**OOB(Out-Of-Business) Flow with custom CSS**

* When you create a file can’t be able to add the custom CSS to the existing flow.
* Check below normal flow with by default SLDS class

<https://hughes--hrdevdpro--c.sandbox.vf.force.com/flow/Sanjay_Flow1/3013K000000GHfXQAW>

* Use LWC or AURA component to use a custom CSS for your flow builder.
* In Salesforce flows, the Out-of-Box (OOB) screens and components are built using standard Salesforce styling, but you can customize the appearance using CSS (Cascading Style Sheets) into normal flow builder. You must use LWC or AURA to customize your CSS.

**OOB(Out-Of-Business) Flow with LWC component CSS**

* Create the LWC component eg:- myLWCComponent its creates a lwc component
* In **myLWCComponent.html** add the below code

**<template>**

**<div class="container">**

**<h1 class="header">Welcome to My LWC Component</h1>**

**<p class="content">This is a sample component in Salesforce Flow.</p>**

**</div>**

**</template>**

* In css file **myLWCComponent.css** add the below code

**.container { background-color: #f2f2f2; padding: 10px; border-radius: 5px;}**

**.header { color: #333333; font-size: 20px;}**

**.content { color: #666666; font-size: 16px;}**

Reference - <https://hughes--hrdevdpro--c.sandbox.vf.force.com/flow/Sanjay_Flow2/3013K000000GHfhQAG>

**OOB(Out-Of-Business) Flow with AURA component CSS**

* Create the Aura component. Eg:- myAuraComponent
* In myAuraComponent.cmp

**<aura:component implements="lightning:availableForFlowScreens">**

**<ltng:require styles="{!join(',', $Resource.hrwd + '/hrwd/CSS/custom.css')}" />**

**<div class="bodyClass myAuraComponent">**

**<h1 class="header">Welcome to My Aura Component</h1>**

**<p class="content">This is a sample aura component in Salesforce Flow. </p>**

**</div>**

**</aura:component>**

**Here if you observe I am using <ltng:require> for to store the static resource files to load css for the aura component**

Reference - <https://hughes--hrdevdpro--c.sandbox.vf.force.com/flow/Sanjay_Flow3/3013K000000GHfsQAG>

**Issues related to CSS leak while you integrate multiple LWC components in single flow**

* Specificity conflicts: Let's say you have the following CSS rules:

**Example:**

**h1 { color: blue; }**

**.page-title { color: red;** }

If you have an <h1> element with the class "page-title," the color specified by the class rule will override the color specified by the element selector, causing a specificity conflict. So, use CSS selectors and give some meaningful class name that you can use it in multiple places so that I wont leak in other places.

* Inheritance issues: Consider the following scenario

**Example:**

**<div class="container">**

**<p>Some text</p>**

**</div>**

**.container {font-size: 18px;}**

**p {font-size: inherit;}**

If the parent container has a specific font size set, the <p> element will inherit that font size, potentially leading to inconsistent font sizes across pages. So please use class selector to make consistent across all pages.

* Selector clashes: Let's assume you have two separate stylesheets, each defining a different background color for a specific element

**Example:**

**<link rel="stylesheet" href="styles1.css">**

**<link rel="stylesheet" href="styles2.css">**

**/\* styles1.css \*/**

**#header {background-color: red;}**

**/\* styles2.css \*/**

**#header {background-color: blue;}**

In this case, the background color of the "header" element will depend on the order in which the stylesheets are loaded, leading to inconsistency. To avoid this make single static resource file use reusable class so that all pages make you as consistent as across all browsers.

* Browser compatibility: Let's say you have the following CSS rule

**Example:**

**.button {**

**border-radius: 5px;**

**}**

In some older versions of Internet Explorer, the border-radius property may not be supported, resulting in inconsistent button styles between browsers. To address this, you can use vendor prefixes or consider alternative ways to achieve rounded corners in older browsers, such as using background images or JavaScript polyfills.

* Responsive design challenges: Consider a scenario where you have a responsive website with a navigation menu that should change its layout and behavior on smaller screens. Here's an example of CSS code using media queries to adjust the menu

**Example:**

**.nav { display: flex; }**

**@media (max-width: 768px) { .nav { display: block; } }**

In this case, the menu is displayed as a horizontal list when the screen width is above 768 pixels, but it switches to a vertical stacked list when the screen width is below that threshold. Ensuring consistent behavior and appearance across different screen sizes can be a challenge, and thorough testing on various devices and viewport sizes is crucial.

