

**MADCAP FLARE 2023 r2**

# Styles

Copyright © 2023 MadCap Software. All rights reserved.

Information in this document is subject to change without notice. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of those agreements. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of MadCap Software.

MadCap Software  
9171 Towne Center Drive, Suite 335  
San Diego, California 92122  
858-320-0387  
[www.madcapsoftware.com](http://www.madcapsoftware.com)

**THIS PDF WAS CREATED USING MADCAP FLARE.**

# CONTENTS

---

## CHAPTER 1

Introduction .....	6
--------------------	---

## CHAPTER 2

Basics of CSS .....	9
W3C and CSS Resources .....	11
What Can CSS Do? .....	12
CSS Styles and Page Structure .....	13
Inline, Embedded, and External CSS .....	14
Cascading .....	17
Important Style Terms and Concepts .....	20
Selectors .....	22
Attributes .....	24
Declaration Blocks .....	25
Advanced Selectors .....	28
Classes .....	33
Style Classes (e.g., for Notes, Tips, Examples) .....	33
Generic Classes .....	38
Identifiers .....	48
Pseudo Class Expressions .....	55
Pseudo Elements .....	60
Inheritance .....	64

### **CHAPTER 3**

General Information for Styles .....	70
Stylesheet and Formatting Options .....	71
Types of Styles in Flare .....	74
Primary and Local Stylesheets (and Precedence) .....	86
MadCap-Specific Styles and Properties .....	99
Where's My Style? .....	101

### **CHAPTER 4**

Main Activities for Styles .....	103
Creating Stylesheets .....	104
Creating Selectors .....	106
Editing Styles in a Regular Stylesheet .....	111
Applying Styles to Content .....	136
Associating Primary Stylesheets With All Files .....	151
Associating Stylesheets Locally With Specific Files .....	153

### **CHAPTER 5**

Regular Stylesheets .....	157
Stylesheets .....	159
Styles .....	168
Property Values .....	215
CSS Variables .....	218
Style Inspector .....	235
Style Reports .....	269

### **CHAPTER 6**

Table Stylesheets .....	270
Creating Table Stylesheets .....	272
Setting Table Styles for Print Output .....	275



Editing Table Stylesheets .....	279
Applying Table Stylesheets to Tables .....	294
<b>CHAPTER 7</b>	
Mediums and Media Queries .....	298
Mediums .....	299
Media Queries .....	309
General Information for Mediums and Media Queries .....	312
Main Activities for Mediums and Media Queries .....	319
<b>APPENDIX A</b>	
MadCap-Specific Styles .....	337
<b>APPENDIX B</b>	
MadCap-Specific Properties .....	350
<b>APPENDIX C</b>	
PDFs .....	365
Tutorials .....	365
Cheat Sheets .....	366
User Guides .....	367
<b>INDEX .....</b>	<b>368</b>

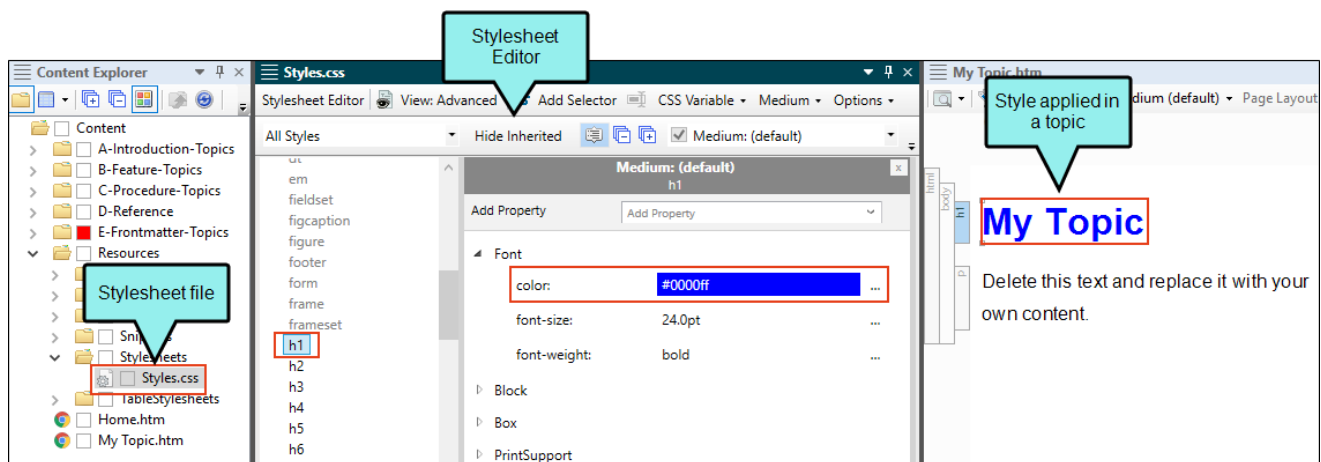
## CHAPTER 1

# Introduction

Styles are used to control the look and feel of your documentation, and keep the content separate from its presentation. The styling is based on cascading stylesheets (CSS), which is an international standard for formatting web content, developed by the World Wide Web Consortium (or [W3C](#)).

Flare lets you work with stylesheets in a number of ways. You can use a primary stylesheet, automatically associating it with all files at the target level or project level (recommended). However, if you have some topics that you want to use a different stylesheet, you also have the option of associating those individual files with that other stylesheet. Once you've set up your stylesheet, you can apply its styles to the different pieces of content in your topics and snippets.

As much as possible, you should avoid the opposite of styles, which is local formatting. For example, you can highlight some text and make it green and italic right where that content exists. But if you make that same change in many places, it takes a lot longer and it's a lot more work to control the look of that content if you later change your mind.



