- learnsalesforce@madhu

WHAT IS LWC???

Lightning Web Component (LWC) is a new programming model developed by Salesforce. It is used to develop robust and responsive single-page UI based applications for mobile and desktop.

The new model co-exists with the Aura Components model and delivers unparalleled performance. The UI framework is built using native HTML and modern JavaScript.

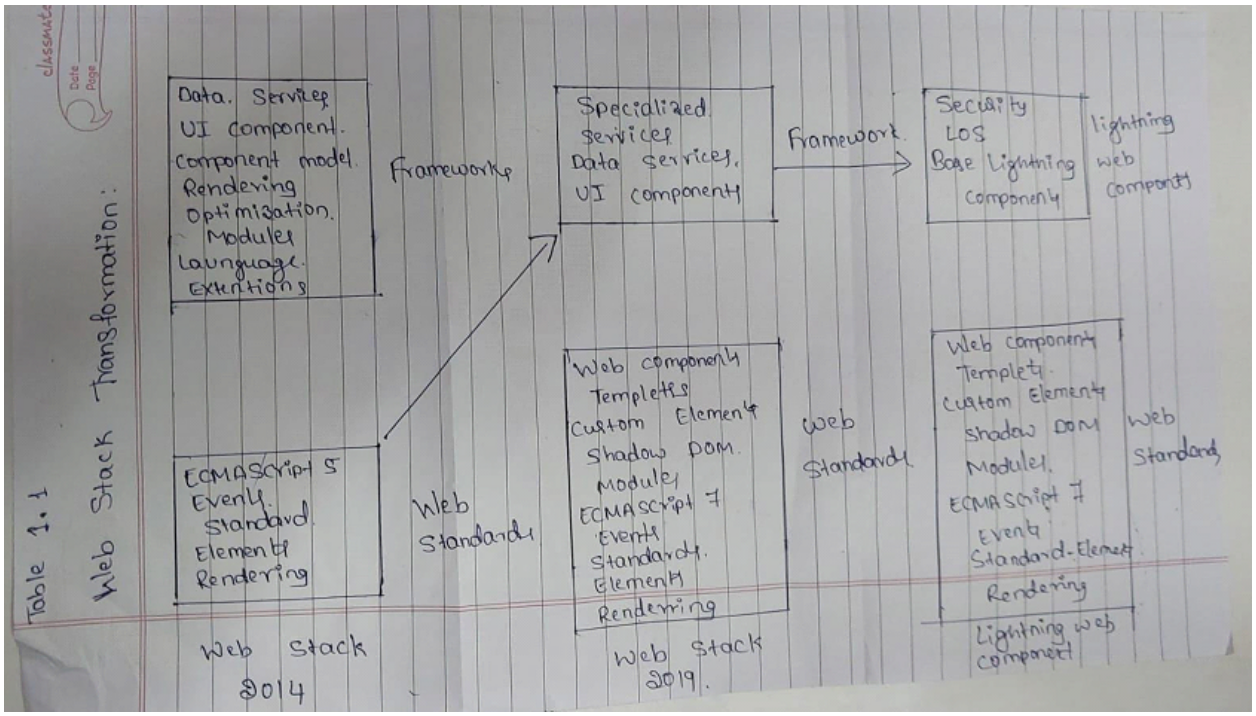
It uses core web component standards and uses custom elements like templates, modules, shadow DOM, and other new language constructs of ECMAScript7.

Moreover, development using LWC is easy compared to AURA as developers knowing HTML and JavaScript can easily code in it.

