Use SLDS Best Practices to Opt In to Enhanced Lightning UI

Shelby Hubick, Principal Architect
Timothy Yeh, Product Manager

Shelby Hubick

Principle Engineering Architect

Timothy Yeh

Product Manager

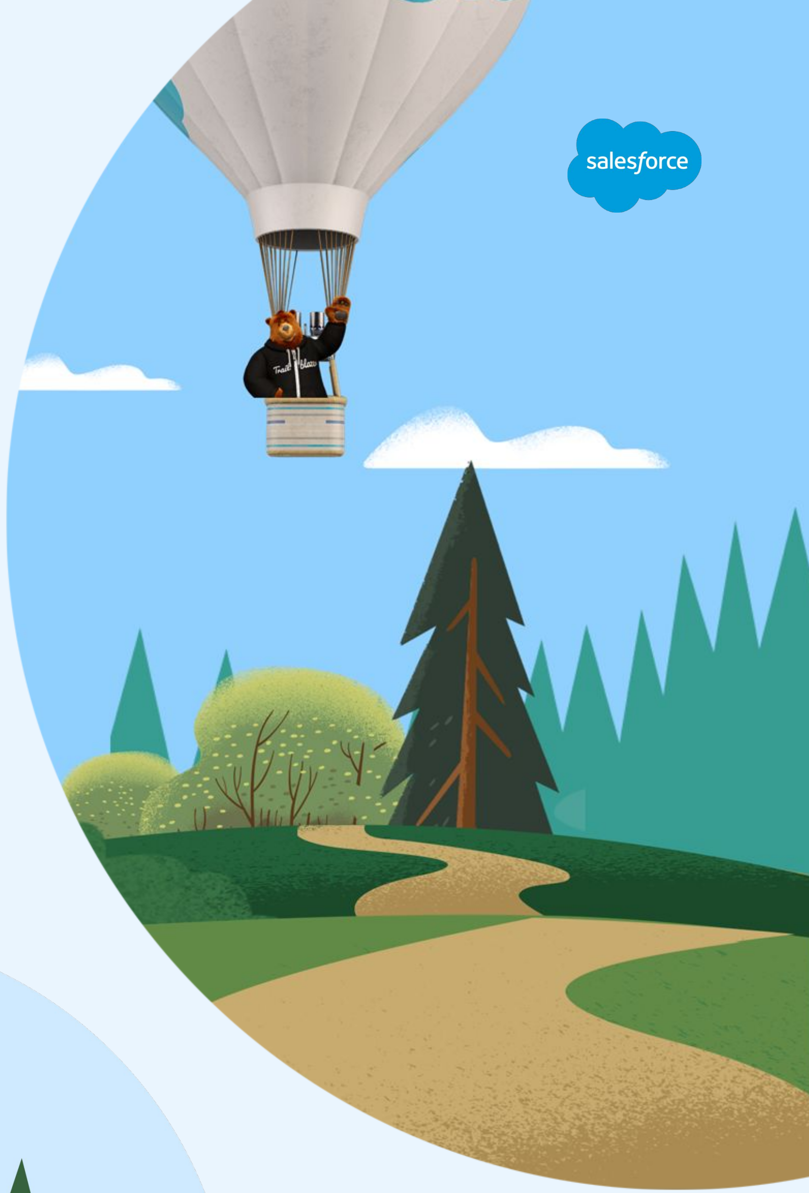
Forward Looking Statements

This presentation contains forward-looking statements about, among other things, trend analyses and statements regarding future events, anticipated growth and industry prospects, and our strategies, expectation or plans regarding product releases and enhancements. The achievement or success of the matters covered by such forward-looking statements involves risks, uncertainties and assumptions. If any such risks or uncertainties materialize or if any of the assumptions prove incorrect, results or outcomes could differ materially from those expressed or implied by these forward-looking statements. The risks and uncertainties referred to above include those factors discussed in Salesforce's reports filed from time to time with the Securities and Exchange Commission, including, but not limited to: our ability to meet the expectations of our customers; uncertainties regarding AI technologies and its integration into our product offerings; the effect of evolving domestic and foreign government regulations; regulatory developments and regulatory investigations involving us or affecting our industry; our ability to successfully introduce new services and product features; our ability to execute our business plans; the pace of change and innovation in enterprise cloud computing services; and our ability to maintain and enhance our brands.

Last updated: April 25, 2024

Agenda

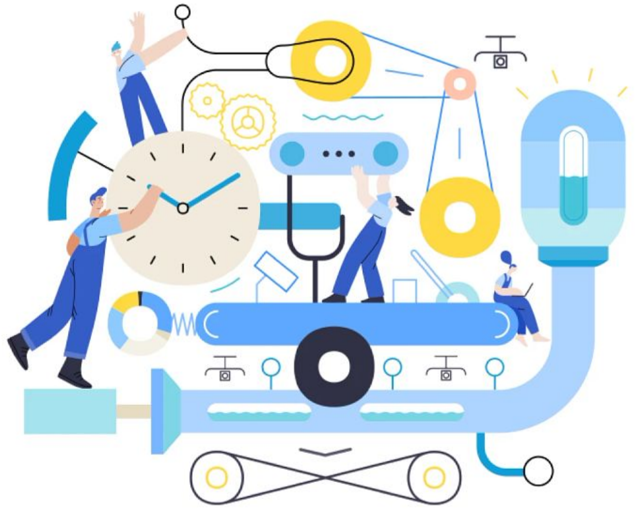
- Overview of new design for Lightning
- Salesforce Lightning Design System 2
- Developer best practices
- Tooling: SLDS Validator



Goals for Today's Session

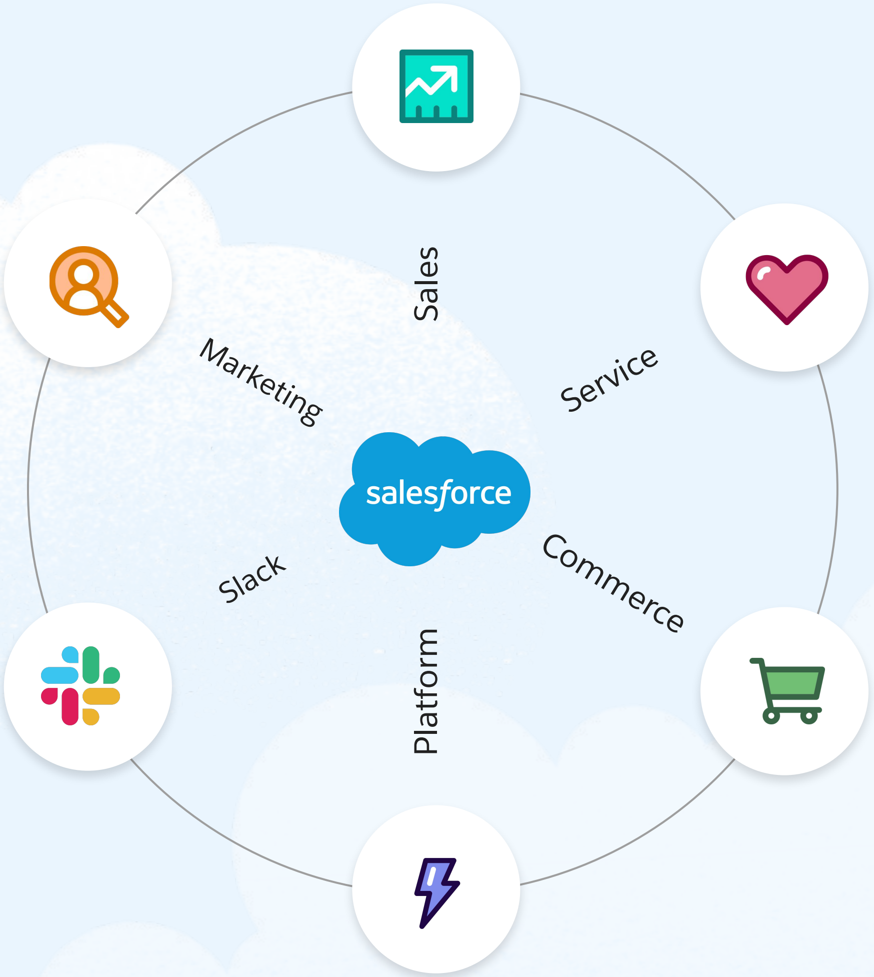
Reduce Tech Debt

Create Better Looking UI

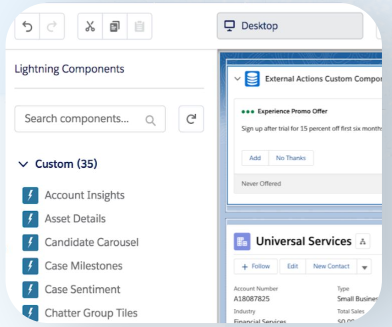


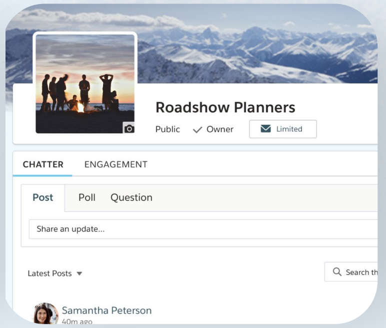
Create More Reliable Apps

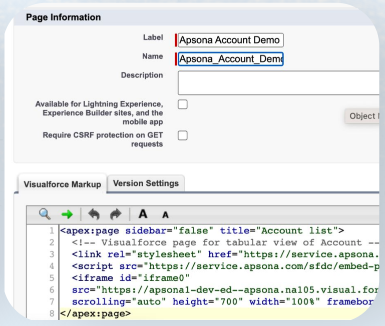
A Common Salesforce Journey



Many Years of Customizations






Page Information

Label: Apsona Account Demo
Name: Apsona_Account_Demo
Description:
Available for Lightning Experience, Experience Builder sites, and the mobile app: ☐
Require CSRF protection on GET requests: ☐

