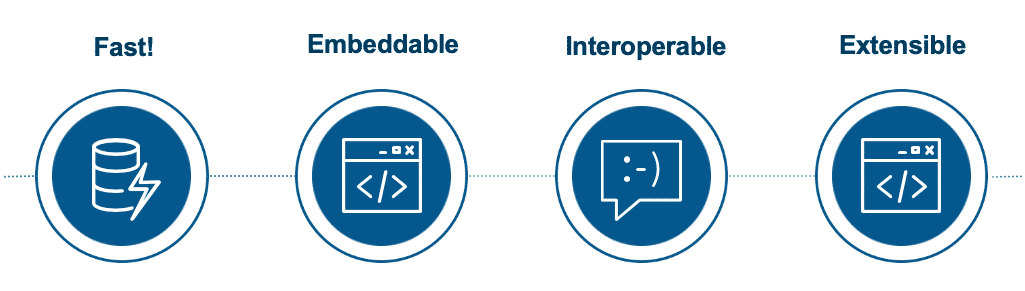
**Digital Commerce Lightning Web Components Overview**

**What are Lightning Web Components?**

Web components utilize the latest in web standards while freeing you from Javascript framework abstractions. However, when you are building a completely on-platform Digital Commerce experience, you will want to utilize Lightning web components.  
  
**Lightning Web Components (LWC)** is the programming model for web components that run inside Salesforce.  This model implements the web component standard but uses select Salesforce libraries and packaging conventions. **Lightning** refers to a Salesforce framework that uses Javascript in the browser rather than server-side code to render the UI.



Lightning web components are the newest components available for on-platform experiences, and have many strengths—they have fast performance, are easily embedded in pages, and have deep extensibility.

Lightning web components provide only what’s necessary to perform well in browsers supported by Salesforce. Because they're built on code that runs natively in browsers, Lightning web components are lightweight and deliver exceptional performance.

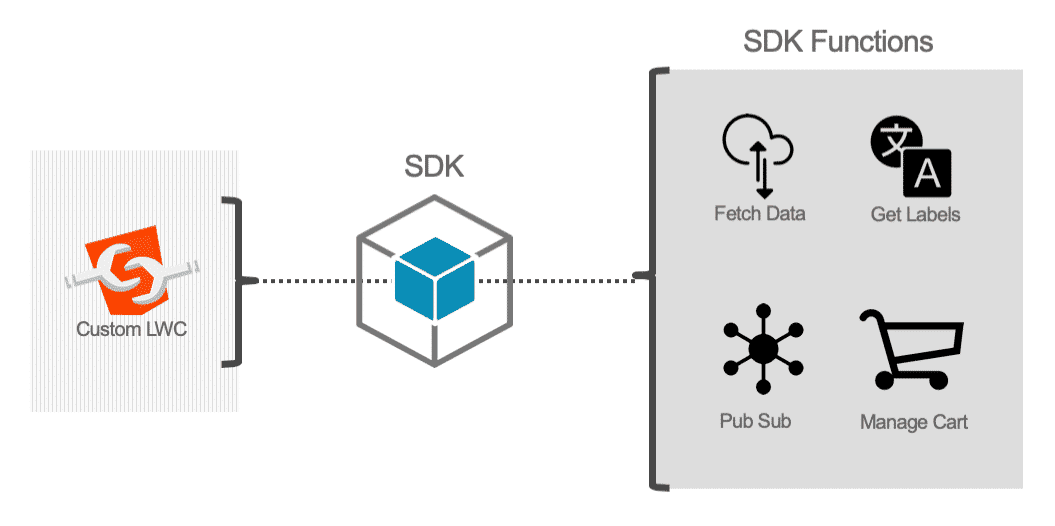
**What are Digital Commerce Lightning Web Components?**

**Digital Commerce Lightning web components**refer to Lightning web components that have been specifically created to build Digital Commerce experiences (e.g., dcCatalog to render a given catalog list). Salesforce has created many Digital Commerce Lightning web components that can be used and extended, and are available within the managed package.

**Digital Commerce Lightning Web Components and SDK**

Using the Salesforce LWC standard, a library of Digital Commerce Lightning web components makes developing digital commerce applications faster and easier. Digital Commerce Lighting web components utilize the Digital Commerce SDK to perform a variety of functions. By using Digital Commerce Lightning web components and the Digital Commerce SDK together, you will be able to achieve a wealth of functionality and extensibility that is reusable, safe, and maintainable.

For example, you can put custom JavaScript functions in an extended Digital Commerce Lightning web component, and it will use the SDK to do things like deleting products from the cart.



**Setting Up A Development Environment**

To manage and develop Industries Lightning web components, use **Salesforce DX** to fetch and deploy to your Salesforce org. Both development environments have source control.

* To manage and develop your Digital Commerce Lightning web components with Salesforce DX, see [Set Up Your Development Environment](https://trailhead.salesforce.com/en/content/learn/projects/quick-start-lightning-web-components/set-up-salesforce-dx).
  + When you're ready to deploy your Digital Commerce Lightning web components, follow the instructions [here](https://docs.vlocity.com/en/Deploying-Lightning-Web-Components.html).

We recommend using [Visual Studio Code](https://code.visualstudio.com/) as a free, cross-platform, and feature-rich code editor. Additionally, it has an integrated terminal.

The Digital Commerce Sample App

The Digital Commerce sample app is built using Digital Commerce Lightning Web Components (LWC) and allows customers to browse and configure products as well as convert their basket to a Salesforce Order by providing the ability to log in to CRM as an identified user (Account), call a payment gateway system, and submit their order.  
  
