Branding LWR Sites for Experience Cloud (Developer Preview)

Salesforce, Spring ’21, Version 51.0

@salesforcedocs

Last updated: February 22, 2021

Important: Branding for LWR sites is available as a developer preview. The feature isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. All commands, parameters, and other features are subject to change or deprecation at any time, with or without notice. Don't implement functionality developed with these commands or tools. You can provide feedback and suggestions for branding LWR sites in the Lightning Web Runtime Experiences group in the Trailblazer Community.

Contents

Overview

The new Lightning Web Runtime (LWR) powers the latest template from Experience Cloud—Build Your Own (LWR)—and introduces a new branding system that uses `--dxp` styling hooks. With the new system, you can modify base Lightning components and custom components more easily to achieve a consistent look and feel across your site.

Note: To use this developer preview branding system, your site must be based on the Build Your Own (LWR) template. For more information, see the LWR Sites for Experience Cloud guide.

The Salesforce Lightning Design System (SLDS) currently uses styling hooks for base components. Styling hooks use CSS custom properties, which are variables within your CSS that cascade to all descendents within the scope of a selector. For example, the lightning-button component uses the styling hook `--sds-c-button-color-background` to change its background color. You can define the hook in any selector.

```
<link rel="stylesheet" href="{ basePath }/assets/styles/dxp-styling-hooks.min.css?{
versionKey}" />
<link rel="stylesheet" href="{ basePath
}/assets/styles/dxp-slds-extensions.min.css?{ versionKey}" />

<style>
  /**
   * Scoped to the root of the document and all its descendant elements.
   */
  :root {
    --sds-c-button-color-background: peachpuff;
  }

  /**
   * Scoped to any element with the class applied and all its descendant elements
   */
  .container {
    --sds-c-button-color-background: peachpuff;
  }
</style>
```

Although it's helpful to define these variables in the head markup when you want to test changes, a more permanent solution is to add a global stylesheet as a static resource. By using these styling hooks, you can programmatically define branding for individual base components used throughout your application.

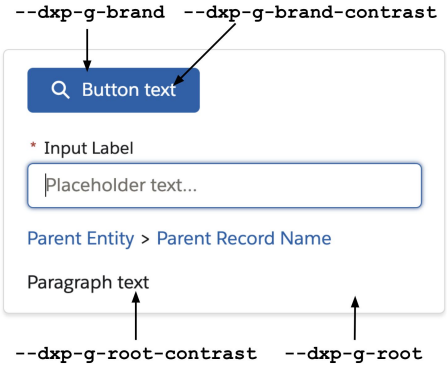
However, accurately reflecting your brand across every component in a site or application, including associated variations and states, often requires hundreds of styling hook definitions. Also, most colors come in pairs—a background and foreground—such as the text on top of a card background. These color pairings must also have sufficient contrast to meet accessibility standards. And manually defining every styling hook can be tedious and error prone.

But with the introduction of the new `--dxp` styling hooks, you can brand an entire application much more easily.

--dxp Styling Hooks

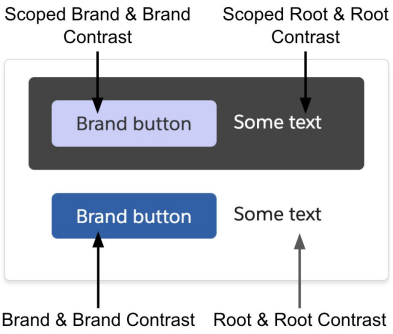
The new `--dxp` styling hooks are a reduced set of custom properties that map to the lower-level component styling hooks. With `--dxp` styling hooks, you can set a single hook that affects many individual components.

For example, setting the hook `--dxp-g-brand` affects the background color of the button, the link color, and the focus border color of the input.



Root is the background color of the container, with root-contrast the foreground color. Each color pairing must maintain an acceptable contrast ratio for accessibility.

Any container can inherit styles (default) or define new styles. If a scoped container sets its own root (background color), you must reevaluate all other `--dxp` hooks to make sure that they're accessible against the new root.