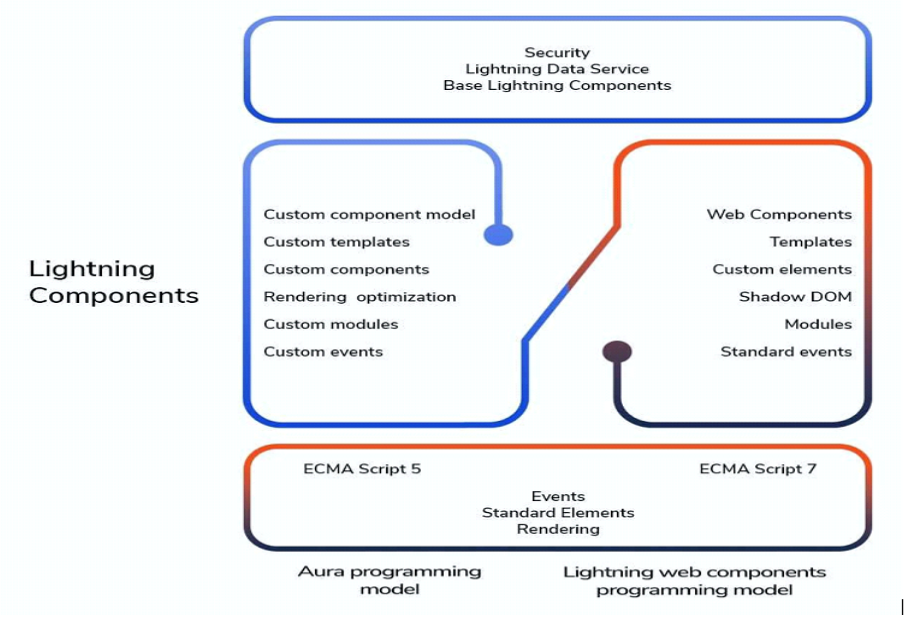
If you want to develop single-page web applications using LWC, hire Salesforce developers who have in-depth knowledge about it and can fulfill your business requirements.

WHY LWC???

We need to understand the evolution of the web stack in the last few years.



The new development stack will look like this:



In 2014, web standards only have ECMAscript 5, Events, standard elements and rendering only and the key elements were not part of web standards so web standards were not strong enough to create UI Components so several frameworks were there to create UI components such as Angular, React JS, Angular etc D

Benefits of LWC Over Aura components :-

1. Better performance - render faster than aura components.

2. More standards, less proprietary.

3. Easy to Learn 4. Faster Loading

5. Easier to ramp for developers.

Coexistence and Interoperability

1. Aura components and LWC can co-exist on the same

page.

2. Aura components can include LWC components whereas LWC components cant have Aura components.

3. Aura components and LWC share the same base lightning components and share the same underlying services like lightning data service, User Interface API etc.

What are web components?-

Web component is a suite of different technologies allowing you to create reusable custom elements.

Custom Elements :- Custom elements are developer defined HTML elements which works with other standard HTML elements.

HTML Template :- Earlier we used the same markup structures repeatedly on a web page, but now we can use some kind of template rather than repeating the same structure over and over again. The <template> element enables you to write markup

templates. These can be reused multiple times as the basis of a custom element structure.

Shadow DOM :- A set of Javascript APIs for attaching an encapsulated "shadow" DOM tree to an element which is rendered separately from the main document DOM. In this way, you can keep an element's features private.

ES Modules :- ES modules specification defines the inclusion and reuse of JS documents in a standard and modular way.

Let's start development:-

1. Select Ctrl+shift+P in VS code to open command palette 2. Create Project with Manifest - Standard Project

Template - specify the name of the project as

LWCPractice1 3. Now authenticate your org, from the command palette

- Authorize a org - project default - provide alias name - click on Enter

4. In force-app-classes Ctrl+shift+p Create Apex class - SFDCHello - Enter Location - Right Click on Class Name - Deploy

source to Org

5. In the salesforce org, do the changes in the SFDCHello class. 6. Right click on class Name - Retrieve source from org.

Naming Conventions of Lightning Web component:-

1. Must begin with a lowercase letter.

2. Must contain only alphanumeric or underscore characters.

3. Must be unique

4. Can't include whitespaces.

5. Can't end with underscore

6. Can't contain two consecutive underscores.

7. Can't contain a hyphen.

Render List :

render() Render is mainly used to conditionally render a template. It defines the business logic to decide which template (HTML file) to use. Enter the render() method. In LWC this method doesn't do the actual rendering but determines what template to use to render the component.
To render a list of items, use for: each directive or the iterator directive to iterate over an array. Add the directive to a nested <template> tag that encloses the HTML elements you want to repeat.
To traverse over the list => use for:each directive

For:each -> used to specify the list to be iterated
For:item -> to access current element
For:index -> to access the index of the current element

Eg .
<template>
 <template for:each={list} for:item='varName' for:index="he">
 varName.property
 </template>
<template>

Rendering List with Iterator :

Whenever we use for:each or Iterator we need to use a key directive on the element on which we are doing iteration. Key gives a unique id to each item. Remember, without a key, we cannot do iteration.
To apply spl behaviors like first/last element to the list then we should use iterator over for each loop
iterator:iteratorName={list} //iterator name can be anything

Component Composition :

You can add components within the body of another component.
Composition enables you to build complex components from simpler building-block components.
Composition : Let's consider an example of Car, we all know Car is a product which consists of a car wheel, car body, car glasses etc. Now they all are manufactured or created in a different company.
Similarly, In our lightning application, we create such small-small reusable components separately and install them (compose them) inside another component. This concept is called component composition.
Now, the obvious question is why to use other components inside the main component ?
So there are many advantages to this Like:
- No need to re-write the code again.
- Easy to call other components.
- Reduce code size.

