Day 11 :

Topic: Lightning Web Components

Why Lightning Web Components(LWC)?

AURA framework which was used for current Lightning Components was based on standards of 2014 but are outdated now and it was time for change because for the following reasons:

- Rendering could be optimized.
- Standard UI elements were scarce.
- Lacked modern constructs.
- Was not fit for modular apps.
- Web standards were updated.
- AURA Framework became skill and had its own learning curve.

Additionally, Lightning Web Components(LWC) can coexist and interoperate with Aura components.

What is Lightning Web Components(LWC)?

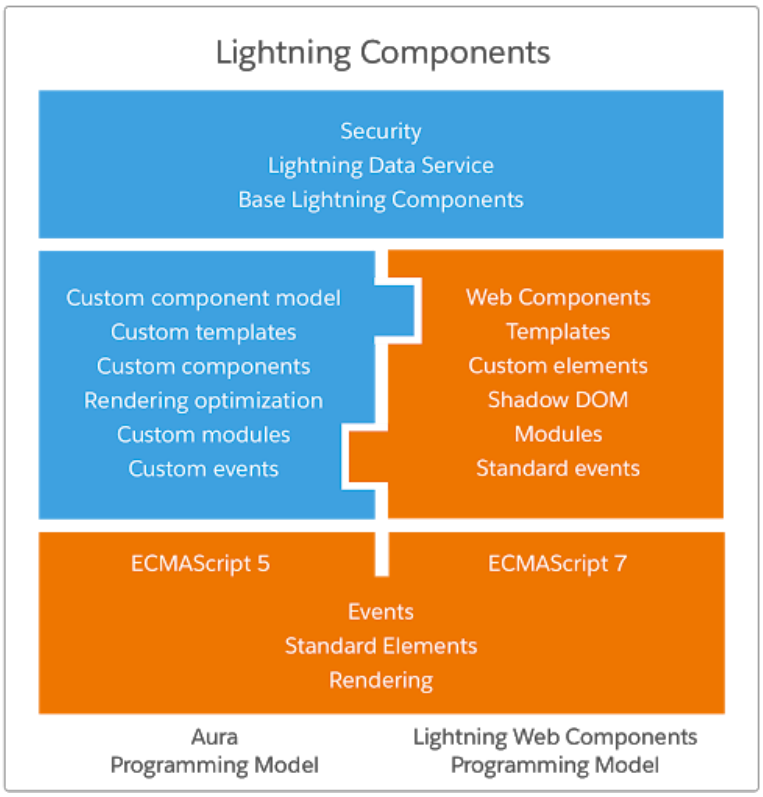
LWC is a new programming model levering the recent web standards. Rather than being a totally custom and development wise rigid framework, It's quite flexible. It's mostly the common Web Standards and a Thin Layer of Specialized services to make it a perfect fit for Modern Rich UI Implementations in Salesforce. This thin layer of specialized services contain Base Lightning Components, Lightning Data Service and User Interface API which work behind the curtain for LWC.

A thin layer of specialized services on top of a standard web stack results in:

- Ease of development for large scale modular apps.

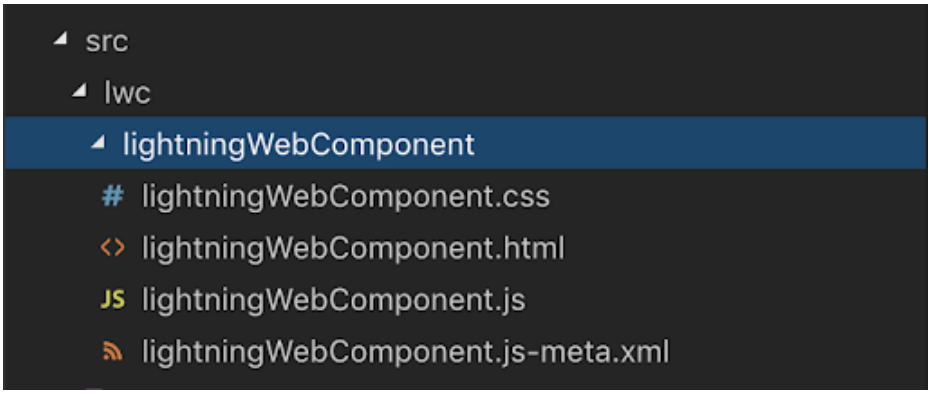
- Ease of Leveraging the latest web functionalities and constructs.
- A common model and transferable skills.
  (Any web developer working on modern JS frameworks could easily ramp-up LWC).
- Interoperable components.
- Better performance.

So, the new development stack looks like:



How is a Lightning Web Component(LWC) formed?