## Basics of CSS

- "W3C and CSS Resources" on page 11
- "What Can CSS Do?" on page 12
- "CSS Styles and Page Structure" on page 13
- "Inline, Embedded, and External CSS" on page 14
- "Cascading" on page 17
- "Important Style Terms and Concepts" on page 20
- "Inheritance" on page 64

## General Information

- "Stylesheet and Formatting Options" on page 71
- "Types of Styles in Flare" on page 74
- "Primary and Local Stylesheets (and Precedence)" on page 86
- "MadCap-Specific Styles and Properties" on page 99
- "Where's My Style?" on page 101

## Main Activities

- "Creating Stylesheets" on page 104
- "Creating Selectors" on page 106
- "Editing Styles in a Regular Stylesheet" on page 111
- "Applying Styles to Content" on page 136
- "Associating Primary Stylesheets With All Files" on page 151
- "Associating Stylesheets Locally With Specific Files" on page 153

## Other Features and Activities

- "Regular Stylesheets" on page 157
- "Table Stylesheets" on page 270
- "Mediums and Media Queries" on page 298



**NOTE** If you are interested in your project's look and feel from a branding perspective, consider the `Branding.css` file. This stylesheet groups together common branding elements (e.g., logo, hero image, font, color palette) to match the output with your company's brand.

## CHAPTER 2

---

# Basics of CSS

To understand how to use styles for maximum benefit in Flare, you should review the basics of cascading stylesheets (CSS).

This chapter discusses the following:

W3C and CSS Resources .....	11
What Can CSS Do? .....	12
CSS Styles and Page Structure .....	13
Inline, Embedded, and External CSS .....	14
Cascading .....	17
Important Style Terms and Concepts .....	20
Selectors .....	22
Attributes .....	24
Declaration Blocks .....	25
Advanced Selectors .....	28
Classes .....	33
Style Classes (e.g., for Notes, Tips, Examples) .....	33
Generic Classes .....	38
Identifiers .....	48
Pseudo Class Expressions .....	55
Pseudo Elements .....	60

Inheritance .....64

# I W3C and CSS Resources

The vast majority of the styles and properties in Flare were developed by the World Wide Web Consortium (W3C). For more thorough information about each W3C style and property, refer to <https://www.w3.org> and use the search feature on the website.

In addition to the W3C, several recorded webinars dealing with CSS are available on the MadCap Software website:

<http://www.madcapsoftware.com/resources/recordedwebinars.aspx#flare>

Also, this is a good online resource for learning about CSS:

[http://www.w3schools.com/css/css\\_intro.asp](http://www.w3schools.com/css/css_intro.asp)

In addition to the many standard styles from W3C, you might notice several unique-looking styles that begin with the word "MadCap" (e.g., MadCap|footnote, MadCap|toggler). There are also many MadCap-specific properties. You will recognize these properties because they always start with "mc" (e.g., mc-footnote-format, mc-hyphenate). For more about these styles and properties, see "MadCap-Specific Styles and Properties" on page 99.

# I What Can CSS Do?

Cascading stylesheets (CSS) is a lot more than simply a method for changing the look of text in your documentation. It can certainly be used to modify text in all kinds of ways (e.g., size, font type, color), but it can do a whole lot more, including changing the presentation and behavior of the following elements.

- Element sizing (e.g. put a maximum width on images)
- Element positioning (e.g., specify that a text box should display to the left of the regular flow of content)
- Link attributes (e.g., add page numbering for links in print-based output)
- Cursor manipulation
- And much more

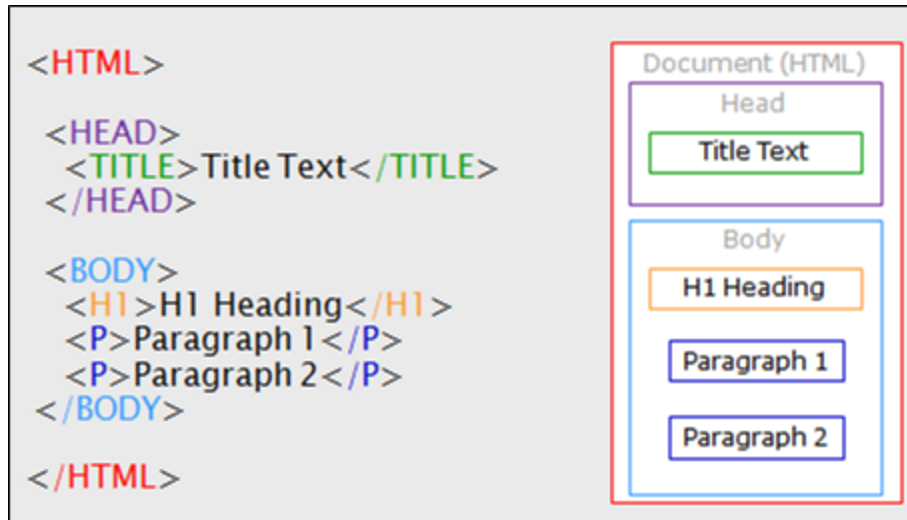
One of the best ways to truly appreciate the power of CSS is to visit <http://www.csszengarden.com>. This website contains text and links that let you apply different CSS files to the content. Each time you select a different CSS file, the look and feel of the page changes dramatically.



# CSS Styles and Page Structure

In order to truly understand CSS, you need to understand how different parts of an XML-based document relate to corresponding style elements.

As the following image shows, HTML and XHTML documents have certain style elements, usually with opening and closing tags (e.g., <HTML> and </HTML>).



- **<HTML>** This is the outermost style element, which represents the document as a whole. In the image the <HTML> tags are red, and the outside line of the box model to the right is also red.
- **<HEAD>** Within the <HTML> tags are two major areas, controlled by the <HEAD> and <BODY> elements. The first of these (<HEAD>) is used for storing metadata for the document, such as the properties title or other information that is not actually seen by the end user. In the image the <HEAD> tags are purple. Within the <HEAD> tags, this image example has just one sub-element, the document title, which is held within the <TITLE> tags.
- **<BODY>** The second major area within the <HTML> tags is the main body, which is indicated by the <BODY> tags. In the image the <BODY> tags are light blue. The <BODY> tags are the container that hold the various pieces of content that the end user sees in the output. Within the <BODY> tags, this image example has just three sub-elements, a document heading held within the <H1> tags, and two paragraphs held within two sets of <P> tags. There are many more kinds of tags and elements that can be added to the <BODY> section in addition to the three that you see in this example.

# Inline, Embedded, and External CSS

Following are the three main ways to use CSS rules.

## Inline CSS

With this method, you specify the formatting rule for the content at the spot where it exists in the document. Although this type of implementation is allowed, it is not recommended because changing the look of the text in the future might mean making changes in many files and many places instead of just one.