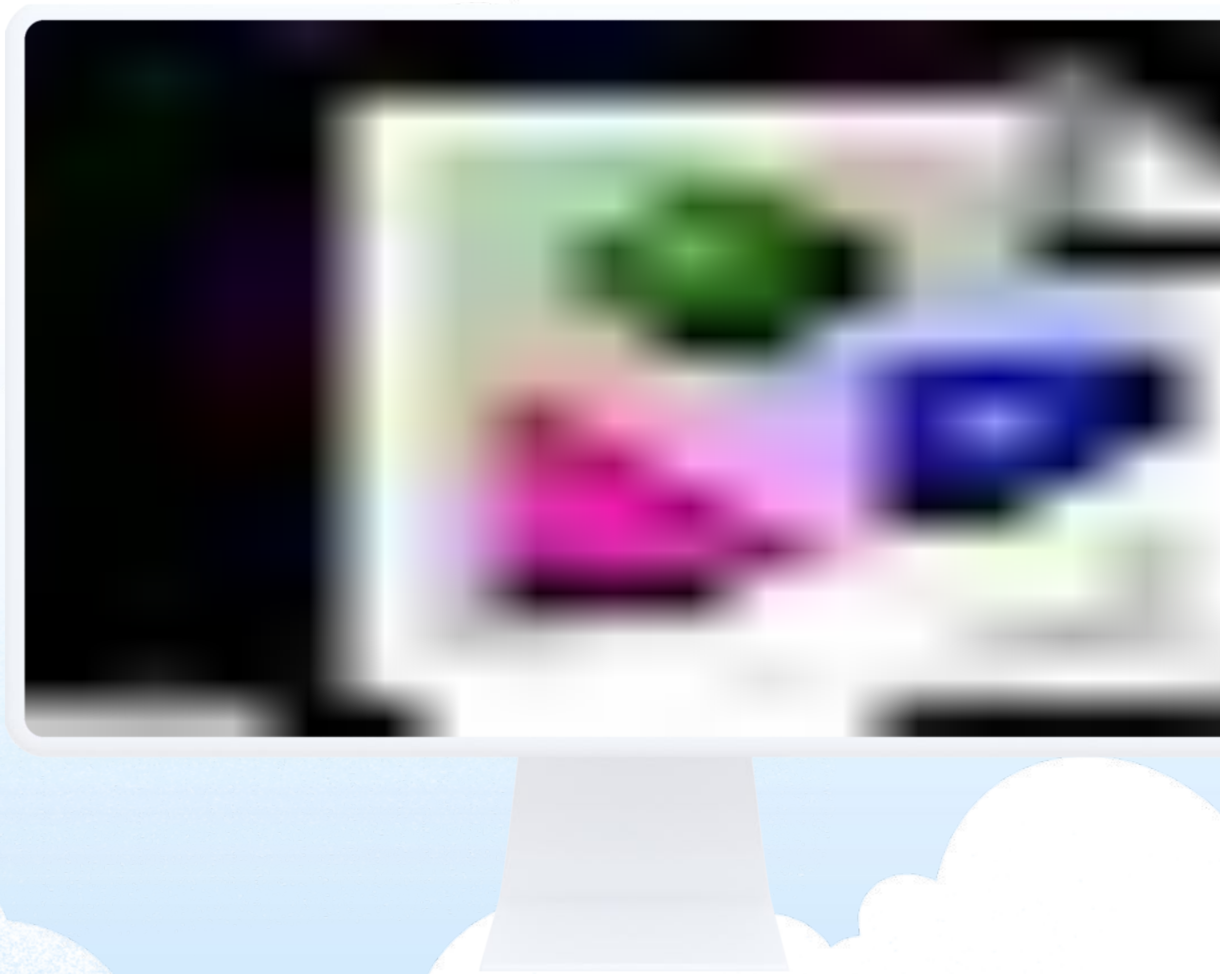
```
<apex:page sidebar="false" title="Account list">
  <!-- Visualforce page for tabular view of Account --
  <link rel="stylesheet" href="https://service.apsona.
  <script src="https://service.apsona.com/sfdc/embed-p
  <iframe id="iframe0"
  src="https://apsona1-dev-ed--apsona.na105.visual.for
  scrolling="auto" height="700" width="100%" frameboro
</apex:page>
```

New Design
For Lightning UI

Improves navigation, ease of use, and accessibility.

Introduces new colors, icons, borders, typography, and more.

Has clearer indicators of success and prioritization.



Availability by Edition

New and Existing Starter Orgs | GA Today

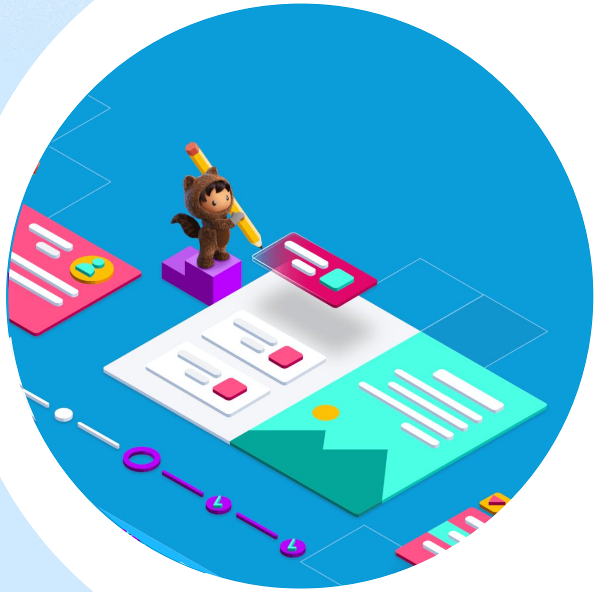
New Sales Professional and Enterprise Orgs | GA Today

All Other Orgs | Coming Soon!

Salesforce Lightning Design System 2 (SLDS 2)

The next evolution of SLDS is coming soon

SLDS



SLDS 2

Same HTML as SLDS

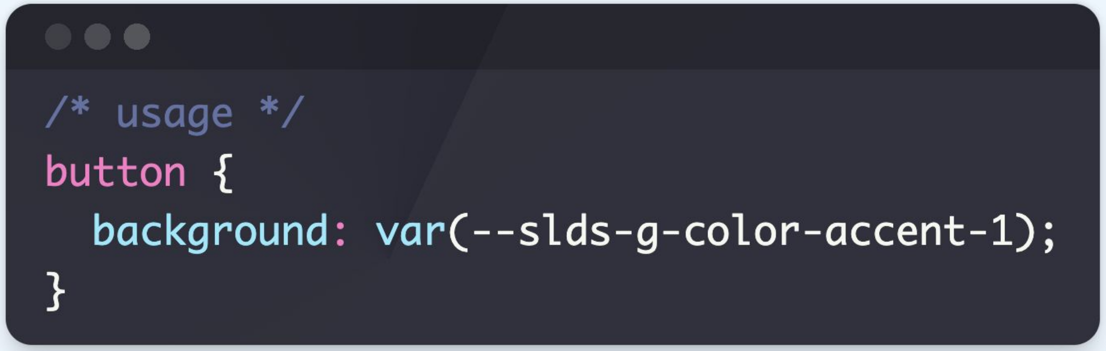
Same CSS Classes as SLDS

Uses New Styling Hook API

Enhanced Theming and Branding

What Are Styling Hooks?

