**LWC**

1. **async** makes a function return a Promise
2. **await** makes a function wait for a Promise

**Note** - The await keyword can only be used inside an async function. The await keyword makes the function pause the execution and wait for a resolved promise before it continues:

let value = await promise;

**Embed VF page in LWC** –

1. Visualforce pages and Lightning Components are served from different domains.
2. We will be using window.postMessage() for pass data.
3. Use [iframe](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe) to embed the VF page.

**what is bubbling in lwc –**

In LWC we have two flags which determines the behavior of event in event bubbling phase. bubbles A Boolean value indicating whether the event bubbles up through the DOM or not. Defaults to false. composed A Boolean value indicating whether the event can pass through the shadow boundary.

**How to Use Static Resources In LWC –**

1. First we need to create static resource from quick find box and then import

Import static resources from the **@salesforce/resourceUrl** scoped module. Static resources can be archives (such as .zip and .jar files), images, style sheets, JavaScript, and other files.

|  |
| --- |
| import myResource from '@salesforce/resourceUrl/resourceReference'; |

When static resource has namespace

|  |  |
| --- | --- |
| 1 | import myResource from '@salesforce/resourceUrl/namespace\_\_resourceReference'; |

**what is lightning message services in LWC- In** LWC we will be using LMS when we want to send data from one LWC to another LWC which are not in the hierarchy.(means there is no parent – child relationship between them)

**pubsub in lwc -----** Pub-Sub Model is essentially used to transfer the information between the component which are not in the same DOM hierarchy — which means that the components are not connected/ related in any way with each other using the parent-child relationship.

**Non-primitive data type**- Non-Primitive data types are classified as Array, Class, String, and Interface.

**Queueable -**

* queueable---for queuebale we need to use queueable interface and after qneue the job we get job id
* how to enqueue job by queuable - System.enqueueJob(new className());
* ID jobID = System.enqueueJob(new className());
* we pass data in queueable by using----constructor
* You can monitor the status of your job programmatically by querying AsyncApexJob or through the user interface in Setup by entering Apex Jobs in the Quick Find box, then selecting Apex Jobs.
* AsyncApexJob jobInfo = [SELECT Status,NumberOfErrors FROM AsyncApexJob WHERE Id=:jobID];

**connected callback** -The connected callback is executed when the component is inserted into DOM. Connected callback runs once when the component is inserted. The execution flow of connectedCallback is parent to child. So you cannot access child elements in the connectedCallback, because they are not inserted yet.

**renderedcallback-**This gets called when the component is rendered. Basically when all the elements on the component are inserted. And it is run from child to parent so we can access child elements. Imperative and wire services

**Imperative---**we need to use Imperative when Trigger a callout by virtue of a user driven event ie. click of a button, etc. Consider the case when you do NOT want an apex method to be called when the component is loaded. In order to invoke the apex method on click of a button, or in response to a certain logic, we could invoke the apex method in an imperative fashion.

**wire services ---** when we want to call apex when components is load or on change of a value of a parameter, we can't write services inside an function

**Query selector-----**In LWC this.template.querySelector is used to select an element, Here element can be div, span, input or any tag inside lwc component.

In querySelector we use classname to uniquely identify a particular element and if there are more than one element we use querySelectorAll and run a loop on it.

**inbound and outbound integration----**

outbound----in outbound integration we need to use remote site settings to store URL in org or name credentials in our org and in this we send request from salesforce to third party and get response from third party .

inbound ----in inbound integration we get request and from our side we send response to third party for inbound integration we need more security so to this we need to created connected app with third party.

**How to access dom element ---** The standard way to access elements in the DOM is to use querySelector(). To locate shadow DOM nodes, use querySelector() or querySelectorAll() on this.template. For light DOM nodes, use one of those methods on this.

// shadow DOM

this.template.querySelector('div');

// light DOM

this.querySelector('div');

**deep clone in js ----** deep clone on the value by doing JSON.parse(JSON.stringify(value));

**Data mutation in lwc(data transfer )------**In LWC data mutation is one directional if we change anything in parent the it will automtically reflect in child but if we change anything in child it will not reflect in parent for new value in parent we need to pass an event from child to parent.

**Data mutation in aura (data transfer )------**In aura data mutation is by directional if we change anything in parent the it will automatically reflect in child but if we change anything in child it will also automatically reflect in parent

**why LWC is better the aura –**

1. Aura-based Lightning components are built using both HTML and JavaScript, but LWC is built directly on the Web stack.

2. Because of the absence of an abstraction layer, LWC is likely to load and run much faster than Aura components. It is also lightweight and memory-efficient as it is built on Web Components.

3. As there is no additional framework, LWC developers can make transitions much easier.

**difference between aura and lwc----**

Bundle structure: LWC requires the manual creation of a folder to host all your component files. Component HTML, JS, and Configuration files are mandatory. CSS and SVG are optional.

Naming Convention: The only noted difference in naming is while you’re referring component in another component. The new syntax uses kebab-case, instead of the camelCase ( <c-hellowrold> instead of <c:helloworld>) and the component must have a closing tag.

Events: Unlike component or application events in Lightning Component, LWC used Standard DOM events.

Lightning Locker: Lightning Locker is enabled by default in all custom LWC.

**Life cycle hook in lwc -**

constructor called!! wire called! connectedCallback called!! Getter called!! renderedCallback called!! wire called when data/error received