Tip: If you define a new scoped container, make sure that your custom component references the root hooks:

```
background-color: var(--dxp-g-root);
color: var(--dxp-g-root-contrast);
```

Important: Use the `--dxp` styling hooks to make general changes across all components. Then use the `--sds` styling hooks to fine-tune individual components where necessary.

```
<link rel="stylesheet" href="{ basePath
}/assets/styles/dxp-styling-hooks.min.css?{ versionKey}" />
<link rel="stylesheet" href="{ basePath
}/assets/styles/dxp-slds-extensions.min.css?{ versionKey}" />

<style>
  :root {
    /** Use --dxp-g to make broad changes **/
    --dxp-g-root: #1a1b1e;
    --dxp-g-root-contrast: #fff;
    --dxp-g-brand: #5eb4ff;
    --dxp-g-brand-contrast: #fff;
    --dxp-g-neutral: #76716b;
    --dxp-g-neutral-contrast: #fff;

    /** Use --sds-c to fine-tune where necessary **/
    --sds-c-button-color-background: peachpuff;
  }
</style>
```

Available --dxp Styling Hooks

DXP provides a number of hooks for specific semantic use cases. It's important to use these global hooks correctly, or your custom component can respond incorrectly when the hooks are defined.

The color palette is divided into families of colors. Each family has a specific use case with a scale of possible values. The derivation of those values are increasingly contrasted against the root. For example:

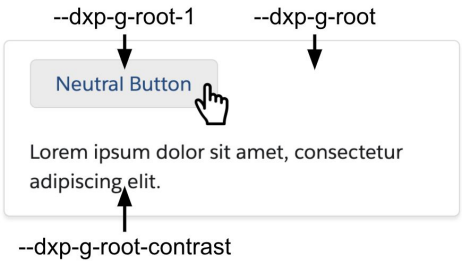
- If the root background color is dark, derivation becomes increasingly lighter to contrast against the background.
- If the root background color is light, derivation becomes increasingly darker to contrast against the background.

These derivation colors are often used for interaction states. For example, the color of a button can change from `--dxp-g-brand` to `--dxp-g-brand-1` on hover.

Important: Use the correct family of hooks for your use case. For example, if the main color of your brand is red, don't use `--dxp-g-destructive` because it also happens to be red. Use `--dxp-g-destructive` only for error and invalid states. Instead, to define your brand color, use `--dxp-g-brand`.

Root

The background color of the page or a section within the page. Root-1 is often used for components that retain the root background color, but have an interaction state—for example, the background hover state of neutral buttons.

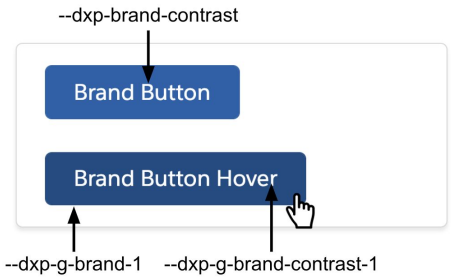


- `--dxp-g-root`
- `--dxp-g-root-contrast`
- `--dxp-g-root-1`
- `--dxp-g-root-contrast-1`

Tip: If you redefine `--dxp-g-root` on a section within a page, reevaluate all other `--dxp` styling hooks to make sure that they're accessible against the new root. If they're not, you must redefine the hooks.

Brand

The primary brand color of your site. For Salesforce, the color is blue. Commonly used on buttons, links, focus states, and so on.

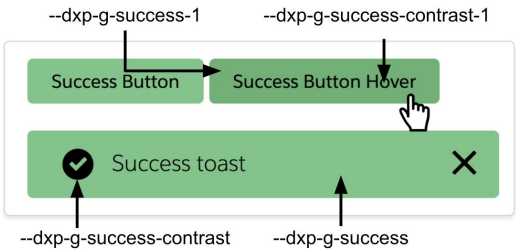


- `--dxp-g-brand`
- `--dxp-g-brand-contrast`
- `--dxp-g-brand-1`
- `--dxp-g-brand-contrast-1`

Success

Communicates success. Commonly used on badges, alerts, toasts, and success variant buttons.

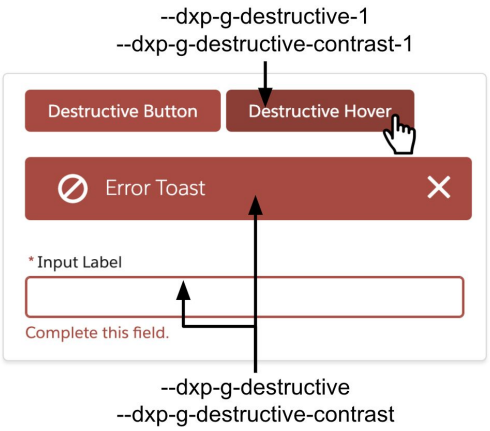
- `--dxp-g-success`
- `--dxp-g-success-contrast`
- `--dxp-g-success-1`
- `--dxp-g-success-contrast-1`



Destructive

Communicates an error or an invalid state. Used on alerts, badges, toasts, form fields in an error state, and destructive variant buttons.