Styling Hooks are **CSS Variables (Custom Properties)** which store values like colors, fonts, sizes, and all of the other styles associated with the new design.



```
/* usage */
button {
  background: var(--slds-g-color-accent-1);
}
```

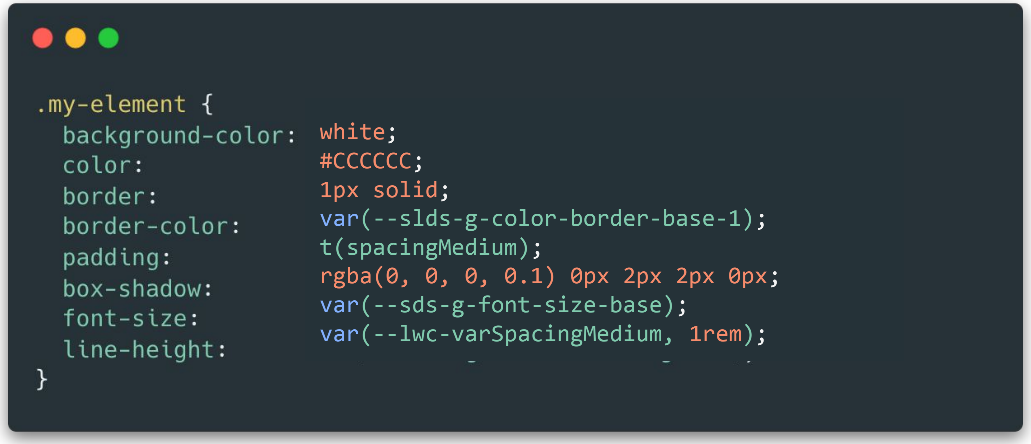
```
--slds-g-color-surface-1
```

[Say Yes! To the Power of Styling Hooks]() Wed 2:30pm

Differences Between SLDS and SLDS 2

SLDS 2 streamlines by using styling hooks

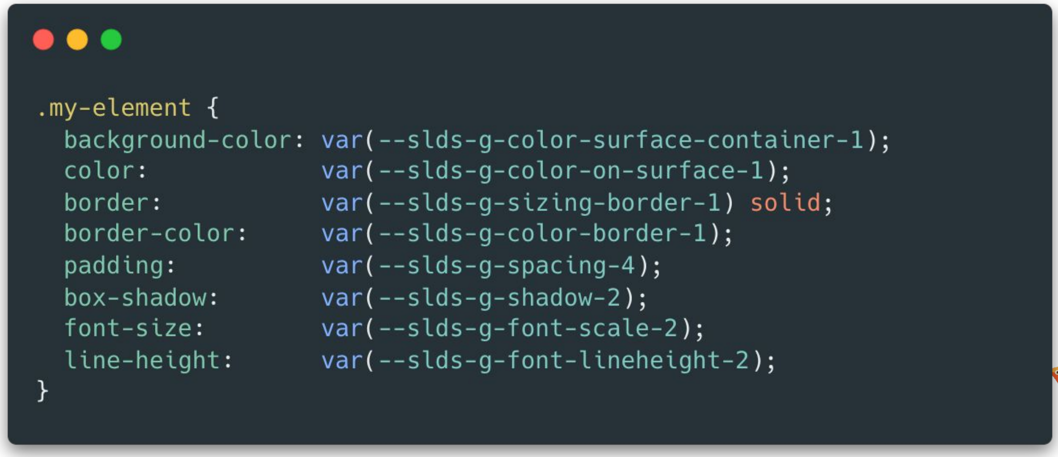
SLDS Using Design Tokens and Hard Coded Values



```
.my-element {
  background-color: white;
  color:            #CCCCCC;
  border:           1px solid;
  border-color:     var(--slds-g-color-border-base-1);
  padding:          t(spacingMedium);
  box-shadow:       rgba(0, 0, 0, 0.1) 0px 2px 2px 0px;
  font-size:        var(--sds-g-font-size-base);
  line-height:      var(--lwc-varSpacingMedium, 1rem);
}
```

- Flexible CSS
- Mix of hard coded values, design tokens, and limited styling hooks
- Limited theming and customization capabilities

SLDS 2 Using Styling Hook Values



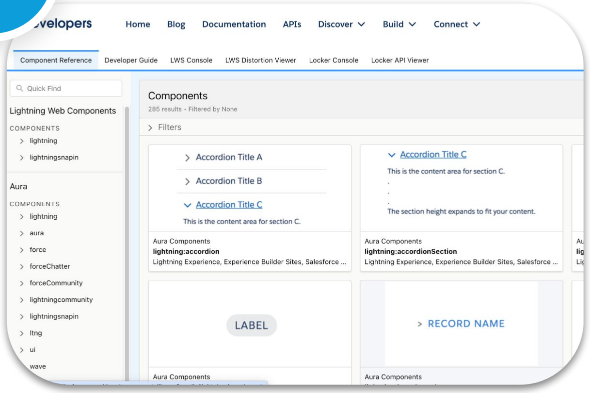
```
.my-element {
  background-color: var(--slds-g-color-surface-container-1);
  color:            var(--slds-g-color-on-surface-1);
  border:           var(--slds-g-sizing-border-1) solid;
  border-color:     var(--slds-g-color-border-1);
  padding:          var(--slds-g-spacing-4);
  box-shadow:       var(--slds-g-shadow-2);
  font-size:        var(--slds-g-font-scale-2);
  line-height:      var(--slds-g-font-lineheight-2);
}
```

- Composable CSS
- New, streamlined styling hook architecture
- Enhanced theming and customization capabilities

How Best to Use SLDS?

To author and customize experiences

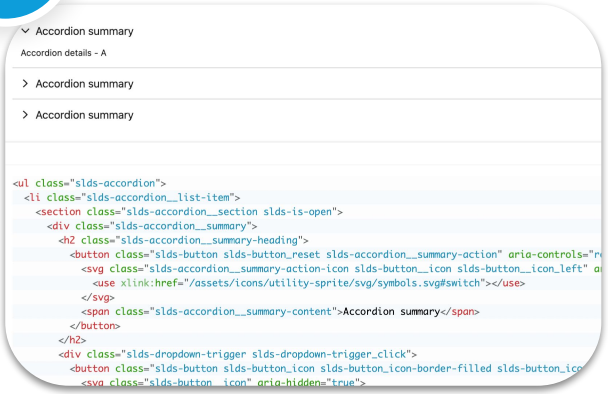
1



Use Lightning Base Components

Best practices built inside: accessibility, branding, security and more.

2

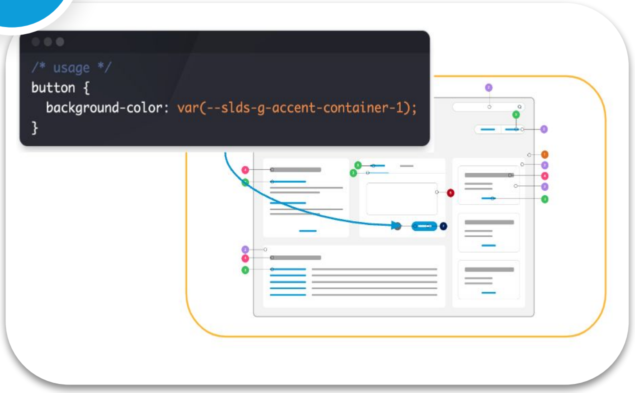


```html
<ul class="slds-accordion">
  <li class="slds-accordion__list-item">
    <section class="slds-accordion__section slds-is-open">
      <div class="slds-accordion__summary">
        <h2 class="slds-accordion__summary-heading">
          <button class="slds-button slds-button_reset slds-accordion__summary-action" aria-controls="r
            <svg class="slds-accordion__summary-action-icon slds-button__icon slds-button__icon_left" a
              <use xlink:href="/assets/icons/utility-sprite/svg/symbols.svg#switch"></use>
            </svg>
            <span class="slds-accordion__summary-content">Accordion summary</span>
          </button>
        </h2>
        <div class="slds-dropdown-trigger slds-dropdown-trigger_click">
          <button class="slds-button slds-button_icon slds-button_icon-border-filled slds-button_ico
            <svg class="slds-button__icon" aria-hidden="true">
```

Use SLDS Blueprints

Accessible HTML/CSS for faster and flexible development.

3



```css
/* usage */
button {
  background-color: var(--slds-g-accent-container-1);
}
```

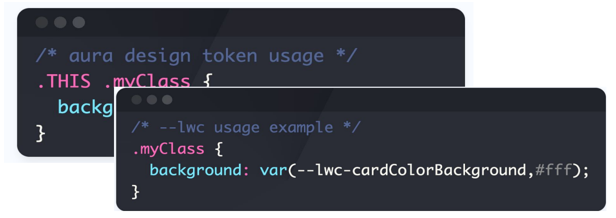
Use Styling Hooks

Powers theming and branding, and powers advanced customizations.