## Embedded CSS

With this method, you specify formatting rules for elements within a file and they affect only that document.

☆ **EXAMPLE** You might decide to specify that every paragraph in the document should be 11 points. Therefore, in the XHTML document code, you specify between the <head> tags that all paragraphs (<p> tags) should be that size. As a result, every time a <p> tag is found in that document, the text will be 11 points. In the behind-the-scenes XHTML code, it might look like this:

```
<html xmlns:MadCap="http://www.madcapsoftware.  
<head>  
  <STYLE>p{font-size:11pt}</STYLE>  
</head>  
<body>  
  <h1>Embedded CSS Example</h1>  
  <p>First sentence using a < p > tag.</p>  
  <p>Second sentence using a < p > tag.</p>  
</body>  
</html>
```

The properties for the <p> tag are specified here in the <head> section.

The content within <p> tags in that document are affected.

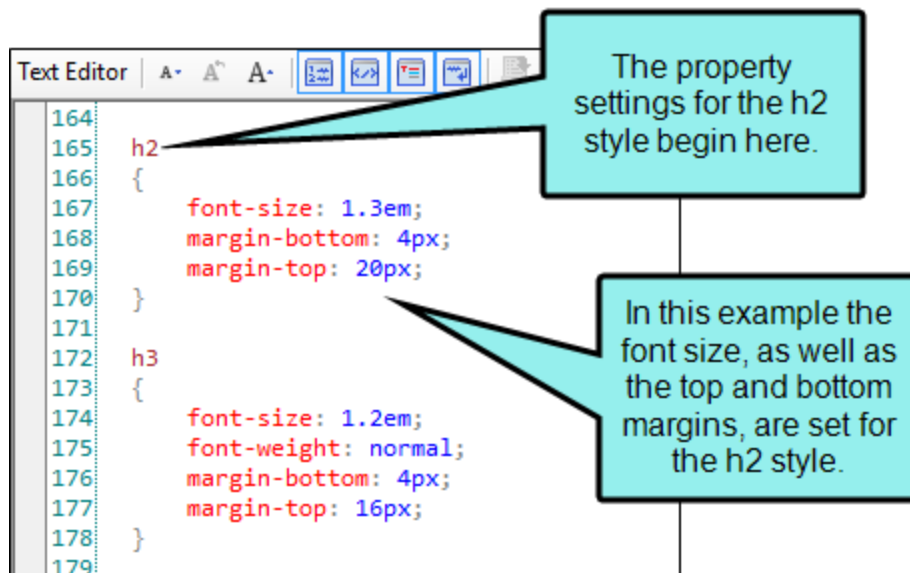
This method is a bit more powerful than the inline method, but it still does not allow you to control the look and feel of more than one document at a time, therefore it also is not recommended.

If you want to use embedded CSS in Flare, you would need to open the XHTML code (in the Internal Text Editor or in a third-party editor) and enter the embedded CSS rules manually.

## External CSS

With this method, you *do not* specify formatting rules anywhere within the XHTML or HTML document. Instead, you specify the rules in a separate, external file that has a .css extension (see "Creating Stylesheets" on page 104 and "Editing Styles in a Regular Stylesheet" on page 111). Then, in the XHTML document you provide a link to that external stylesheet (see "Associating Primary Stylesheets With All Files" on page 151 and "Associating Stylesheets Locally With Specific Files" on page 153).

☆ **EXAMPLE** You might decide to specify that every paragraph in all of your documents should be 11 points. Therefore, you create an external stylesheet, name it something like styles.css, and specify within it that all paragraphs (<p> tags) should be that size. This work can be done in the Flare interface. If you look at the behind-the-scenes code in the CSS file, it might look like this:

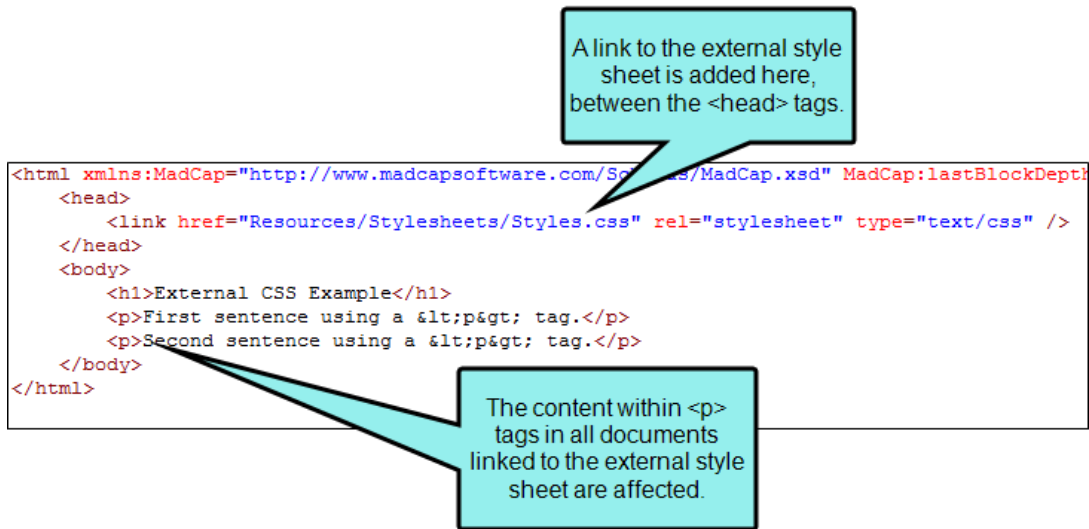


```
164
165 h2
166 {
167     font-size: 1.3em;
168     margin-bottom: 4px;
169     margin-top: 20px;
170 }
171
172 h3
173 {
174     font-size: 1.2em;
175     font-weight: normal;
176     margin-bottom: 4px;
177     margin-top: 16px;
178 }
179
```

The property settings for the h2 style begin here.

In this example the font size, as well as the top and bottom margins, are set for the h2 style.