Similar to an AURA component, the main contents of a LWC are also html, javascript. There are optional content like css. But then in addition to these for LWC, an xml configuration file is also included which defines the metadata values for the component.



HTML

- Has a root tag <template> which contains your component's HTML.
- When renders, the <template> tag is replaced with <namespace-component-name>.

Javascript

- Import functionality declared in a module eg-lwc(the core module), use the import statement.
- To allow other code to use functionality in a module, use the export statement.
- LightningElement is custom wrapper of the standard HTML element and we extend it in the component and export.

```
JS lightningWebComponent.js ×
1  import { LightningElement, track } from 'lwc';
2
3  export default class lightningWebComponent extends LightningElement {
4      @track message;
5
6      handleChange(event) {
7          this.message = event.target.value;
8      }
9  }
```

Configuration

XML file that defines the metadata configuration values for the component eg-

- Components Label
- Availability
- Configuration Attributes
- Builder Attributes

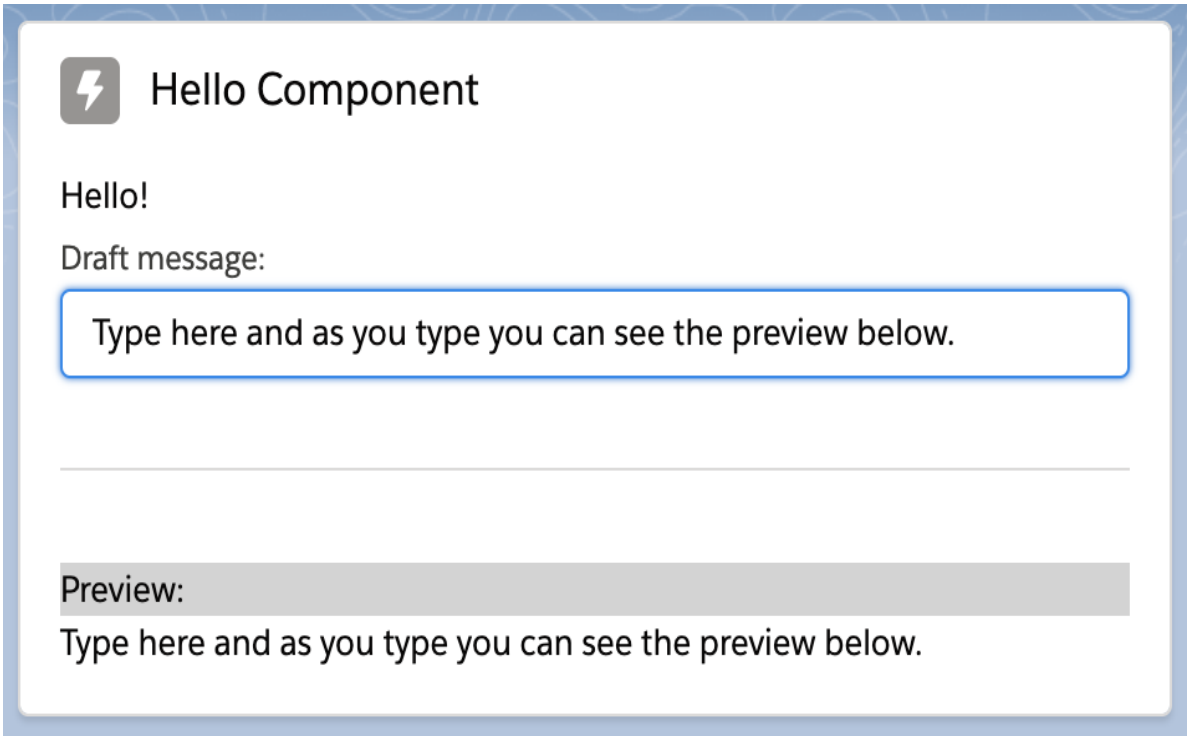
```
lightningWebComponent.js-meta.xml ×
1  <?xml version="1.0" encoding="UTF-8"?>
2  <LightningComponentBundle xmlns="urn:metadata.tooling.soap.sforce.com" fqn="LightningWebComponent">
3      <apiVersion>45.0</apiVersion>
4      <isExposed>true</isExposed>
5      <targets>
6          <target>lightning__AppPage</target>
7          <target>lightning__RecordPage</target>
8          <target>lightning__HomePage</target>
9      </targets>
10 </LightningComponentBundle>
```

CSS

- To style a component.
- The style sheet is applied automatically

```
# lightningWebComponent.css ×
1  .background-grey {
2      background-color: lightgrey;
3  }
```

Component UI


Hello Component
Hello!
Draft message:
Type here and as you type you can see the preview below.
Preview:
Type here and as you type you can see the preview below.