Best Practices Bird's Eye View

1

Upgrade your styling API

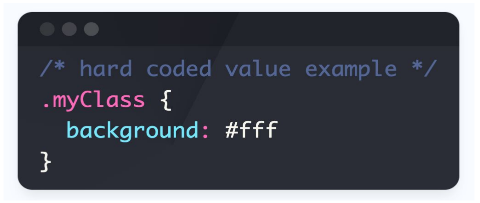


```
/* aura design token usage */
.THIS .myClass {
  backg
}
```

```
/* --lwc usage example */
.myClass {
  background: var(--lwc-cardColorBackground,#fff);
}
```

2

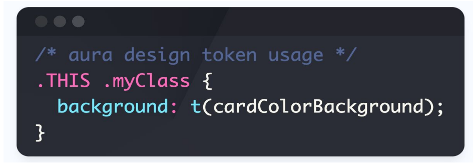
Avoid hard-coded values



```
/* hard coded value example */
.myClass {
  background: #fff
}
```

3

Just say NO to styling SLDS classes

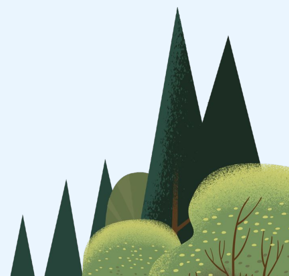


```
/* aura design token usage */
.THIS .myClass {
  background: t(cardColorBackground);
}
```

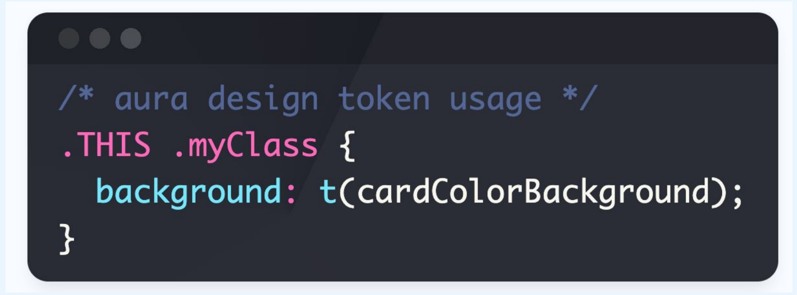
And many more…

- Replace --sds- with --slds- hooks
- Avoid relying on a specific DOM structure
- Replace deprecated dash-dash BEM selectors
- Be wary of !important
- Prefer the new global styling hooks
- Use fallbacks to support backwards compatibility

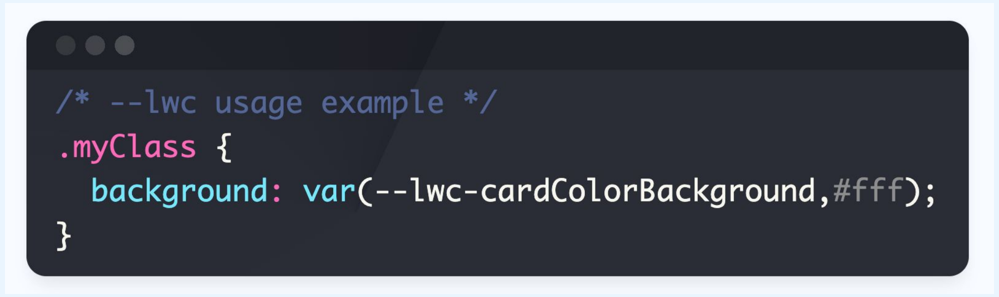
→ SLDS Developer Best Practices



Best Practice: Upgrade Your Styling API

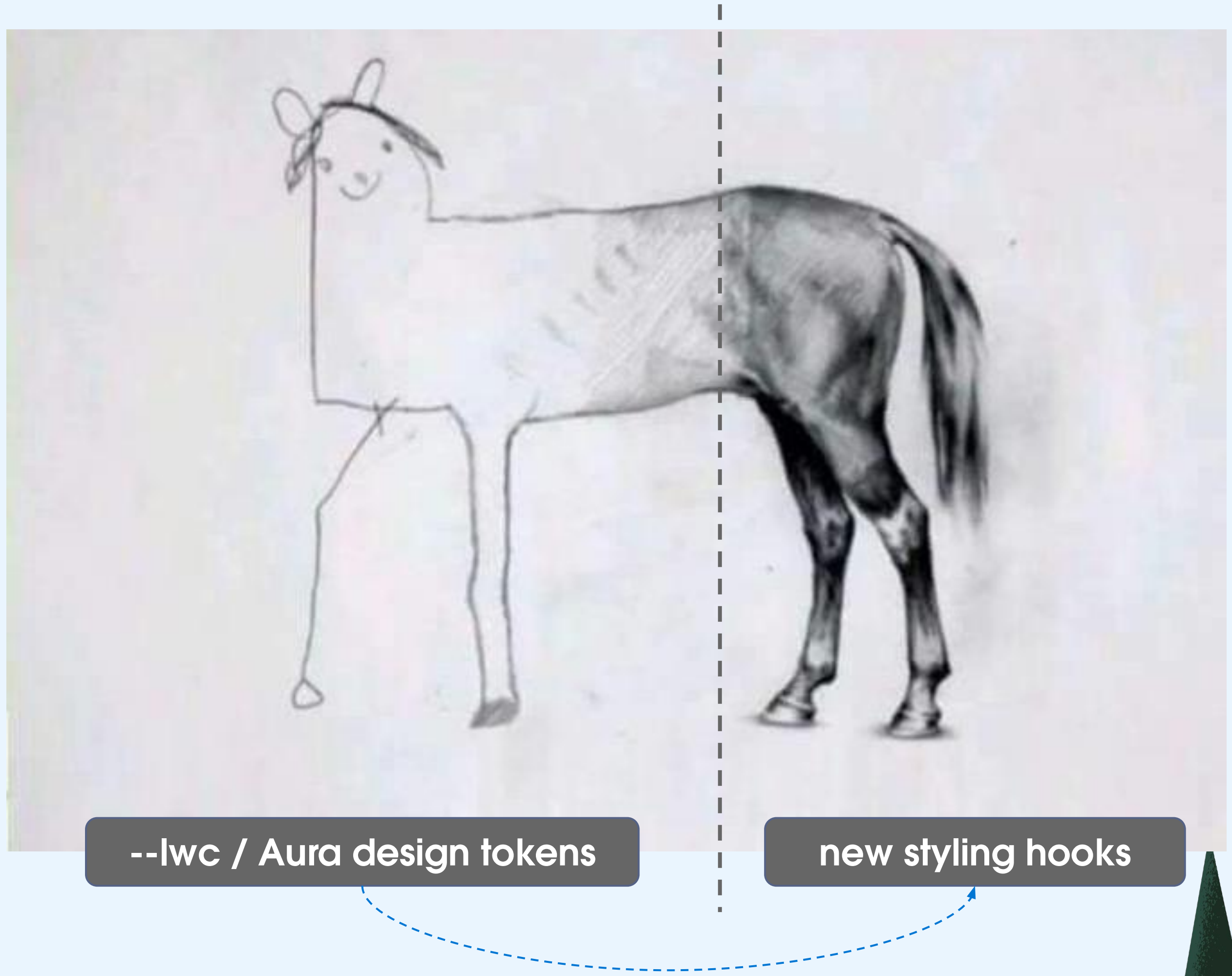


```
/* aura design token usage */
.THIS .myClass {
  background: t(cardColorBackground);
}
```



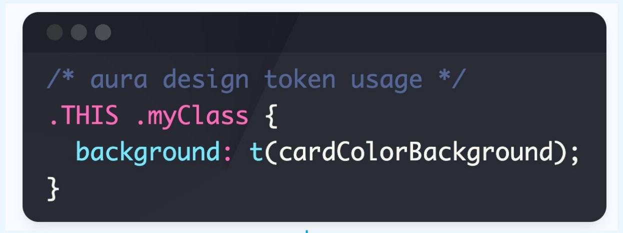
```
/* --lwc usage example */
.myClass {
  background: var(--lwc-cardColorBackground,#fff);
}
```

```
/* --sds styling hook example */
.THIS .myClass {
  background: var(--sds-g-color-surface-container-1);
}
```



STEP-BY-STEP GUIDE

Best Practice: Upgrade Your Styling API



```
/* aura design token usage */
.THIS .myClass {
  background: t(cardColorBackground);
}
```

ELEMENT: CARD

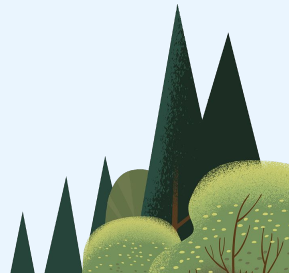
STYLE: BACKGROUND

VALUE: #ffffff (white)

Step 1 of 4

DETERMINE CONTEXT

Are you styling a button, card or a tab, is this a border or background or a hover state. The design token name can help with this too, but can sometimes be misleading or abstract.



STEP-BY-STEP GUIDE

Best Practice: Upgrade Your Styling API

Step 2 of 4

FIND CLOSEST CONTEXT MATCH

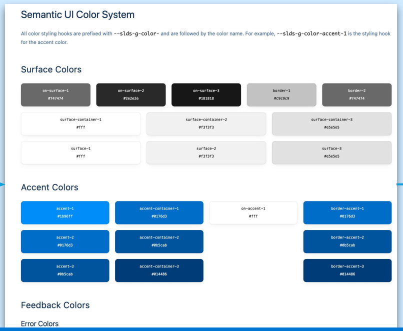
With context, find the global semantic hook that has the closest **semantic** match to the one you're replacing.

```
/* aura design token usage */
.THIS .myClass {
  background: t(cardColorBackground);
}
```

ELEMENT: CARD

STYLE: BACKGROUND

VALUE: #ffffff (white)



Semantic UI Color System

All color styling hooks are prefixed with `--slds-g-color-` and are followed by the color name. For example, `--slds-g-color-accent-1` is the styling hook for the accent color.

