**Using Custom Styles**

Use the <apex:stylesheet> tag or static HTML to include your own style sheet or styles.

For HTML tags, you can define inline CSS code, just like in a regular HTML page.

|  |  |
| --- | --- |
| 1 | <apex:page> |
| 2 | <style type="text/css"> | |

|  |  |  |
| --- | --- | --- |
| 3 | p { font-weight: bold; } | |
| 4 | </style> |

|  |  |
| --- | --- |
| 5 |  |
| 6 | <p>This is some strong text!</p> | |

|  |  |
| --- | --- |
| 7 | </apex:page> |

This example references a style sheet that is defined as a static resource. First, create a style sheet and upload it as a static resource named customCSS.

|  |  |
| --- | --- |
| 1 | h1 { color: #f00; } |
| 2 | p { background-color: #eec; } | |

|  |  |
| --- | --- |
| 3 | newLink { color: #f60; font-weight: bold; } |

Next, create a page that refers to this static resource.

|  |  |
| --- | --- |
| 01 | <apex:page showHeader="false"> |
| 02 | <apex:stylesheet value="{!$Resource.customCSS}" /> | |

|  |  |
| --- | --- |
| 03 | <h1>Testing Custom Stylesheets</h1> |
| 04 | <p>This text could go on forever...<br/><br/> | |

|  |  |  |
| --- | --- | --- |
| 05 | But it won't!</p> | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | <apex:outputLink value="[http://www.salesforce.com](http://www.salesforce.com/)" styleClass="newLink"> | |
| 08 | Click here to switch to www.salesforce.com |

|  |  |  |
| --- | --- | --- |
| 09 | </apex:outputLink> | |
| 10 | </apex:page> |

Tip

If you’re not using Salesforce styles, you can shrink your page size by preventing the standard Salesforce style sheets from loading. To prevent loading, set the standardStylesheets attribute on the <apex:page>component to false.

|  |  |  |
| --- | --- | --- |
| 1 | <apex:page standardStylesheets="false"> | |
| 2 | <!-- page content here --> |

|  |  |
| --- | --- |
| 3 | </apex:page> |

If you don’t load the Salesforce style sheets, components that require them don’t display correctly.

Visualforce components that produce HTML have pass-through style and styleClass attributes. These attributes allow you to use your own styles and style classes to control the look and feel of the resulting HTML. style allows you to set styles directly on a component, while styleClass lets you attach classes for styles defined elsewhere. For example, the following code sets the class of the <apex:outputText> and applies a style.

|  |  |  |
| --- | --- | --- |
| 01 | <apex:page> | |
| 02 |  |

|  |  |
| --- | --- |
| 03 | <style type="text/css"> |
| 04 | .asideText { font-style: italic; } | |

|  |  |  |
| --- | --- | --- |
| 05 | </style> | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | <apex:outputText style="font-weight: bold;" | |
| 08 | value="This text is styled directly."/> |

|  |  |
| --- | --- |
| 09 |  |
| 10 | <apex:outputText styleClass="asideText" | |

|  |  |  |
| --- | --- | --- |
| 11 | value="This text is styled via a stylesheet class."/> | |
| 12 |  |

|  |  |
| --- | --- |
| 13 | </apex:page> |

To apply a style using a DOM ID, use CSS attribute selectors for the style definition. See [Defining Styles for a Component’s DOM ID](https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_styling_custom_using_component_id.htm#pages_styling_custom_using_component_id).

If you intend to use images in your style sheet, zip the images with the CSS file, and upload the file as a single static resource. For example, suppose your CSS file has a line like the following.

|  |  |
| --- | --- |
| 1 | body { background-image: url("images/dots.gif") } |

Combine the entire images directory and the parent CSS file into a single zip file. In this example, the zip file resource name is myStyles.

|  |  |
| --- | --- |
| 1 | <apex:stylesheet value="{!URLFOR($Resource.myStyles, 'styles.css')}"/> |