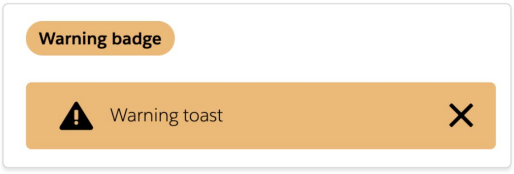
- `--dxp-g-destructive`
- `--dxp-g-destructive-contrast`
- `--dxp-g-destructive-1`
- `--dxp-g-destructive-contrast-1`



Warning

Communicates a warning to the user. Used on badges, alerts, and toasts.

- `--dxp-g-warning`
- `--dxp-g-warning-contrast`

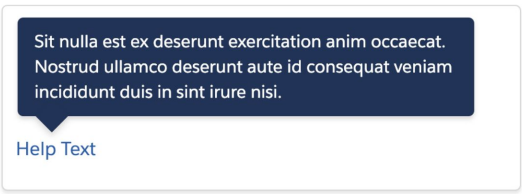


--dxp-g-warning & --dxp-g-warning-contrast

Info

Communicates non-urgent information. Used on tooltips and popovers.

- `--dxp-g-info`
- `--dxp-g-info-contrast`
- `--dxp-g-info-1`
- `--dxp-g-info-contrast-1`

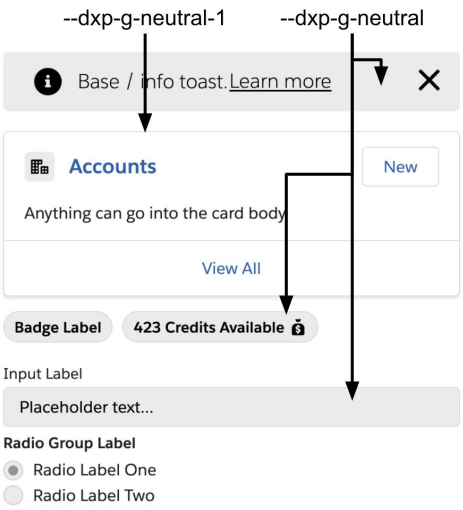


--dxp-g-info & --dxp-g-info-contrast

Neutral

Used to break flow between elements with borders and shadows. Neutral colors are also used for non-urgent informational elements, such as toasts and badges, and elements that don't have interaction, such as icons and disabled inputs.

- `--dxp-g-neutral`
- `--dxp-g-neutral-contrast`
- `--dxp-g-neutral-1`
- `--dxp-g-neutral-contrast-1`
- `--dxp-g-neutral-2`
- `--dxp-g-neutral-contrast-2`
- `--dxp-g-neutral-3`
- `--dxp-g-neutral-contrast-3`



Start Using the -dxp Styling Hooks

To use the `--dxp` styling hooks, add the DXP branding stylesheet to your LWR site. Click **Settings** | **Advanced** | **Edit Head Markup**, and include the following code in the Head Markup editor.

```
<link rel="stylesheet" href="{ basePath }/assets/styles/dxp-styling-hooks.min.css?{
versionKey}" />
<link rel="stylesheet" href="{ basePath
}/assets/styles/dxp-slds-extensions.min.css?{ versionKey}" />
```

Tip: Make sure that the `dxp-styling-hooks.min.css` and `dxp-slds-extension.min.css` files are loaded *after* `salesforce-lightning-design-system.min.css`.

Brand Your Custom Components

To build a custom component that uses the new branding system, you must use the `--dxp` styling hooks.

This sample shows the code for a custom combobox component.

```
<template>
  <input type="text">
  <ul>
    <li>Option 1</li>
    <li>Option 2</li>
  </ul>
</template>
```

To ensure that the input looks similar to other base Lightning components that also respond to branding changes, the CSS must reference the `--dxp` styling hooks as follows.

```
input {
  border-color: var(--dxp-g-neutral);
}

input:focus {
  border-color: var(--dxp-g-brand);
}
```

Important:

- Only reference `--dxp` hooks within your custom components. Don't reference `--sds` hooks.
- Values of CSS custom properties are resolved at the time of evaluation. For example, let's say you have a CSS custom property that references another CSS custom property. If you update the value of the latter CSS custom property in a lower scope, the value of the former CSS custom property doesn't reflect the new value.

Add Custom Fonts

You can add custom fonts by uploading the font file as a static resource. Alternatively, you can reference a file that's hosted externally.