Now let's understand how we can create a composition in the lightning web component. Basic Idea is we create the main component (parent component) and call other reusable child components inside main component.
A basic structure is like this :

```
----------------------------------------
< Parent Component >
 <Child Component 1>
 <Child Component 2>
 .....
</Parent Component >
----------------------------------------
```

As you can see in the above structure how we can include the child components in the parent component.
Now's let understand the syntax in the Lightning web component :
SYNTAX :
parentComponent.html

```
----------------------------------------
<template>
 <p>I am inside Parent Component</p>
 <br/>
<!-- below is child component-->
<c-child-component></c-child-component>
```

</template>

In the above syntax as you can see we have just included a couple of words inside a paragraph tag.
Then below that, we have included the child component inside as a markup.
IMPORTANT: When we call child component inside the parent component We use the kebab case to represent the name of it.
Hence as our component name is childComponent, we have written it starting with <c- this is a default namespace. After this, If we use camel case like oneTwo we write it in kebab case format as one-two.
Hence we called the child component as <c-child-component> in the above parent syntax.

And finally, when we preview the above parent component we can see both parent and child component data together as we composed them together.
This is how we do component composition in salesforce lightning web component.

It was simple right ??
Now let's create something creative composition of the components like this

Component Composition In Salesforce Lightning Web Component (LWC)

Excited, right ??

They are moving too 😁. So In the above snap, you can see there are three characters: Bat (Spreading Virus), Doctor who is saving Child from the bat spreading virus.
I picked up this idea as currently, Coronavirus is trending nowadays hence I would like to demonstrate with this simple example of how doctors are fighting with the virus and saving us.

So let's begin.....

COMPONENT 1 : So We are going to create the main component which consists of some titles and characters of doctor and bat.

COMPONENT 2 : In the second component we are going to include the small kid walking. So that when we preview the Component 1 then we can see three of the characters on the same screen. I hope the idea is clear now for this composition.

Custom Label : are text values that we can access in LWC ,aura ,apex code , VF pages Are useful for multilingual application …to change sentence into different languages How to create custom labels in salesforce : Setup->quick find box->custom label->new custom label.

You can create up to 5,000 custom labels, and each label can have up to 1,000 characters.

Static resources : allows you to upload content which we can use in VF pages ,aura classes,

These resources can be JS,CSS,(archieves)zip jar.

First, 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.

The syntax for importing static resource in LWC
1.
import myResource from '@salesforce/resourceUrl/resourceReference'; When static resource has namespace

Static resources can be archives (such as . zip and . jar files), images, style sheets, JavaScript, and other files.

Life cycle Hooks:

A lifecycle hook is a callback method triggered at a specific phase of a component instance's lifecycle.
constructor() Called when the component is created. This hook flows from parent to child, which means that it fires in the parent first. You can't access child elements because they don't exist yet.

constructor () -> no public property is available , It is called once in the whole life cycle of component … just like init method -> after every refresh of the component this constructor will call.. If you define a constructor then there must be a call to its

superclass by using super() keyword.. It must be the first statement in the js file of ur lwc…

connectedCallback() -> is called when a component is added to the DOM. public properties are available here as constructor is called.components may not be visible completely .

renderedCallback() -> component is rendered to the browser.. This method is called multiple times (means if we change anything on component like giving input in box or clicking some button or any action on component ..then renderedCallback() method will be called),depending upon your business logic

disconnectedCallback()-> component removed from dom Render() -> is used to override the default rendering

errorCallback() ->

Navigation in LWC :

Note: We can see output in org only not in app(developer console) as they are referring to standard pages of org Use the navigation service, lightning/navigation, to navigate to many different page types, like records, list views, and objects. Also use the navigation service to open files.

Step 1: import { NavigationMixin } from 'lightning/navigation';

Step 2 : export default class MyCustomElement extends NavigationMixin(LightningElement) {}

Step 3 : Create a plain JavaScript PageReference object that defines the page.

Step 4: To dispatch the navigation request, call the navigation service’s [NavigationMixin.Navigate](pageReferenceObject.

HOW TO ACCESS APEX CLASS USING LWC:

There are two ways to call Apex method from Lightning Web Component: Call apex method
Using Wire services.
Call the apex method imperatively.