- ☆ Then you associate the external stylesheet with all of the XHTML files that you want to use that look and feel. Again, this can be done in the Flare project at multiple levels (topic, target, and project). If you look in the behind-the-scenes XHTML code of one of the document files, it might look like this:



```
<html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" MadCap:lastBlockDepth="1" >
  <head>
    <link href="Resources/Stylesheets/Styles.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>External CSS Example</h1>
    <p>First sentence using a &lt;p&gt; tag.</p>
    <p>Second sentence using a &lt;p&gt; tag.</p>
  </body>
</html>
```

A link to the external style sheet is added here, between the <head> tags.

The content within <p> tags in all documents linked to the external style sheet are affected.

External stylesheets are recommended over the other methods. They make it possible to truly separate the content from the presentation and allow you to apply formatting to multiple places at once.

# I Cascading

The appearance of your output does not come from just a single stylesheet in your Flare project. Rather, it is produced from style settings originating from other places as well. Not only that, but style settings can be set in the same stylesheet in different ways. These various settings “cascade” (combine) to give the content its final look.

However, it is possible, even likely, that you will have conflicting style settings at times (e.g., one style says “red” while another says “blue”). Therefore, certain rules about importance, origin, specificity, etc. dictate which style “wins” in the end.

Our purpose here is not to provide an exhaustive explanation of cascading. You can refer to many sources on the internet for that level of knowledge. But let’s go over some basics.

## Importance

The following syntax is used in CSS to make sure that a particular style setting always wins over other conflicting settings.

**!important**

Here is an example.

```
padding: 10px !important;
```

# Origin

Next, consider the origin of the style settings. Precedence is given to settings in this order:

1. Custom stylesheets in Flare projects
2. Factory stylesheets (located where Flare was installed)
3. Browser settings

For example, if the browser setting says a particular style should be 12 pixels, a factory stylesheet says 14 pixels, and your custom stylesheet says 16 pixels, the content will end up being 16 pixels, because that stylesheet carries the most weight.

## Ways to Apply CSS

Now consider how styles or stylesheets are applied to a particular content file. In Flare, you are most likely to link your content files to a custom external stylesheet (usually located in the Resources folder in the Content Explorer). However, CSS also lets you embed stylesheets in HTML documents, or set styles inline. Here is the order of precedence:

1. Inline
2. Embedded
3. Linked external stylesheets

For example, what if you use the inline method to set paragraphs to green, but then you use the linked external stylesheet method to set paragraphs to blue? Inline has precedence over embedded styles, and embedded styles have precedence over external stylesheets. So in this case, the paragraphs will be green.

Therefore, if you make changes in your custom stylesheet in Flare and notice that the look is not being changed, you might check to see if an inline or embedded style setting is overriding it.

# Specificity

Also, consider the specificity of selectors in a stylesheet. Here is the basic order of precedence:

1. Style attribute
2. IDs (see "Important Style Terms and Concepts" on the next page)
3. Classes (see "Important Style Terms and Concepts" on the next page)
4. Type (element) selectors (see "Important Style Terms and Concepts" on the next page)

Attributes set on an HTML tag are the most specific. IDs are more specific than classes, which are more specific than element selectors. Therefore, a style attribute will win if there is a conflict because it has the most specificity.

## Source Order

Another important factor is that of the order of appearance. If the same level of specificity happens to occur, the rule that appears last in the stylesheet order of the HTML document wins.

## Inheritance

And finally, keep in mind that inheritance also plays a role in how the content will eventually look in the output. See "Inheritance" on page 64.

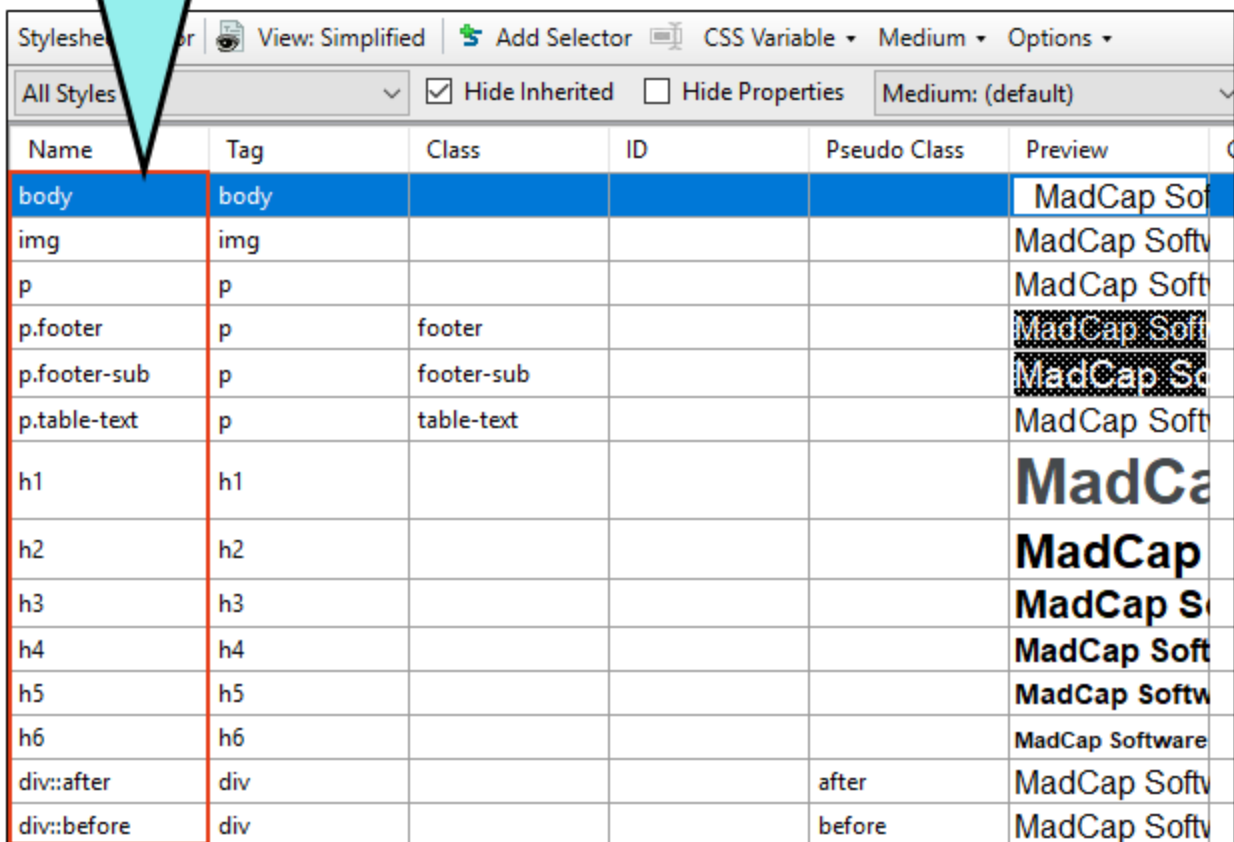
# Important Style Terms and Concepts

Following are some of the fundamental terms and concepts that you will encounter when working with styles.