* Scalability issues:

As a website grows larger and more complex, maintaining consistent styles across different sections can become challenging. For instance, if you have a large e-commerce website with various product pages, ensuring consistent styling for product listings, descriptions, and reviews can be difficult without a scalable CSS architecture or a component-based approach.

Following these CSS best practices, such as using modular or component-based approaches like BEM, can help prevent CSS leakage between different sections or components of your website. By encapsulating styling within specific components or modules, changes made to one component will not inadvertently affect other components or pages. This ensures that the CSS remains contained and isolated, promoting consistent styling and avoiding any unintended leaks or conflicts, even when working with complex flows that utilize multiple components in your flow builder.

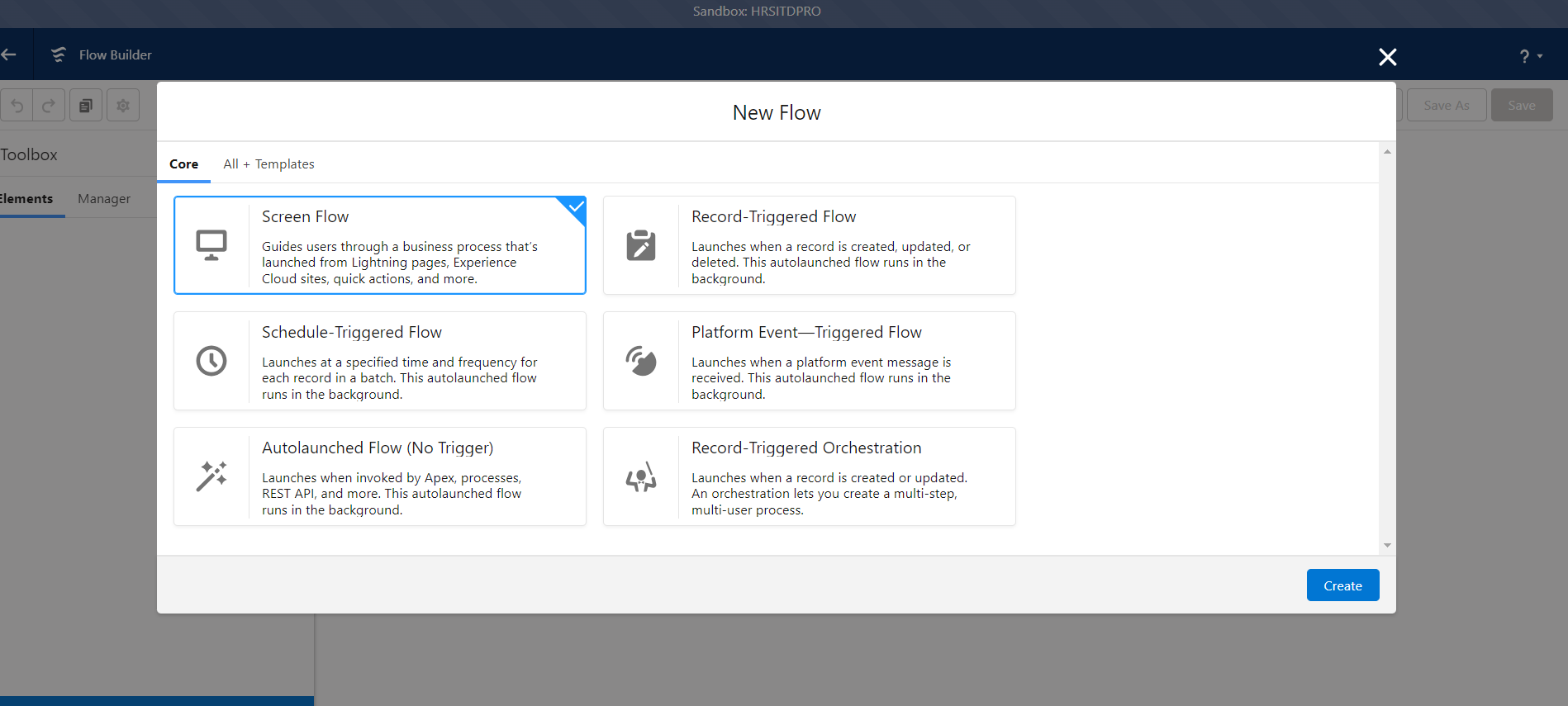
**Integration with existing LWC component in salesforce flows.**