Using Wire method:

To call the apex method in the lightning web component, First, we have to create the apex class and add the @AuraEnabled method at the first line, i.e., before starting the method. To call it from Wire Service, the method should be cacheable. Hence, add cacheable=true in @AuraEnabled. And in the LWC js controller, we need to write the import method using the @salesforce/apex/className.MethodName; at the start of the js controller and in the lightning, we need to write the @wire to get the records we provide in the apex class.

step1: we have to create the apex class

step2 : add the @AuraEnabled method at the first line, i.e., before starting the method.

step3: To call it from Wire Service, the method should be cacheable. Hence, add cacheable=true in @AuraEnabled.

step4: And in the LWC js controller, we need to write the import method using the @salesforce/apex/className.MethodName.

step5: at the start of the js controller and in the lightning, we need to write the @wire to get the records we provide in the apex class.

Using Imperatively method:

Calling the apex method in the lightning web component without using the wire method is more accessible than the wire method. Making the AuraEnabled method cacheable is not mandatory to call it imperatively. While using the wire method, we need to add the data, but there is no need for this in the imperative method. It is the easiest way to call the apex method in the LWC.

When we have combo box -> onchange event -> = Text box -> button -> onclick -> = Text box ->onchange event => like operator

Communication between components allows you to exchange the data between components

- Communication from parent component to child component.
- Communication from child component to parent component.

- Communication between independent components.

Communication from parent component to child component:

To pass data to the child component, we need to define a variable with @api decorator in the child component to be public and accessed from the parent component. Note: @api decorator used to define public property, which is reactive.

Communication from child component to parent component:

Update a Public Property The @api decorator in the child component exposes a property, making it public, so that the parent component can update it. Import the api decorator from the lwc module. Save the file.

Communication between independent components:

Publish-Subscribe (pubsub) Model in LWC is used to make communication between two independent components. Please note that the Publish-Subscribe Model only works for components that are on the same page.

Lightning Data Services to access Salesforce data :

It is a cache framework . Using this framework we can fetch data using base components.
It simply creates,update or view a record.

These components are built on the top of the UI Api. same ui is used to build This framework shows only the records which currently logged in (all the security settings are applicable)
Here u will not create any apex class , how u can make that class with sharing option available..?

Communicate between two Components:

1. Child to Parent Communication
2. Parent to Child Communication
3. Two unrelated components Communication

1.Parent to Child Communication

As the name suggests, Parent to Child communication happens between two components when the parent component contains the child component tag in its HTML file and passes data to the child component. To pass data to the child component, we need to define a variable with @api decorator in the child component to be public and accessed from the parent component.

Note: @api decorator used to define public property, which is reactive.

2.Child to Parent Communication

As we have seen, passing a public property from a parent and receiving it from the child is the easiest way to achieve Parent to Child communication. In the case of Child to Parent communication, it is a bit more complicated. From the child component, we will pass the value to the parent component using CustomEvent.

Note: The CustomEvent constructor has one required parameter: a string, which refers to the event type.

Lightning web components dispatch standard events, components can also dispatch custom events in order to exchange data between components.. We can ALSO use event listeners …

3. Communication between two unrelated components

- Create a Lightning Message Channel.
- Create the Publisher Component.
- Create the Subscriber Component.
- Add the New Components to the Event Comms App.
- Resources.

Communication across the DOM

There are two ways to communicate between components that aren't in the same DOM tree. Use Lightning message service (lightning/messageService), or use a singleton library that follows the publish-subscribe pattern.

lwc <-> aura<-> vf <->lwc

communication between lwc and aura component:

find(“lwcComp”). receiveData(stringToSend) line in the passDataToLWC method in the Aura Components Javascript controller. This is finding the lwc that we imported into our Aura Component by its aura:id and then calling the receiveData method in the LWC (the method with the @api decorator) and passing it data.

communication between lwc and VF component:

If multiple visualforce pages are included in a lightning component, then you need to get the window object of every visualforce page and send the message to each window object by calling window. postMessage().

Lightning Message Service Limitations:

Keep the following in mind when working with Lightning message service. Supported Experiences Lightning message service supports only the following experiences

Lightning Experience standard navigation
Lightning Experience console navigation
Salesforce mobile app for Aura and Lightning Web Components, but not for Visualforce pages
Lightning components used in Experience Builder sites. Support for Experience Builder sites is beta.

Madhu Kumara M

Salesforce Guy