Surface Colors

on-surface-1 #747474	on-surface-2 #2e2e2e	on-surface-3 #181818	border-1 #c9c9c9	border-2 #747474

surface-container-1 #fff	surface-container-2 #f3f3f3	surface-container-3 #e5e5e5
surface-1 #fff	surface-2 #f3f3f3	surface-3 #e5e5e5

Accent Colors

accent-1 #1b96ff	accent-container-1 #0176d3	on-accent-1 #fff	border-accent-1 #0176d3
accent-2 #0176d3	accent-container-2 #0b5cab		border-accent-2 #0b5cab
accent-3 #0b5cab	accent-container-3 #014486		border-accent-3 #014486

Feedback Colors

Error Colors

Global Styling Hooks Guidance

`--slds-g-color-surface-container-1`	`#ffffff`

STEP-BY-STEP GUIDE

Best Practice: Upgrade Your Styling API

salesforce

Step 3 of 4

FIND CLOSEST VALUE MATCH

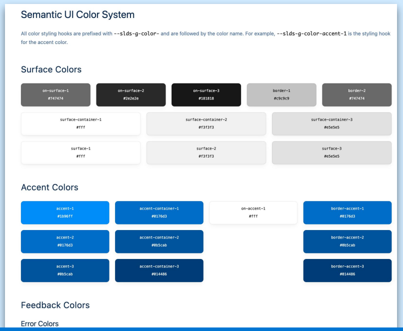
With context, find the global semantic hook that has the closest **value** match to the value that you're replacing.

```
/* aura design token usage */
.THIS .myClass {
  background: t(cardColorBackground);
}
```

ELEMENT: CARD

STYLE: BACKGROUND

VALUE: #ffffff (white)



Semantic UI Color System

All color styling hooks are prefixed with `--slds-g-color-` and are followed by the color name. For example, `--slds-g-color-accent-1` is the styling hook for the accent color.

Surface Colors

on-surface-1 #747474
on-surface-2 #2e2e2e
on-surface-3 #181818
border-1 #c9c9c9
border-2 #747474
surface-container-1 #fff
surface-container-2 #f3f3f3
surface-container-3 #e5e5e5
surface-1 #fff
surface-2 #f3f3f3
surface-3 #e5e5e5

Accent Colors

accent-1 #1b96ff
accent-container-1 #0176d3
on-accent-1 #fff
border-accent-1 #0176d3
accent-2 #0176d3
accent-container-2 #0b5cab
border-accent-2 #0b5cab
accent-3 #0b5cab
accent-container-3 #014486
border-accent-3 #014486

Feedback Colors

Error Colors

[Global Styling Hooks Guidance]

`--slds-g-color-surface-container-1` `#ffffff`

STEP-BY-STEP GUIDE

Best Practice: Upgrade Your Styling API

Step 4 of 4

APPLY FIX WITH FALLBACK

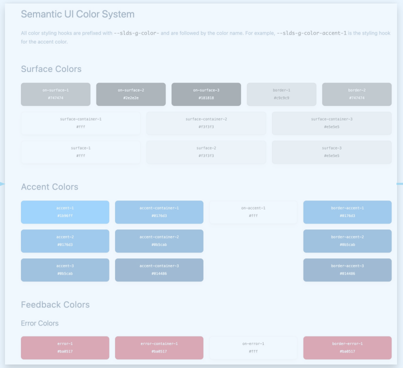
Lastly, take the output styling hook and prepend it to your CSS rule, but don't remove the old hook to help with backwards compatibility.

```
/* aura design token usage */
.THIS .myClass {
  background: t(cardColorBackground);
}
```

ELEMENT: CARD

STYLE: BACKGROUND

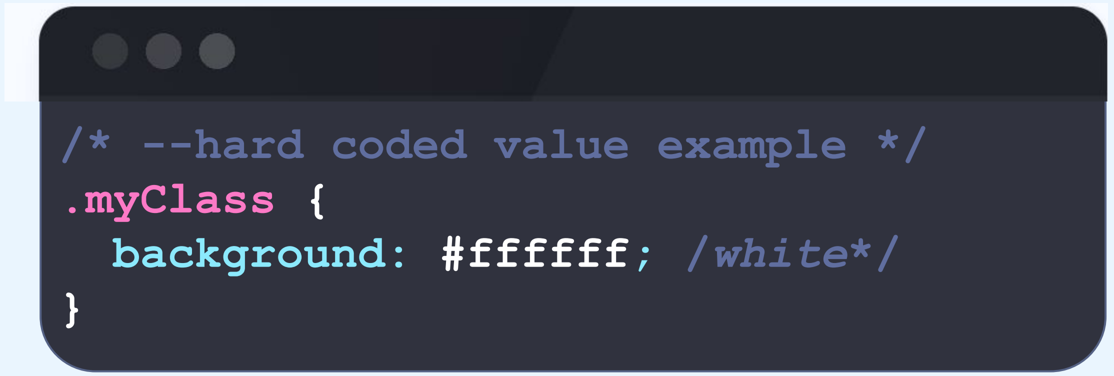
VALUE: #fff



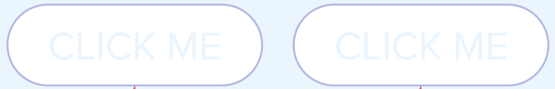
--slds-g-color-surface-container-1	#ffffff

```
/* design token to slds styling hook example */
.THIS .myClass {
  background: var(--slds-g-color-surface-container-1, t(cardColorBackground), #fff);
}
```

Best Practice: Avoid Hard-Coded Values



```css
/* --hard coded value example */
.myClass {
  background: #ffffff; /white*/
}
```



Nothing to see here... literally

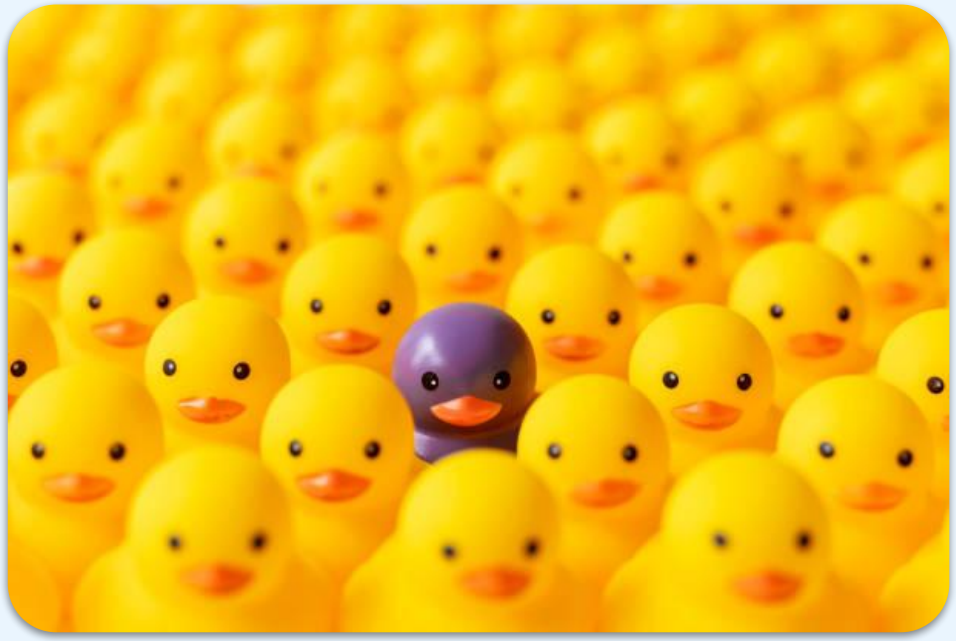
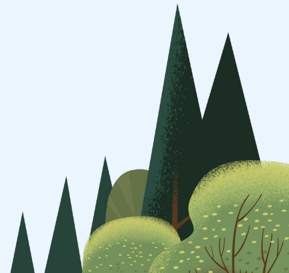


Fig 1.2: Hard-coded ducky, in a dynamic themeable duck-system



Best Practice: Avoid Hard-Coded Values

```
/* --hard coded value example */
.myClass {
  background: #ffffff; /white*/
}
```

CSS TYPES TO FOCUS ON

```
background-color:
color:
border-color:
border-radius:
box-shadow:
fill:
font-size:
font-weight:
padding:
```

Step 1 of 4

DETERMINE CONTEXT

Are you styling a button, card, or tab? Is this a border or background or a hover state?

Step 2 of 4

FIND CLOSEST CONTEXT MATCH

With context, find the global semantic hook that has the closest **semantic** match to the one you're replacing.

Step 3 of 4

FIND CLOSEST VALUE MATCH

With context, find the global semantic hook that has the closest **value** match to the value that you're replacing.

