Lightning Web Components (LWC)

Table of Contents:

1. Introduction to Apex in LWC

Lightning Web Components (LWC) are modern, lightweight components built using JavaScript. Often, business logic or data resides in Apex classes on the server side. LWC can interact with Apex in two primary ways:

Ways to Call Apex:

- **@wire**: For reactive, read-only, cacheable data fetching.
- **Imperative**: For user-triggered actions like button clicks, and when modifying data (insert/update/delete).

2. Calling Apex Using @wire

force-app > main > default > classes > AccountController.cls

```apex
public with sharing class AccountController {
    @AuraEnabled(cacheable=true)
    public static List<Account> getAccounts() {
        return [SELECT Id, Name FROM Account LIMIT 10];
    }
}
```

force-app > main > default > lwc > WireExample > JS WireExample.js > ...

```javascript
import { LightningElement, wire } from 'lwc';
import getAccounts from '@salesforce/apex/AccountController.getAccounts';

export default class WireExample extends LightningElement {
    @wire(getAccounts)
    accounts;
}

/*Explanation:
1. The @wire decorator is used to automatically fetch data when the component loads.
2. cacheable=true is required for @wire and indicates that the method is read-only.
3. This method is reactive and updates automatically when reactive parameters change.*/
```

3. Calling Apex Imperatively:

force-app > main > default > classes > ContactManager.cls

```apex
public with sharing class ContactManager {
    @AuraEnabled
    public static Contact createContact(String name, String email) {
        Contact c = new Contact(FirstName = name, LastName = 'User', Email = email);
        insert c;
        return c;
    }
}
```

force-app > main > default > lwc > ImperativeCallExample > JS ImperativeCallExample.js > ...

```javascript
import { LightningElement } from 'lwc';
import createContact from '@salesforce/apex/ContactManager.createContact';

export default class ImperativeCallExample extends LightningElement {
    async handleCreate() {
        try {
            const contact = await createContact({ name: 'John', email: 'john@example.com'
            console.log('Contact created:', contact.Id);
        } catch (error) {
            console.error('Error:', error.body.message);
        }
    }
}

/*Explanation:
1. Imperative calls are made when user interaction is required.
2. You can use async/await or .then() for handling promises.
3. Ideal for DML operations like insert, update, or delete.*/
```

4: Apex Best Practices

✓ Use with sharing to enforce sharing rules.
✓ Use @AuraEnabled(cacheable=true) for read-only methods.
✓ Validate inputs to Apex methods.
✓ Use Security.stripInaccessible() for FLS compliance:

List<Contact> filteredContacts = (List<Contact>)
Security.stripInaccessible(AccessType.READABLE, contacts).getRecords();

✓ Check field access like:

if (!Schema.sObjectType.Contact.fields.Email.isAccessible()) {
throw new AuraHandledException('Access denied to Email field');
}

✓ Use custom exceptions and logging for traceability.
✓ Do not expose internal objects or business logic directly.

5: Basic Interview Questions

Q1: What is @AuraEnabled?
A: It exposes an Apex method to be accessible from Lightning components including LWC.

Q2: When should you use @wire?
A: When you need to fetch read-only, cacheable data reactively.

Q3: What does cacheable=true do?
A: Enables client-side caching and mandates the method to be read-only.

Q4: Can we pass parameters to a @wire method?
A: Yes, using reactive variables with $ syntax.

Q5: Can @wire be used to insert or update data?
A: No, @wire only supports cacheable, read-only methods.

6: Intermediate Interview Questions

Q1: How do you handle exceptions in LWC from Apex?
A: Catch them using try-catch in JS and display error.body.message.

Q2: How can you refresh data fetched using @wire?
A: Use refreshApex() from @salesforce/apex.

Q3: What is the role of @track in LWC?
A: It marks fields as reactive; required for objects or arrays in older LWC.

Q4: Can you pass dynamic parameters to a @wire method?
A: Yes, but the parameter must be reactive (use $).

Q5: How do you debug Apex errors in LWC?
A: Use browser console, and handle exceptions properly in Apex and LWC.

7: Advanced Interview Questions

Q1: How do you enforce FLS in Apex when exposing data to LWC?
A: Use Security.stripInaccessible() or check individual fields using Schema.

Q2: What's the difference between with sharing and without sharing?
A: with sharing respects the user's access to records; without sharing does not.

Q3: Can multiple Apex methods be called in parallel from LWC?
A: Yes, using Promise.all() in imperative logic.

Q4: How do you secure Apex methods exposed to LWC?
A: Validate inputs, apply FLS/CRUD, avoid exposing sensitive data, and use with sharing.

Q5: How does LWC manage reactivity in @wire?
A: It automatically re-runs the wire method when any reactive parameter changes.

8: Summary & Tips

- Use @wire for auto-updating, cacheable reads.
- Use imperative calls for user actions and DML.
- Always validate and secure data in Apex.
- Use try-catch in both Apex and JS to handle errors cleanly.
- Keep Apex logic clean, modular, and testable.

Chart: @wire vs Imperative Apex Call

Feature	@wire	Imperative
Data Flow	Reactive	Manual (user/event-triggered)
Use Case	Read-only, auto-load, reactive	Create/Update/Delete, dynamic
Annotation Required	@AuraEnabled(cacheable=true)	@AuraEnabled
Caching	Yes (client-side)	No
Reactive Parameters	Yes ($param)	Not required
Error Handling	Inside wire function	try-catch block
Refreshable	Yes with refreshApex()	Call again manually
DML Support	❌ No	✅ Yes
Example Use	Load data on page load	Submit form, click to create

Cheatsheet: Decorators & Utilities in LWC

Decorators:

Decorator	Description
@track	Makes a property reactive (older LWC).
@api	Makes a property public (can be passed from parent).
@wire	Connects to Apex or Salesforce data sources reactively.

Common LWC Utilities:

Utility	Usage
refreshApex()	Refresh data from a wired function.
NavigationMixin	Navigate to standard pages or custom pages.
ShowToastEvent	Show toast notifications (success/error/info).
getFieldValue()	Get specific field from a record wire.
getObjectInfo()	Get metadata info about an object.

Thank you