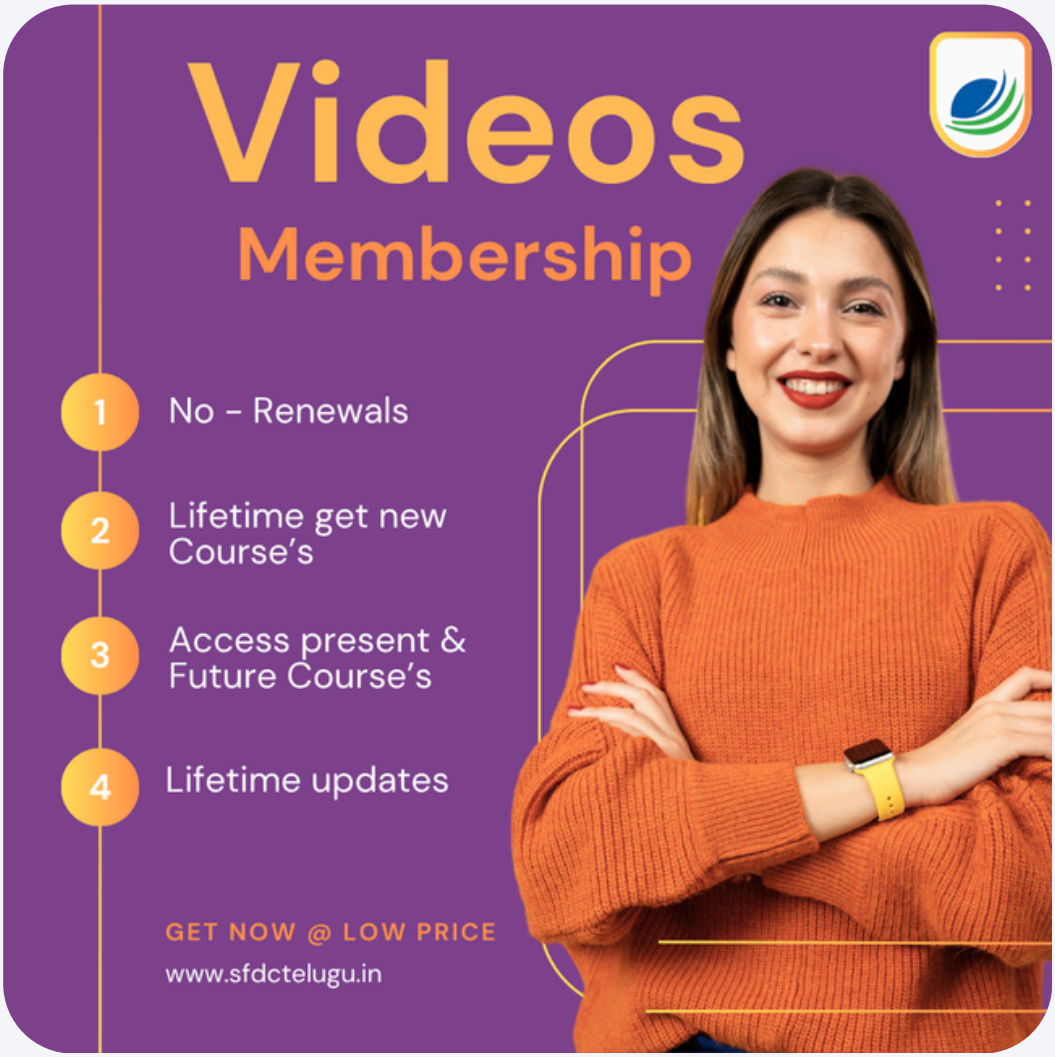
Get Access Now

www.sfdctelugu.in/shop

VISA MasterCard PayPal RuPay G Pay Paytm PhonePe BHIM
OLAMONEY JioMoney freecharge MobiKwik NEFT IMPS NET BANKING BANK TRANSFER