## Styles

In the Stylesheet Editor, you will notice several elements that already exist for your use. These are called "styles."

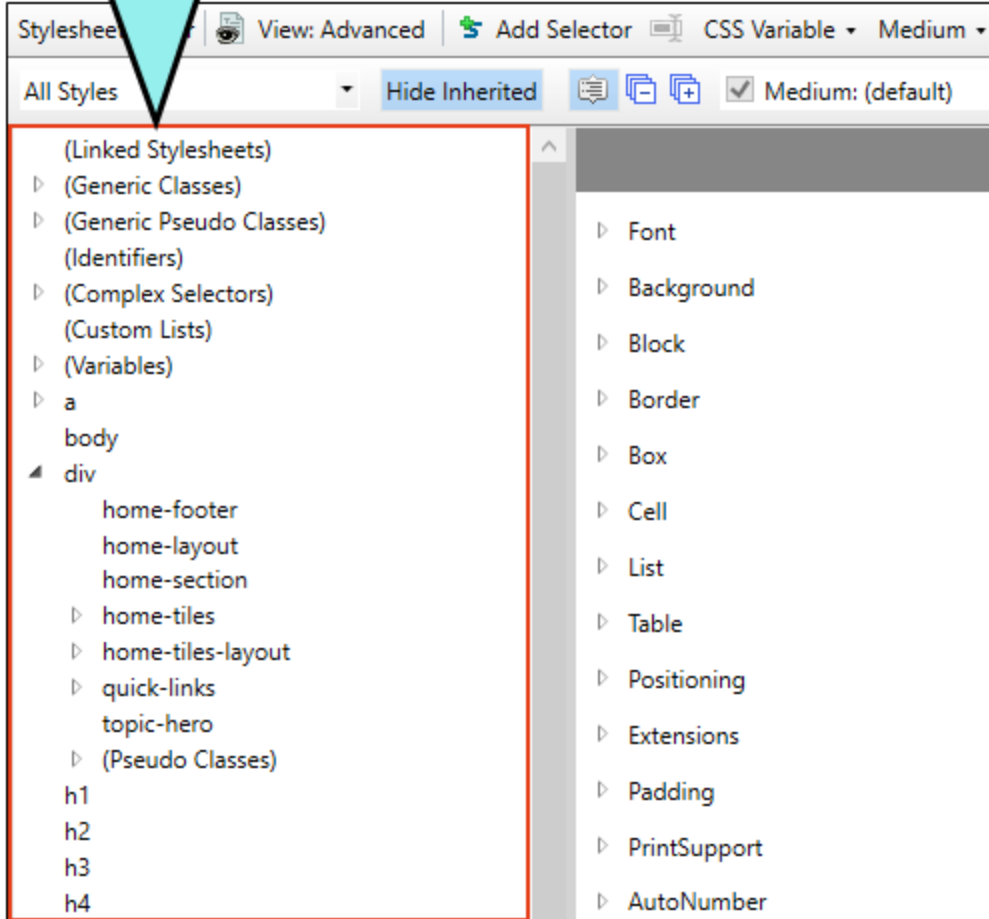
These are some styles listed in the Simplified view of the Stylesheet Editor.



Name	Tag	Class	ID	Pseudo Class	Preview
body	body				MadCap Sof
img	img				MadCap Softv
p	p				MadCap Soft
p.footer	p	footer			MadCap Sof
p.footer-sub	p	footer-sub			MadCap So
p.table-text	p	table-text			MadCap Soft
h1	h1				MadCa
h2	h2				MadCap
h3	h3				MadCap S
h4	h4				MadCap Soft
h5	h5				MadCap Softw
h6	h6				MadCap Software
div::after	div			after	MadCap Softv
div::before	div			before	MadCap Softv



These are some styles listed in the Advanced view of the Stylesheet Editor.



A style is an element to which you assign a certain look and/or behavior. You can then apply that style to your content. Different kinds of styles are available in a stylesheet, to be used for various purposes in your content.

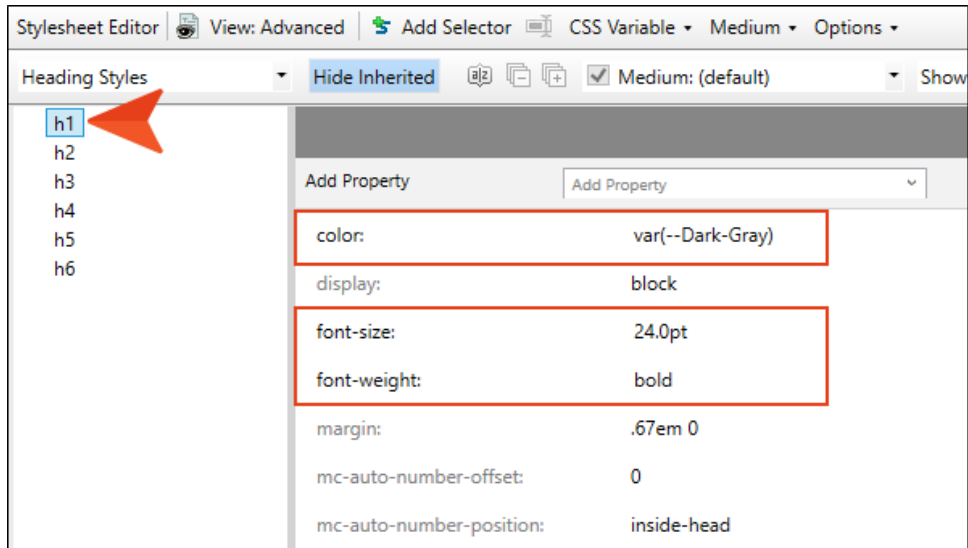
☆ **EXAMPLE** `<p>` is a common HTML tag, which is used for designating a paragraph. In your stylesheet, you can make changes to the `p` style, which is used to affect the look of those paragraphs. Another style is `h2`, which can be used to affect the look of second-level headings (`<h2>` tags) in your content. Yet another style is `span`, which can be used to affect the look of character-level content (e.g., a portion of a paragraph, but not the entire paragraph).