This sample app is intended to be run on-platform.

**Testing and Configuring Digital Commerce Lightning Web Components with OmniScript Designer**

You should consider using Digital Commerce Lightning web components with OmniScript Designer if you are looking to:

* Test your Digital Commerce Lightning web components
* Create a guided selling experience for users.

Note that OmniScript implementations do not have deep-linking Search Engine Optimization (SEO). Deep-linking is an SEO method where page URLs link to specific content pages (e.g., product pages), rather than to the homepage. Search engines are then able to index those pages individually thereby increasing SEO.

**Digital Commerce Lightning Web Components Checkout OmniScript**

Order submission is not limited to the Cart-based APIs. It is possible using the full Digital Commerce stack, including Digital Commerce APIs, SDK, web components, and Lightning web components.  
  
The**Digital Commerce LWC OS Checkout** integrates the checkout experience into the **Digital Commerce LWC OS Sample App**.

**Troubleshooting and Customizing Digital Commerce Lightning Web Components**

In this lesson, we're going to cover how you can troubleshoot and debug Digital Commerce Lightning web components at runtime. You will be able to see how the SDK calls and responses are processed, as well as the corresponding API call.

**Adding custom fields to the SDK response**

What if you want to work with fields that are not included in the SDK response by default, like DisplayText\_\_c? To address this, we provide [PreHook](https://docs.vlocity.com/devdocs/lwc/DCOffersList.html" \l ".fetchOffersPreHook" \t "_blank) methods to retrieve custom fields. Let's look at what the code would look like if we were to extend **dcOffersList** and use the **fetchOffersPreHook** method to get the Display Text field in our response:

Code for our customOffersList Lightning web component:

import { LightningElement, api, track } from 'lwc';

import dcOffersList from 'vlocity\_cmt/dcOffersList';

import customOffersListTemplate from './customOffersList.html';

export default class CustomOffersList extends dcOffersList {

// connectedCallback() {

// super.connectedCallback();

// }

fetchOffersPreHook(input) {

input.customFields = ["DisplayText\_\_c"];

return Promise.resolve(input);

}

render() {

return customOffersListTemplate;

}

}

Example of the fetchOffersPreHook method being used by an extended **dcOffersList** component to retrieve the "IsActive" (custom) field:



**Asset-Based Ordering and Global Header Components**

Asset-based Ordering is supported with Digital Commerce Lightning web components. A large part of this functionality is possible thanks to the [VlocityDCMyAccount](https://docs.vlocity.com/devdocs/lwc/DCMyAccount.html" \t "_blank) component. This component allows a user to sign in to their account and view their assets. It uses the [VlocityDCAssetsList](https://docs.vlocity.com/devdocs/lwc/DCAssetsList.html" \t "_blank) component, which puts the user's assets within an array.

**Getting Started with Experience Cloud**

[Experience Cloud](https://help.salesforce.com/articleView?id=networks_resources.htm) is a great place to run your on-platform Digital Commerce experience. Because they are integrated with Salesforce orgs, you can begin using them with your DC Lightning components right away.

With the addition of the DC Lightning web components Checkout OmniScript, the Experience Cloud experience has been updated. This will also require some minor additional configuration of the dcSampleApp HTML and your Experience Cloud site.

**Configuring SEO for Experience Cloud**

Using Digital Commerce Lightning web components on a public Experience Cloud page and wondering how you can drive traffic to your site? Read on!

**Search Engine Optimization** (SEO) can help improve discovery for your Experience Cloud. Search engines like Google™ can index your Experience Cloud so that customers, partners, and guest users can easily discover pages of your site.

In this lesson, we are going to outline the steps you need to take to use a **web analytics tool** to measure site performance and enable your site to be indexable via **meta tags**.

**Web analytics tools** can help you collect and analyze your site traffic. They allow you to set goals and optimize your pages to drive more traffic. Whether your goal is to create a better user experience, optimize for customer behavior and key conversion metrics, or even to measure the effectiveness of your marketing campaigns, these tools can be greatly helpful.

**Meta tags**are keywords that search engines can use to elevate your site to their users. Updating your page titles and page descriptions will also greatly help your site's SEO.



Configuring a Web Analytics Tool for Experience Cloud

For our example, we will use [Google Search Console](https://search.google.com/search-console/about) as our web analytics tool. You should also ensure that your Experience Cloud site is [public](https://help.salesforce.com/articleView?id=community_builder_page_access_settings.htm&type=5) and [active](https://help.salesforce.com/articleView?id=networks_publish.htm&type=5).  
  
Here's how to configure SEO for Experience Cloud:

1. Navigate to your Experience Cloud list by going to your org's **Setup** and searching for "All Communities" in the **Quick Find** search box.
2. Copy your Experience Cloud **URL**.
3. Sign up for [Google Search Console](https://search.google.com/search-console/about).
4. Select the **URL prefix**box.
5. Paste your Experience Cloud URL into the **Enter URL** box.
6. Click on **HTML tag**. Copy the meta tag from the box.
7. Go to **Experience Cloud Builder** for your Experience Cloud site.
8. Click **Settings** and then the **Advanced** side tab.
9. Click **Edit Head Markup**.
10. Paste in the meta tag.
11. **Save** and **Publish** the Experience Cloud site.
12. Navigate to your Experience Cloud site's public URL in your browser.
13. Go back to Google Search Console and click the **Verify** button.
14. Now that your site is verified, it will take about a day for your data to start showing up. Take this time to update your meta tag keywords, page descriptions, and page titles as shown in the video.