Upload Fonts as a Static Resource

To upload your custom font as a static resource and reference it within the head markup:

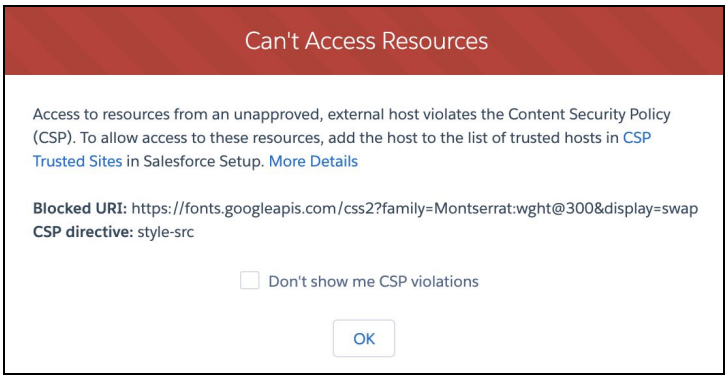
1. In Setup, in the Quick Find box, enter Static Resources, and then select **Static Resources**.
2. Click **New**, upload the file, and give the static resource a name. Keep a note of the resource name. If your site has public pages, select **Public** in the Cache Control setting. If you don't make the font resource publicly available, the page uses the browser's default font instead.
3. To add a reference to the font in your site's head markup, return to Experience Builder, and click **Settings** | **Advanced** | **Edit Head Markup**.
4. Insert the `@font-face` declaration and define the appropriate `--dxp` styling hooks in your desired scope (either `:root{}` to the whole page or within a specific selector or component).
    - `--dxp-g-root-font-family` defines all text other than headers.
    - `--dxp-g-heading-font-family` defines headline text.

```
<link rel="stylesheet" href="{ basePath
}/assets/styles/dxp-styling-hooks.min.css?{ versionKey}" />
<link rel="stylesheet" href="{ basePath
}/assets/styles/dxp-slds-extensions.min.css?{ versionKey}" />

<style>
  @font-face {
    font-family: 'myFirstFont';
    /* Replace myFont with your resource name */
    src: url('{ basePath }/sfsites/c/resource/myFont') format('woff');
  }

  :root {
    /** set the font for all root/body text **/
    --dxp-g-root-font-family: 'myFirstFont', Helvetica, sans-serif;

    /** set the font for headings **/
    --dxp-g-heading-font-family: 'myFirstFont', Times, serif;
  }
</style>
```

Tip: Make sure that the font file format, for example, woff, matches your markup. Also make sure that your fallback values, such as Helvetica, sans-serif, and so on, are properly defined for your brand. To learn more, see @font-face.

Use Externally Hosted Fonts

You can use fonts that are hosted outside Salesforce, such as Google Fonts. However, to access externally hosted files, you must also update the Content Security Policy (CSP) for your org by adding the hosts to your list of CSP Trusted Sites. Otherwise, an error appears indicating that the resources can't be accessed.



For example, for Google Fonts, you would add:

- https://fonts.googleapis.com to access the Google Fonts stylesheet
- https://fonts.gstatic.com to access fonts from Google Font

Step 1: Add Sites to Your Org's List of CSP Trusted Sites

1. From Setup, in the Quick Find box, enter CSP Trusted Sites, and click **CSP Trusted Sites**.
2. Click **New Trusted Site**.
3. To add a trusted site for external stylesheets:
    a. Enter a name—for example, GoogleFontAPI.
    b. Add the URL—for example, https://fonts.googleapis.com.
    c. Make sure that it's **Active**, and select **Allow site for style-src**.
    d. Click **Save & New**.
4. To add a trusted site for external fonts:
    a. Enter a name—for example, GoogleFontStatic.
    b. Add the URL—for example, https://fonts.gstatic.com.
    c. Make sure that it's **Active**, and select **Allow site for font-src**.
    d. Save your changes.

Step 2: Reference the External Resources Within the Head Markup

1. In Experience Builder, click **Settings** | **Advanced** | **Edit Head Markup**.
2. Add the link to the external file. For example, this code sample adds a link to the Google Fonts stylesheet and then refers to the font in the CSS style definition.

```
<link rel="stylesheet" href="{ basePath
}/assets/styles/dxp-styling-hooks.min.css?{ versionKey}" />
<link rel="stylesheet" href="{ basePath
}/assets/styles/dxp-slds-extensions.min.css?{ versionKey}" />

<!-- Load Google Fonts -->
<link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@300&display=swa
p" rel="stylesheet">

<style>
  :root {
    /** set the font for all root/body text **/
    --dxp-g-root-font-family: 'Montserrat', sans-serif;

    /** set the font for headings **/
    --dxp-g-heading-font-family: 'Montserrat', sans-serif;
  }
</style>
```

3. Save your changes.