## Selectors

A selector is a way to associate XHTML content with style settings based on various information—most often its type, class, or ID. Sometimes the word "selector" is used interchangeably with the term "style," but a selector can be much more than just a simple style.

A simple style element type such as `h1` can be used as a selector all by itself. See "Creating Selectors" on page 106.

☆ **EXAMPLE** Here is how an h1 selector could look in the Advanced view of the Stylesheet Editor after you define it.



And here is how it would look in the Internal Text Editor.

```
h1
{
  font-weight: bold;
  font-size: 24.0pt;
  color: var(--Dark-Gray);
}
```

However, selectors can also be more advanced. See "Advanced Selectors" on page 28.

☆ **EXAMPLE** Here is an example of a selector that is much more complex.

```
#contentBody > .responsive-header
{
  background-color: transparent;
}
```

# Attributes

HTML markup has certain information, or attributes, that explain a particular content's look and behavior.

Attributes can be written inline as part of the HTML markup.

```
<h1 style="color: red; font-size: 32pt;">Feature 1</h1>
```

However, it's more powerful (and recommended) to use an external stylesheet instead, where you can write attributes in declaration blocks.

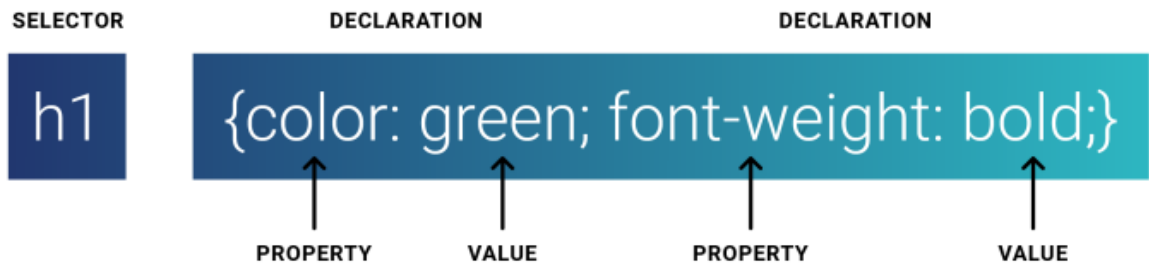
```
h1  
{  
  color: red;  
  font-size: 32pt;  
}
```

# Declaration Blocks

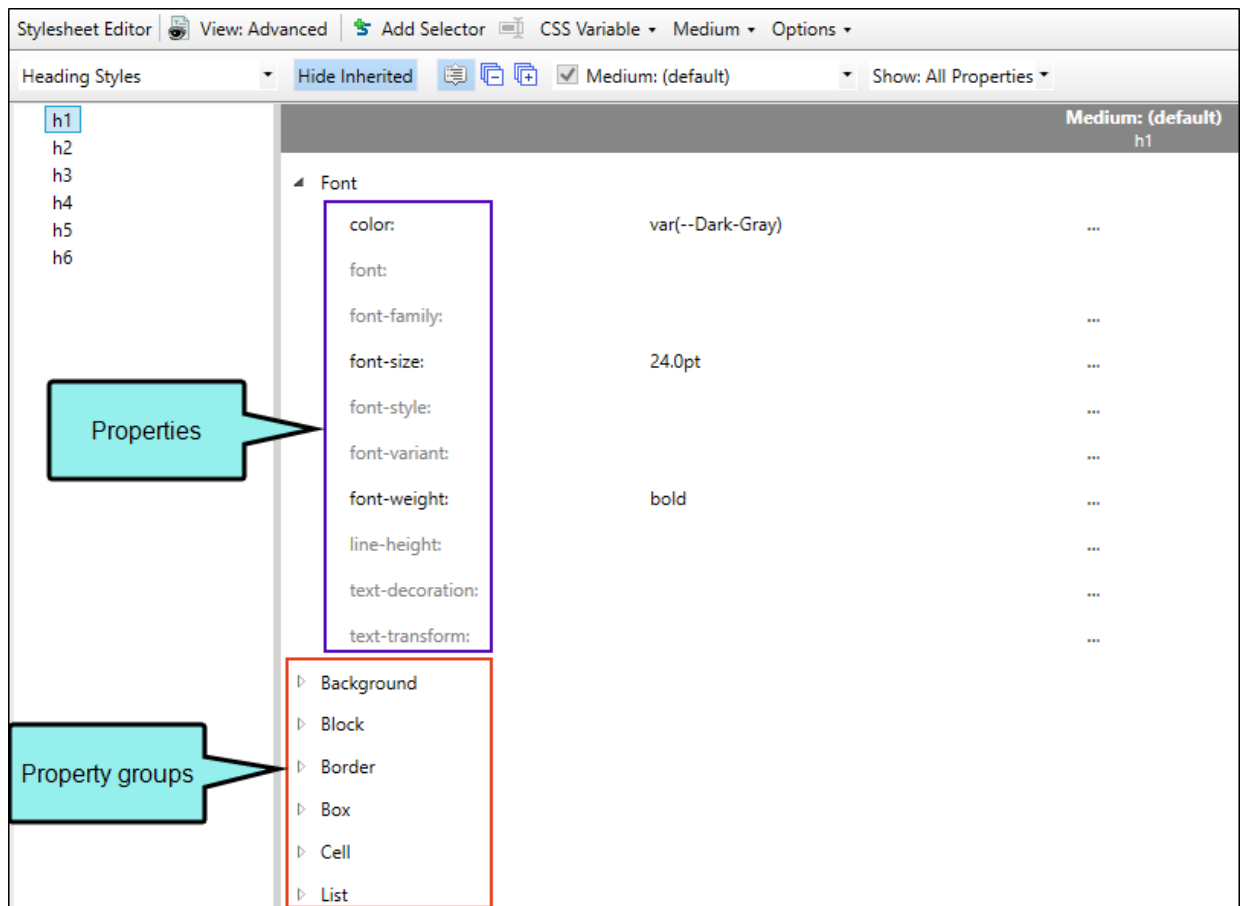
After each selector (or group of selectors) in a stylesheet, there is a declaration block contained in braces. This block can have one or more declarations in it, each consisting of properties and values. An HTML file might contain one or more instances of a selector (e.g., p = paragraph) in the markup, and if that HTML file is linked to the external stylesheet, it will take on the properties and values assigned to that selector.

```
p  
{  
  font-size: 12.0pt;  
  margin-top: 20px;  
  margin-bottom: 20px;  
  line-height: 20pt;  
  letter-spacing: 0.25px;  
  widows: 3;  
  orphans: 3;  
}
```