Best Practice: Avoid Hard-Coded Values

Step 4 of 4

APPLY FIX WITH FALLBACK

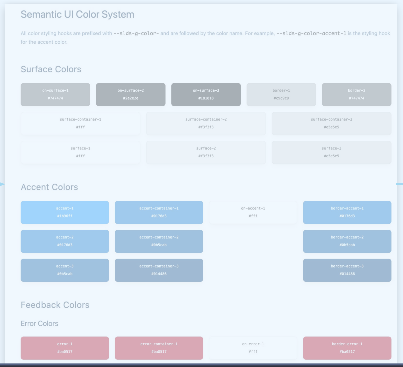
Lastly, take the output styling hook and prepend it to your CSS rule, but don’t remove the old hook to help with backwards compatibility.

```
/* --hard coded value example */
.myClass {
  background: #ffffff; /white*/
}
```

ELEMENT: CARD

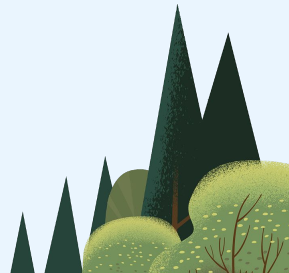
STYLE: BACKGROUND

VALUE: #fff



--slds-g-color-surface-container-1	#ffffff

```
/* hard coded value example */
.myClass {
  background: var(--slds-g-color-surface-container-1,#ffffff)
}
```



Best Practice: Just Say NO to Styling SLDS Classes

Avoid relying on internal DOM structure and SLDS classes.

```css
/* --styling slds class example */
.slds-button {
  border-radius: 1rem;
}
```

Fig 1.3 "Generate me an image of Astro saying no to styling SLDS classes'

Best Practice: Just Say NO to Styling SLDS Classes

```
/* --styling slds class example */
.slds-button {
  border-radius: 1rem;
}
```

✓ OPTION 1: USE CUSTOM CLASS

```
.myClass {
  border-radius: 1rem;
}

// using SLDS blueprint
<button class="slds-button myClass"></button>

// using Lightning Base Component
<lightning-button … class="myClass"></lightning-button>
```

✓ OPTION 2: USE STYLING HOOK

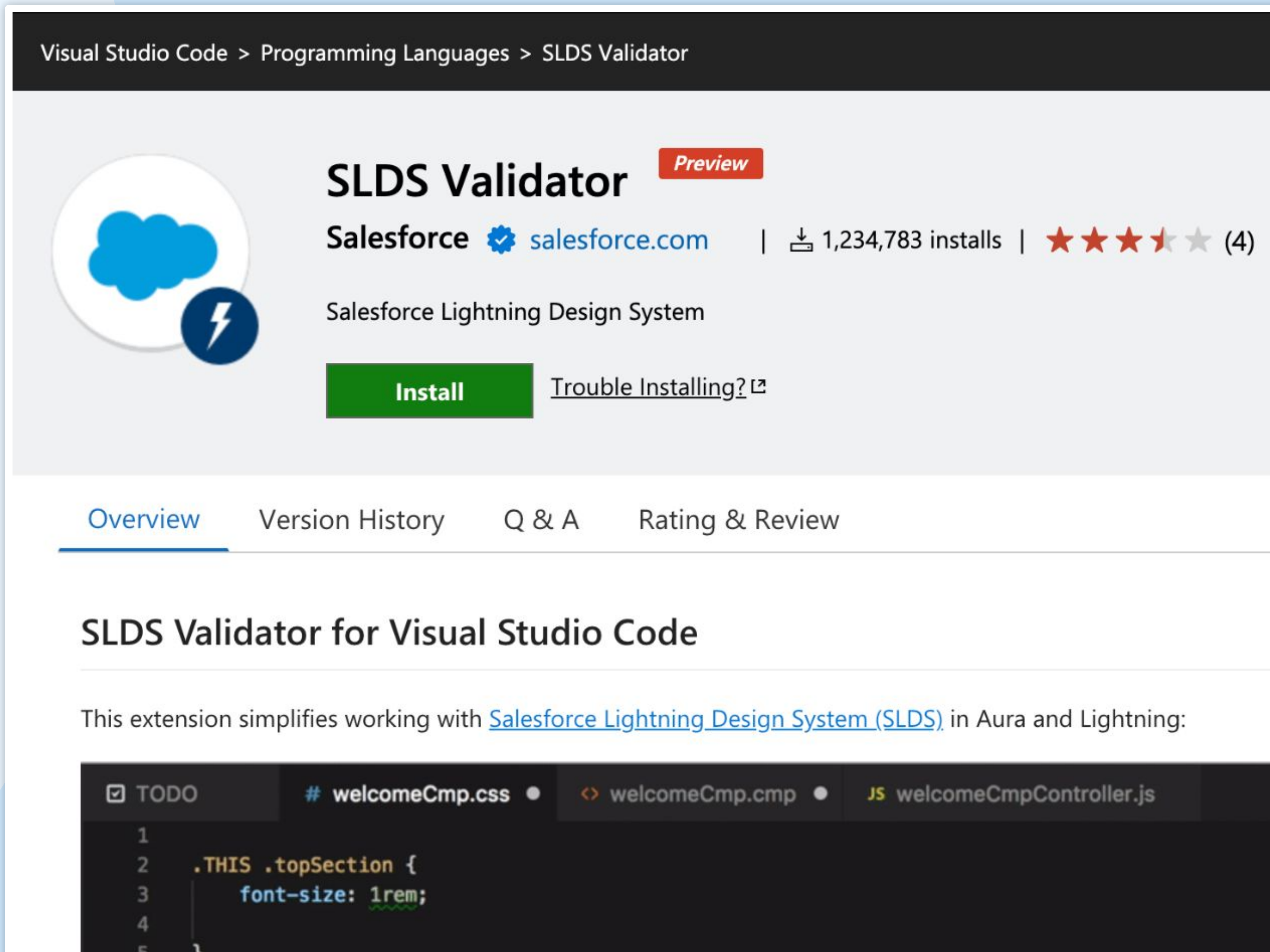
```
.myClass {
  --slds-c-button-radius-border: 1rem;
}

// using SLDS blueprint
<button class="slds-button myClass"></button>

// using Lightning Base Component
<lightning-button … class="myClass"></lightning-button>
```

JUST UPDATED!

IMPROVED SLDS VALIDATION & RECOMMENDATIONS

SLDS Validator

- ✓ Improved recommendations
- ✓ New bulk reporting feature
- ✓ New SLDS 2 (Beta) Rules
- ✓ Many more updates


Visual Studio Code > Programming Languages > SLDS Validator

SLDS Validator Preview

Salesforce salesforce.com | 1,234,783 installs | ★★★★☆ (4)

Salesforce Lightning Design System

SLDS Validator for Visual Studio Code

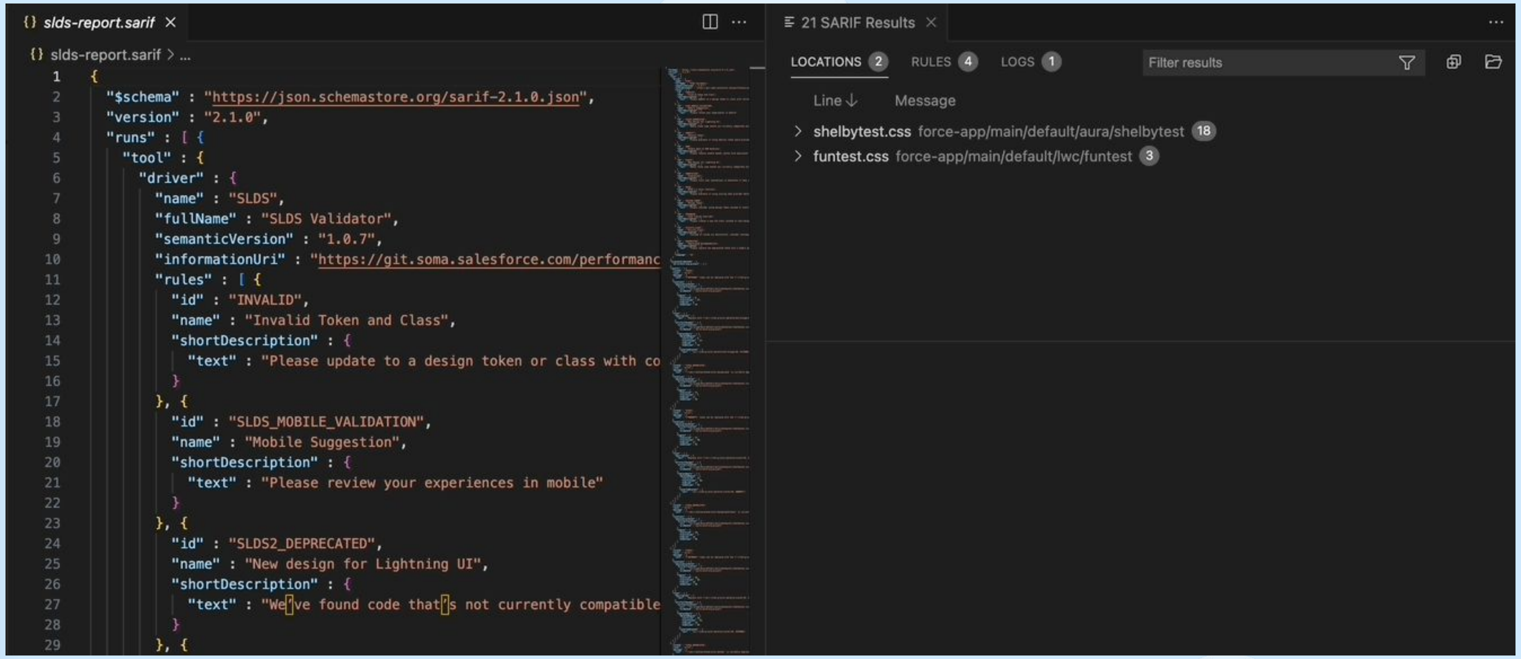
This extension simplifies working with Salesforce Lightning Design System (SLDS) in Aura and Lightning:

```
.THIS .topSection {
    font-size: 1rem;
```

