Component

Connected callback

Page

Constructor

Component

Rendered callback

Disconnected callback

**Lifecycle Hook**

**Constructor()**

Whenever salesforce starts creating a component to start running LWC at the backend, 🡪 then one method will be called that is called constructor.

**Connected callback()**

Then that component will be put it on that page🡪 then another method will be called connected callback

**rendered callback()**

Ones that element is inside that page and there are different elements inside that component, ones all those component are rendered inside those component🡪 then the third method will gets called that is rendered callback

**Disconnected callback()**

If we are removing or destroying the component from inside the page🡪 then another method will be called Disconnected callback

**Error callback()**

Error callback method will be called only when we are having an error in the child component.

(while executing a child component, if we are having an error then error callback will be called)

It will always written in parent component.

|  |  |  |
| --- | --- | --- |
| **Wire Setup** | | **Imperative Apex** |
| Constructor | | Constructor |
| Wire(empty object) | | Connectedcallback |
| Connectedcallback | | Render |
| Render | Renderedcallback | Renderedcallback |
| Renderedcallback | Apex returns data |
| Wire(results) | Wire(results) | Render(if dirty data) |
| Render(if dirty data) | Rendered callback(clean data) | Rendered callback |

**Rendered callback():** it is used to perform logic after a component has finished the rendering phase. Child to parent.

**Errorcallback():** it is used to create an error boundry component that captures errors in all descendent components in its tree.

**Question:**

**What is data binding?**

**Solution:** The process of transferring the data or displaying the data from .js to html is called data binding.

//HTML file

<template>

<h1> hello {firstName}</h1>

</template>

//Js file

Import {Lightning } from lwc;

Public static

{

firstName = “Rohan”;

}

**# SLDS**

In order to take input from the user we first need to create a UI and add fields and buttons. We can make these with the help of SLDS.

**SLDS**🡪 SLDS is a salesforce lightning design system which is CSS framework inside salesforce. And which allows developers to create components similar to the salesforce native interface.

**Grid🡪** Let’s say we have a page(or webpage), we divide it into 12 rows and columns. So that we can give a proper position to different element in the page.

|  |
| --- |
| Field 1 |
| Field 2 |
| Button |

|  |  |
| --- | --- |
| Field 1 | Field 2 |
| Button | |

To do this,

**Step 1**: Make a Card

**Step 2**: Make a Grid

**Step 3**: Putting elements in the grid.

**Alignment:** To place it at a position left, center or right

**Padding:** To give space between columns

**Box:** To give a background

**Question:**

**What is Component Library?**

**Solution:** Component library is a set of reusable component that can be imported into our LWC project easily and also saves time and space because it offers the pre-built components that can be customised according to our business/project requirement.

**this** keyword🡪 To access the value of property we use this keyword.

**Decorators:** they are the design patterns that will be adding functionalities into the properties and functions in LWC.

@api 🡪 It makes properties and functions publicly available into the component.

Lets say we have two component one is parent and other is child. If parent component wants to access the something from child component then that has to be put in property of function and will be declared public using @api. Only then parent component will be able to access it.

@track 🡪 It is a decorator that will be making the property and the method as an private and property and that can access within the component only and that can’t be used outside the component

* will be making the property and the method as an private
* can access within the component only
* can’t be used outside the component

Example: If we have two component one is child component and the other is parent component and we want that some property or method of parent must not be access by child and vise-versa, then we use @track.

@wire 🡪It will be used to call the Apex method and with the help of those apex method we can retrieve the data from the database and we show that data into the LWC Component.

Example:

We need top 5 Account records which are having high revenue and they needs to be displayed into the LWC component.

Here ,

Wire decorator call the apex method and that apex method is responsible for returning list of 5 accounts records having high revenue

We need total of three things

1. LWC component 🡪 HighRevenueAccounts

Wire 🡪import this decorator in js

🡪calling getHighRevenueAccountRecords

🡪Whatever method we create in apex is stored in @salesforce/apex

🡪from @salesforce/apex module, we are going to import apex class

* Call the method inside js as @wire(methodname,{parameter1:value1})
* Then specify a property or handler which will be assigned the value/respond whenever that method gets called

|  |  |
| --- | --- |
| * Syntax 1(Best practice) | * Systax 2 |
| * @wire(methodname) * Handler(response){ * } | * @wire(methodName) * propertyName; |

1. Apex class 🡪 AccountController

Method🡪 getHighRevenueAccountRecords

1. Lightning Component tab🡪 It will display LWC Component

**# JS FILE**

import { LightningElement, wire } from 'lwc';

import getHighRevenueAccountRecords from '@salesforce/apex/AccountController.getHighRevenueAccountRecords';

export default class HighReveneAccounts extends LightningElement {

    @wire(getHighRevenueAccountRecords)

    getAccountsHandler(response){

    }

}

So basically, whenever this wire is going to call this apex method, we are going to get a response, and with that response this getAccountHandler method/handler gets called. This particular handler will be having that response which consist of error and data. Because there is a possibility of getting an error in apex method so there is an error property.

Case 1: If there is no error there error property will will be undefined

Case 2: If there is error then data value will be undefined.

**@AuraEnabled🡪**

**(cacheable=true)🡪** It is used so that the record that apex class is returning is stored in cache memory so that the speed of feching data will get reduced and whenever we need the data, the data will simply come from the cache memory itself.

In @wire case 🡪 It works in realtime data so we needs to add (cacheable=true) so that server keep getting the data in real time.

**Note:** Don’t Use in case you are performing DML operation as it works in realtime we can get the error Too many DML statement. To avoid this don’t use (cacheable=true) in case you performing DML.

In Imperative call case 🡪 It works based on user interaction, so we can use (cacheable= true) here, so that whenever user interact data will be provided.

|  |  |
| --- | --- |
| Wire Method | Imperarive method |
| Apex method get called automatically only in two situations:   * Whenever page loads. * Whenever the value of the property which is passed in wire changes. | We fetch data and can perform actions by calling method when and where we want.   * Page load * Button clicks * Any user interaction |