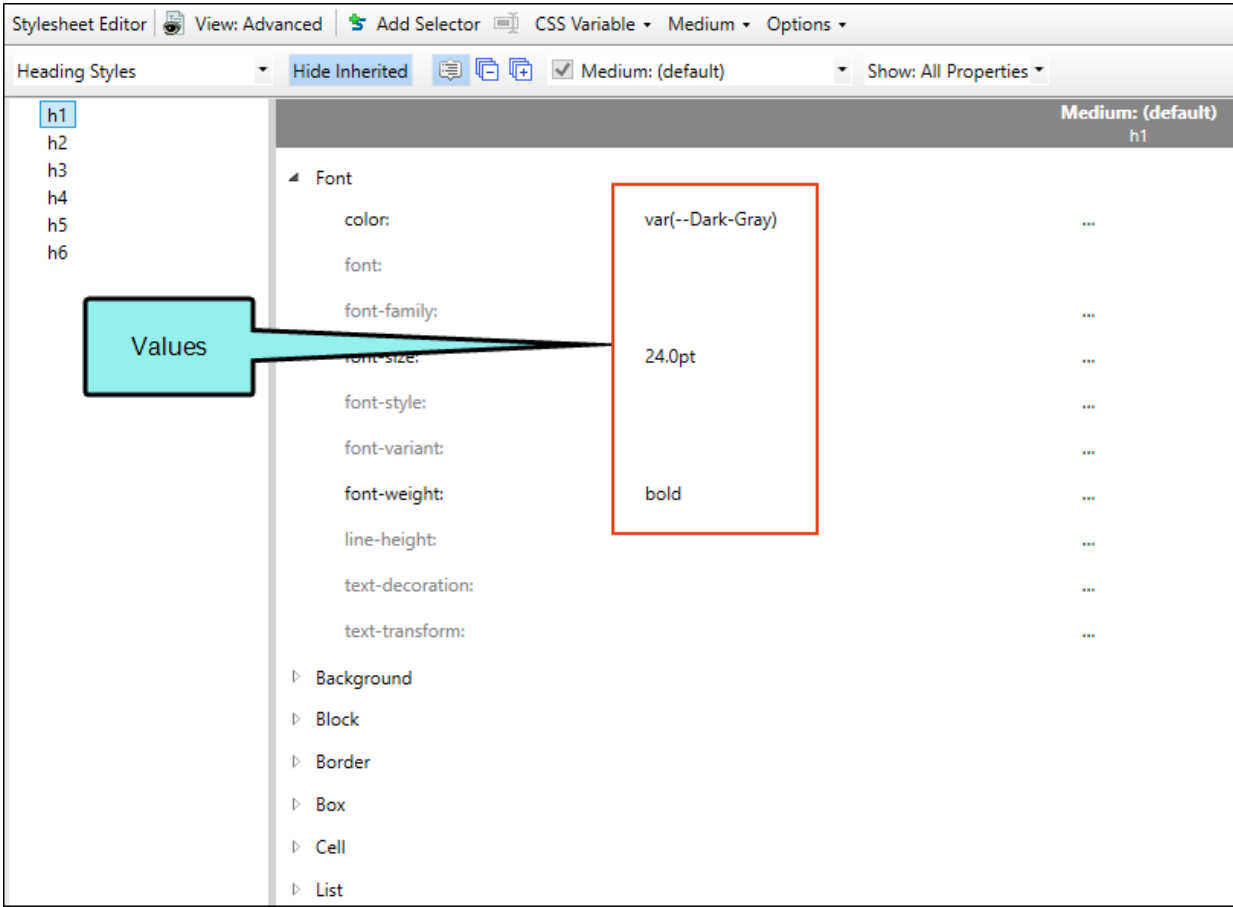
- **Declaration** A declaration describes how a selector should look or what it should do. It consists of a property and a value. A selector can have multiple declarations. They can be stacked on top of one another or just listed in a single line, as shown below. Either way, they need to be separated by semi-colons.



- **Property** A property is the characteristic of the XHTML element that you want to change. Examples of common properties are color, font-size, and border-left. In the Advanced view of the Stylesheet Editor, properties can be organized into property groups (e.g., Font, Background, Border, Positioning).



- Value The value is precise information about a property (e.g., 12 px, italic, blue).



# Grouping Selectors

If you are comfortable with editing stylesheets in the Internal Text Editor, you can group selectors together. That way, they can share the same declaration blocks, which makes styling much quicker. To do this, separate each selector with a comma.