SLDS Validator

New Bulk Reporting Feature

- ✓ Scans all your Aura / LWC components
- ✓ Generates a SARIF file format (requires a SARIF viewer plugin)
- ✓ Includes all warnings with recommendations to refactor

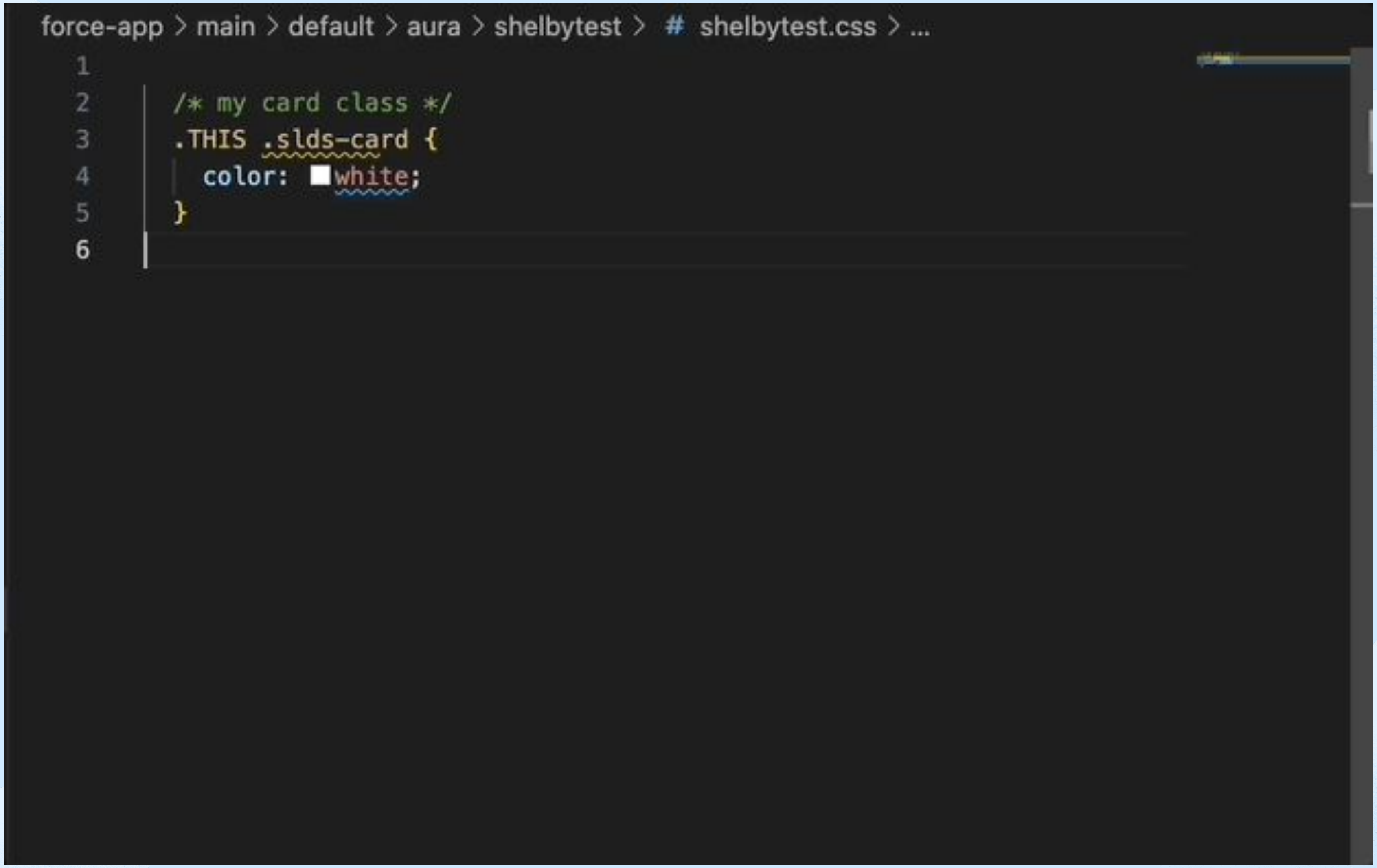


```
{
  "$schema" : "https://json.schemastore.org/sarif-2.1.0.json",
  "version" : "2.1.0",
  "runs" : [ {
    "tool" : {
      "driver" : {
        "name" : "SLDS",
        "fullName" : "SLDS Validator",
        "semanticVersion" : "1.0.7",
        "informationUri" : "https://git.soma.salesforce.com/performanc
        "rules" : [ {
          "id" : "INVALID",
          "name" : "Invalid Token and Class",
          "shortDescription" : {
            "text" : "Please update to a design token or class with co
          }
        }, {
          "id" : "SLDS_MOBILE_VALIDATION",
          "name" : "Mobile Suggestion",
          "shortDescription" : {
            "text" : "Please review your experiences in mobile"
          }
        }, {
          "id" : "SLDS2_DEPRECATED",
          "name" : "New design for Lightning UI",
          "shortDescription" : {
            "text" : "We've found code that's not currently compatible
          }
        }, {
```

```
> shelbytest.css force-app/main/default/aura/shelbytest 18
> funtest.css force-app/main/default/lwc/funtest 3
```

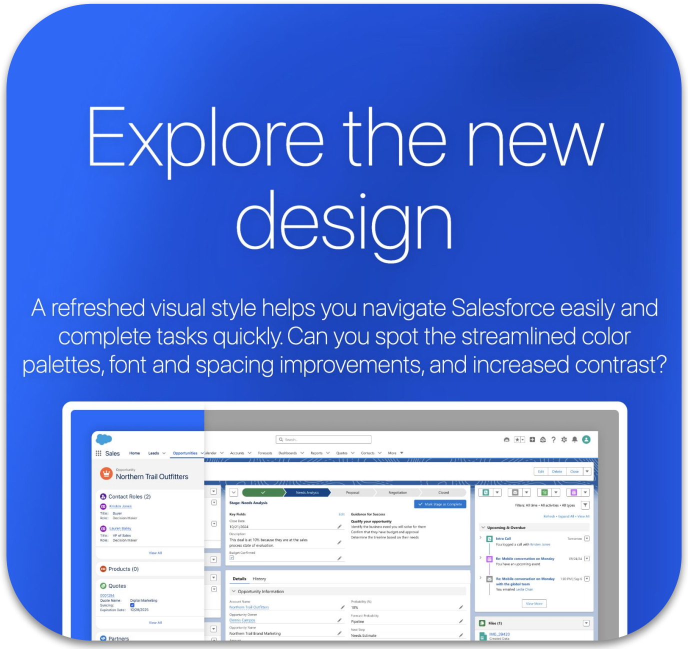
SLDS Validator

Validator + Hard-Coded Value Example


force-app > main > default > aura > shelbytest > # shelbytest.css > ...

```

/* my card class */
.THIS .slds-card {
  color: white;
}

```

Find Everything on lightningdesignsystem.com

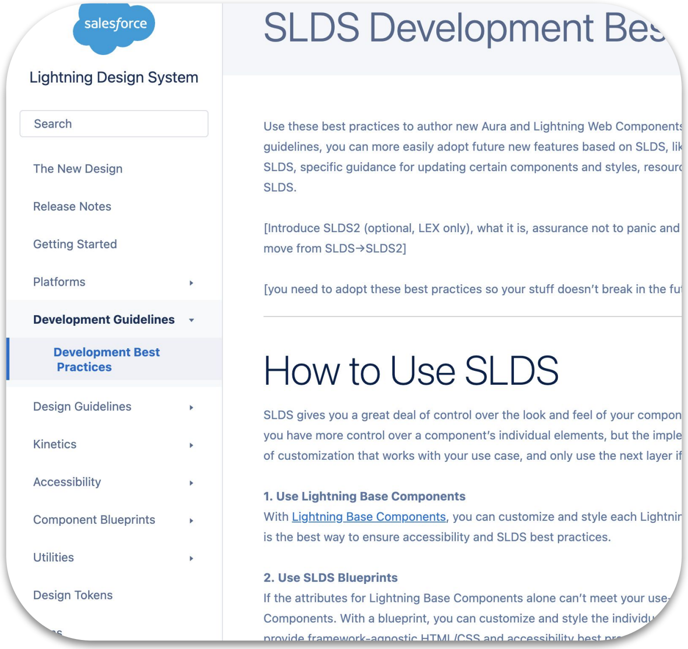


Explore the new design

A refreshed visual style helps you navigate Salesforce easily and complete tasks quickly. Can you spot the streamlined color palettes, font and spacing improvements, and increased contrast?

New Design Site

The main page for all things related to the new design.



SLDS Development Bes

Use these best practices to author new Aura and Lightning Web Components guidelines, you can more easily adopt future new features based on SLDS, lik SLDS, specific guidance for updating certain components and styles, resourc SLDS.