**Step 1:** **Set up the Salesforce Development Environment**

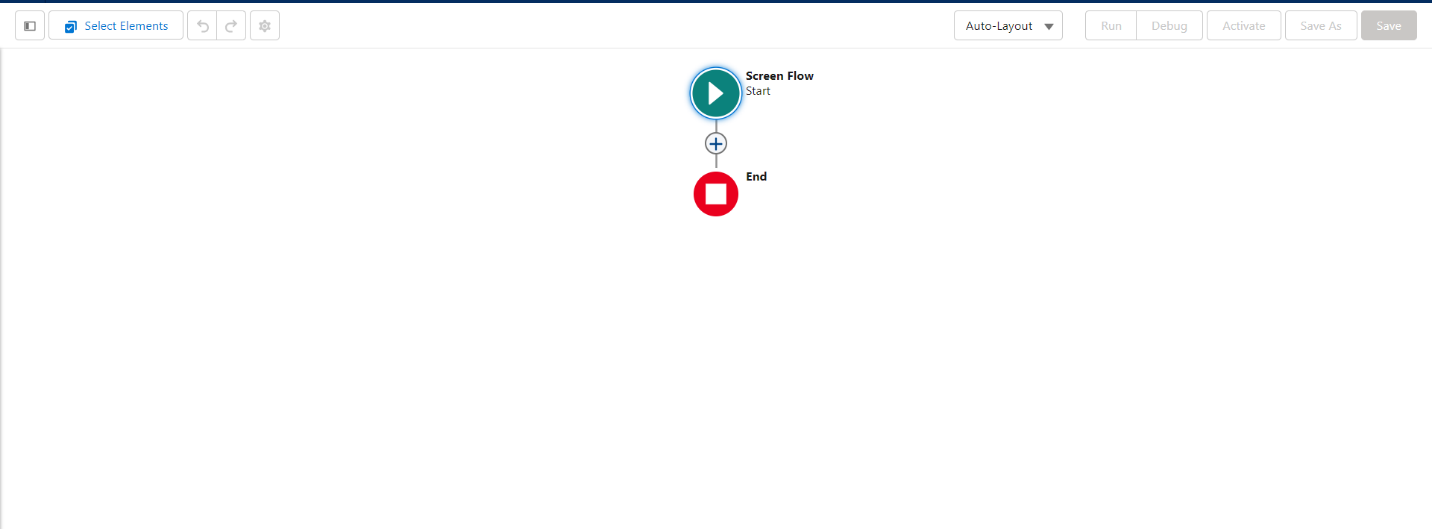
* Make sure you have the Salesforce CLI installed and have set up a Salesforce Developer Org or Sandbox.

**Step 2: Connect LWC Components to Flow**

* Go to your salesforce Org
* Go to setup page
* In quick find search with flows
* In flows page click on New Flow
* Create flow and select the option as screen flow and click on create button so that your flow is ready



* In your flow you can see plus icon click on that and you can add your LWC components to your screen flow



* Once your LWC components added click save and run your flow

**Step 3: Link the Common CSS File**

* Open each of the LWC component files (hrwdLOAAttachments) in a text editor.
* To run this flow what ever @api values you use you have to define it from the flow as well then it can work for your existing LWC component.
* Import the common CSS file by adding the following line at the top of each file

**import { loadStyle } from 'lightning/platformResourceLoader';**

**import COMMON\_CSS from '@salesforce/resourceUrl/commonCss';**

once your import it you must call this in renderCallBack method from all three LWC components

**renderedCallback() {**

**loadStyle(this, COMMON\_CSS)**

**}**

* So, this way we can add the common CSS to all three LWC components, but we cannot add CSS file directly from the flow builder

**Using reusable CSS file in LWC and aura components**

Check the below link which is already used the existing LWC component

<https://hughes--hrdevdpro--c.sandbox.vf.force.com/flow/Loa_Attachment/3013K000000GCiWQAW>