https://www.youtube.com/@sfdctelugu

SFDC STORE



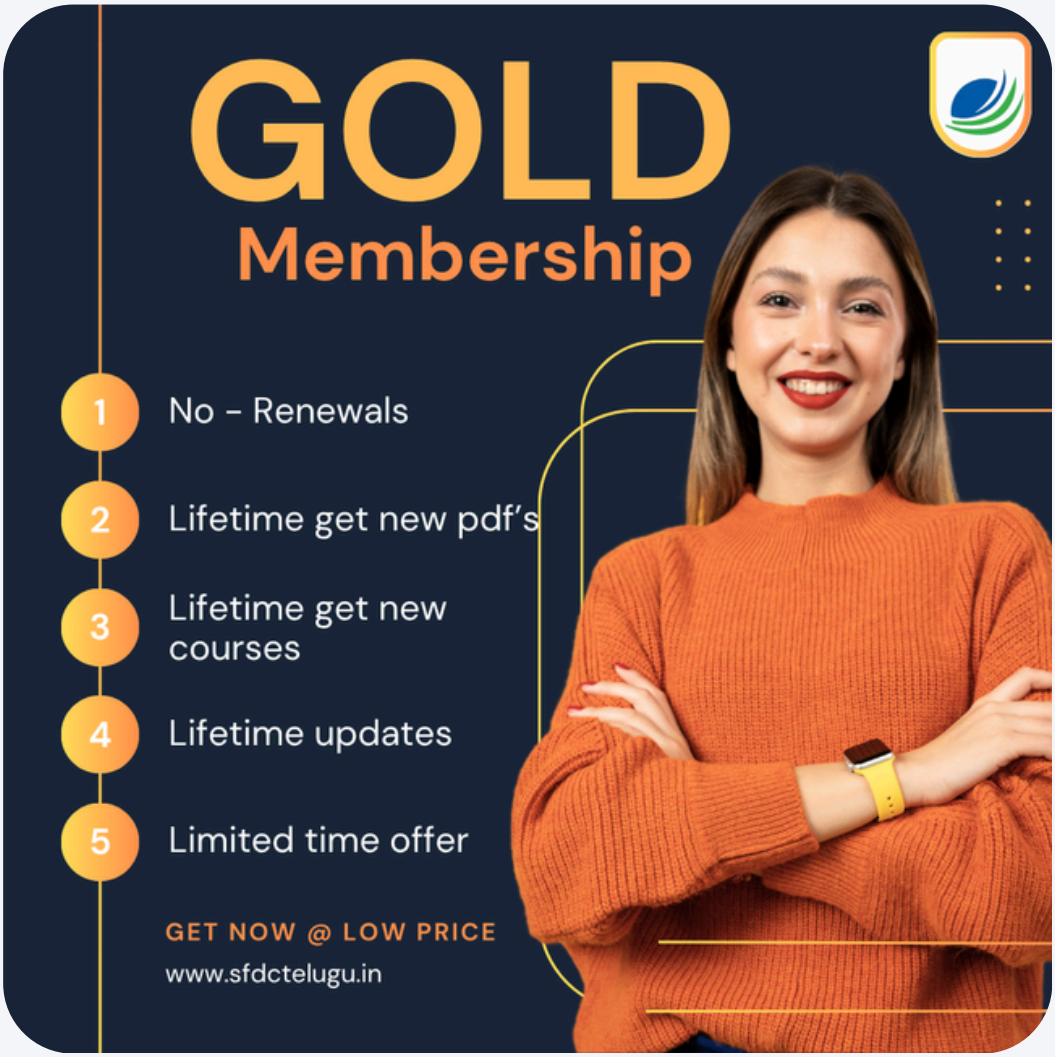
Get Access Now

www.sfdctelugu.in/shop

VISA MasterCard PayPal RuPay G Pay paytm PhonePe BHIM
OLAMONEY JioMoney freecharge MobiKwik NEFT IMPS NET BANKING BANK TRANSFER

https://www.youtube.com/@sfdctelugu

SFDC STORE



Get Access Now

www.sfdctelugu.in/shop

VISA MasterCard PayPal RuPay G Pay paytm PhonePe BHIM
OLAMONEY JioMoney freecharge MobiKwik NEFT IMPS NET BANKING BANK TRANSFER

https://www.youtube.com/@sfdctelugu