[Introduce SLDS2 (optional, LEX only), what it is, assurance not to panic and move from SLDS→SLDS2]

[you need to adopt these best practices so your stuff doesn't break in the fu

How to Use SLDS

SLDS gives you a great deal of control over the look and feel of your compon you have more control over a component's individual elements, but the imple of customization that works with your use case, and only use the next layer if

1. Use Lightning Base Components

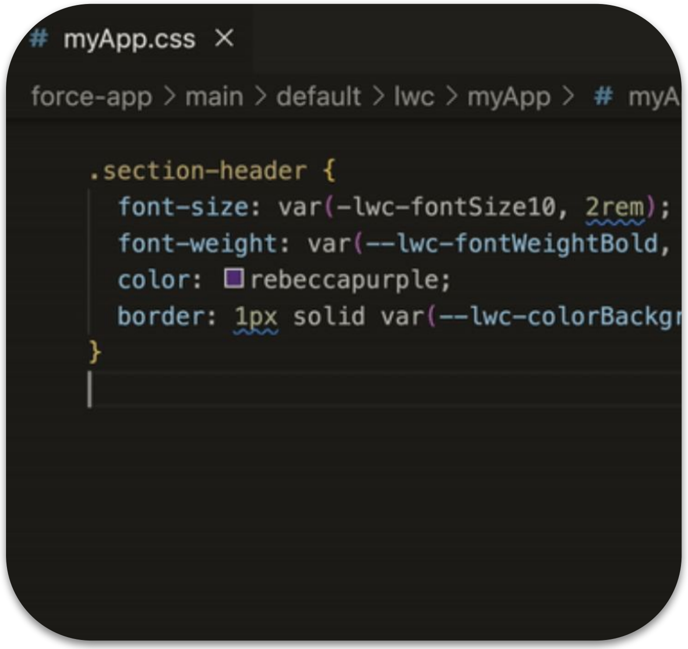
With Lightning Base Components, you can customize and style each Lightnin is the best way to ensure accessibility and SLDS best practices.

2. Use SLDS Blueprints

If the attributes for Lightning Base Components alone can't meet your use Components. With a blueprint, you can customize and style the individu provide framework-agnostic HTML/CSS and accessibility best pra

SLDS Best Practices

Best practices to follow when developing in SLDS or SLDS 2.



```
.section-header {
  font-size: var(-lwc-fontSize10, 2rem);
  font-weight: var(--lwc-fontWeightBold,
  color: ■rebeccapurple;
  border: 1px solid var(--lwc-colorBackgr
}
```

SLDS Validator

Tooling to make following best practices easier.

Find Everything on lightningdesignsystem.com

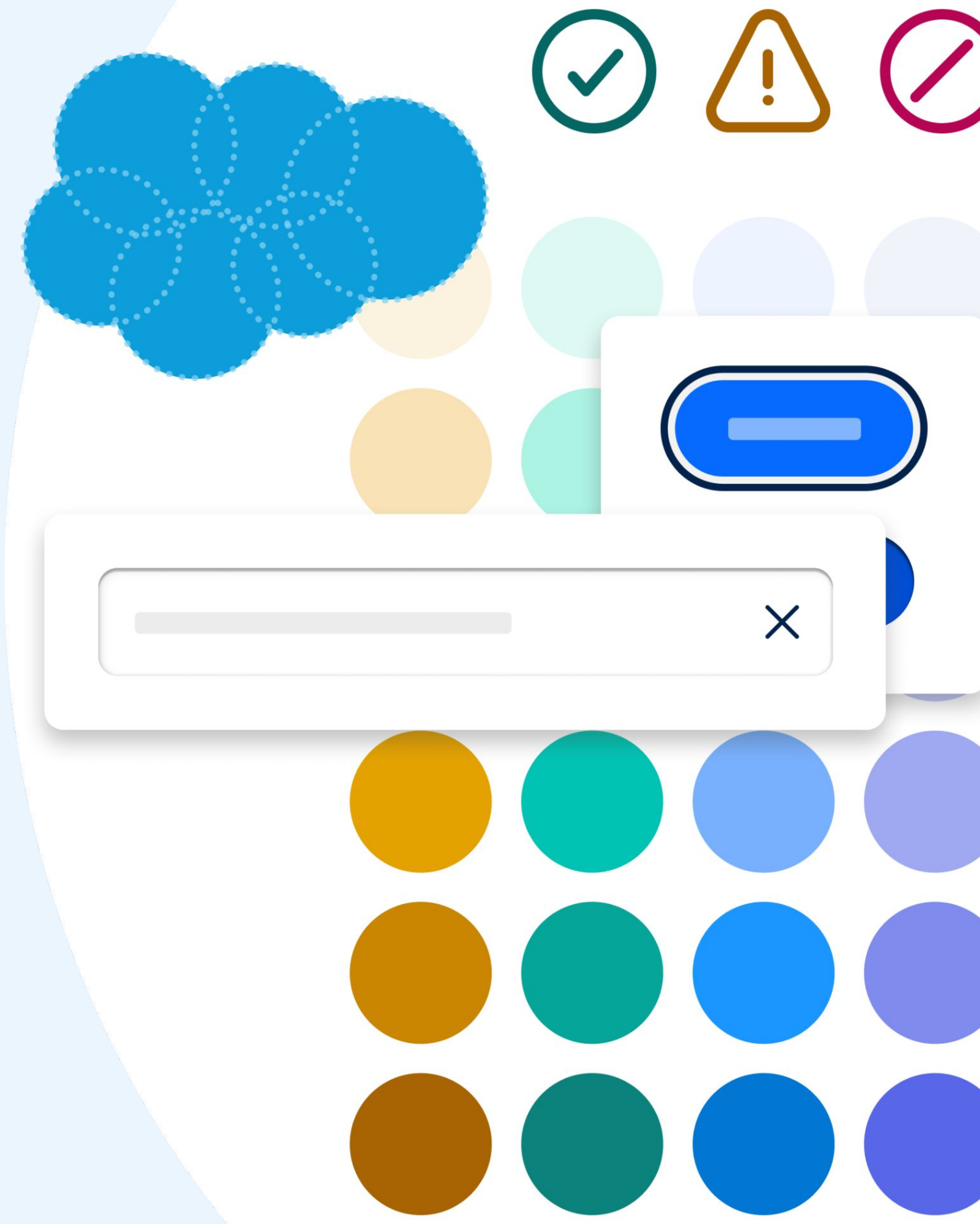
- Learn about the new design
- Explore designer & developer best practices & tools
- Stay up to speed with availability plans

Today's Takeaways

1 Following best practices will ensure seamless adoption of the new design and other future enhancements.

2 Use Lightning Base Components and customize with styling hooks. Avoid hard-coded values and styling SLDS classes

3 Use SLDS Validator to help you follow best practices when coding custom UI.



dreamforce®

Thank you

Coffee on us.

The first 4,000 attendees to provide feedback on this event will receive a $5 Starbucks gift card.

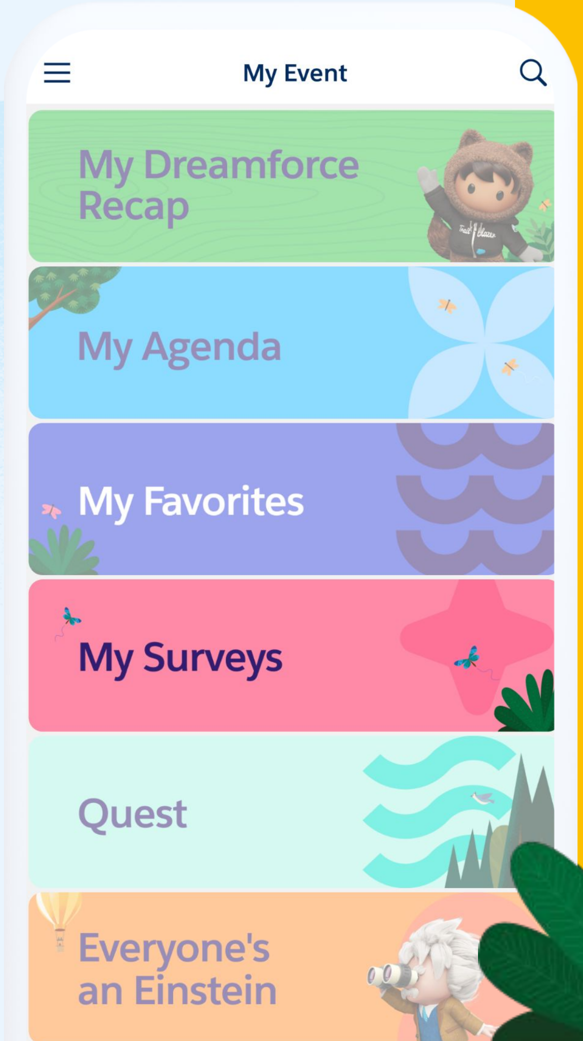
Open the Salesforce Events mobile app.

Navigate to **My Event.**

Select **My Surveys.**

Complete four Session Surveys and present the completed Event Survey page at Badge Pickup to redeem.*

*Restrictions apply. See rules at sforce.co/survey-terms


My Event
My Dreamforce Recap
My Agenda
My Favorites
My Surveys
Quest
Everyone's an Einstein