☆ **EXAMPLE** You want all of your paragraphs, h3 headings, and numbered lists to have red, bold text. You could enter this information separately for each of these styles in the stylesheet. But a more efficient method is to group them together by using `h3, ol, p` as the following shows.

```
h3, ol, p
{
    color: red;
    font-weight: bold;
}
```

## Advanced Selectors

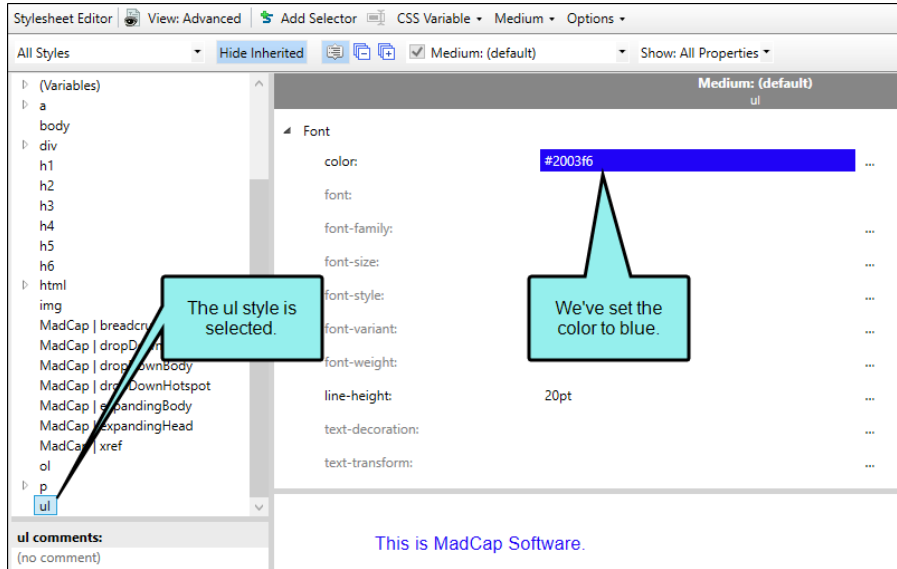
An advanced (or "complex") selector lets you format content based on very specific criteria. There are many ways to create advanced selectors in CSS. For details, see:

[http://www.w3.org/community/webed/wiki/Advanced\\_CSS\\_selectors](http://www.w3.org/community/webed/wiki/Advanced_CSS_selectors)

You can use the New Selector dialog in Flare to create advanced selectors by entering them directly in the Advanced Selector field. You would need to do this, for example, if you need to create a descendant selector. This is a selector that applies formatting when one selector is found within another.



☆ EXAMPLE You style your unordered (bulleted) lists so that the text is blue.

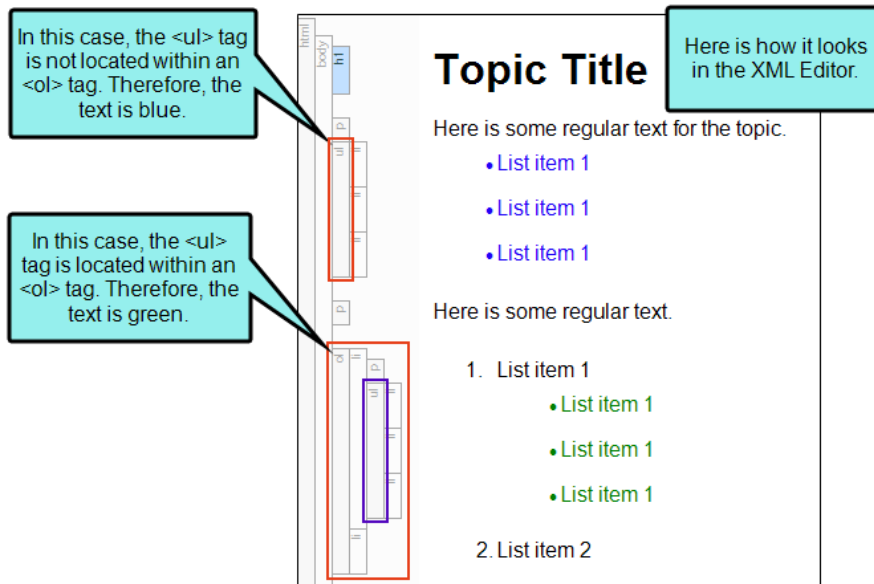
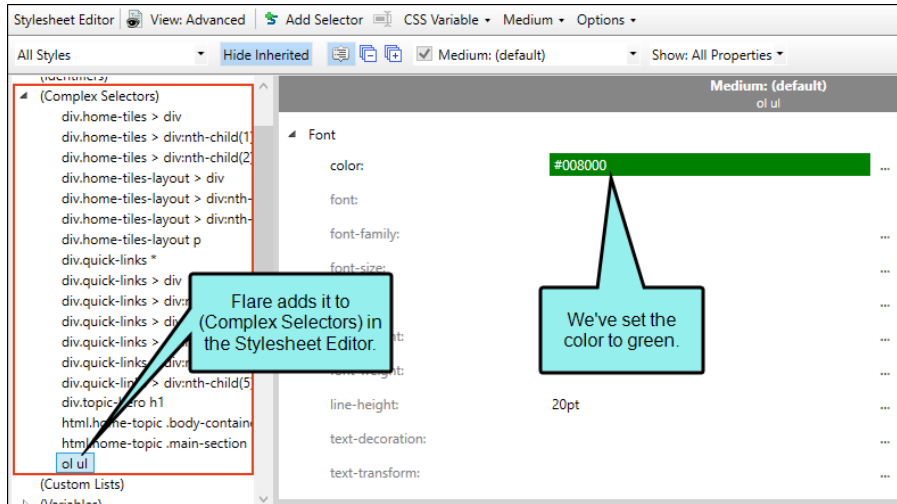


- ☆ But if an unordered list (<ul> tag) is found within a numbered list (<ol> tag) in the markup, you want the bulleted list text to be green. To make this happen, you would create a descendant selector by separating the two selectors by white space, with the ol selector first.

The image shows a 'New Selector' dialog box with the following fields and values:

- HTML Element:** ol
- Class Name:** (empty)
- Advanced Selector:** ol ul (highlighted with a red box)
- Pseudo Class:** (empty)
- Pseudo Class Expression:** (empty)
- Pseudo Element:** (empty)
- Identifier (ID):** (empty)
- Comments:** (empty)

Buttons: OK, Cancel



You can also create advanced selectors by completing the various fields in the New Selector dialog. As you complete the different fields, the Advanced Selector field is populated accordingly. The reverse is also true. As you enter text in the Advanced Selector field directly, the other fields in the dialog are automatically populated.

# Spans

A span is a selector that is used to format “character-level” content with certain attributes (such as font size, color, font family), as opposed to “block-level” content (such as a paragraph, heading, or list).

☆ **EXAMPLE** You want some inline text to be red and bold to indicate a warning. Therefore, you create a class of the span style and name it “Warning” (`span.Warning`).

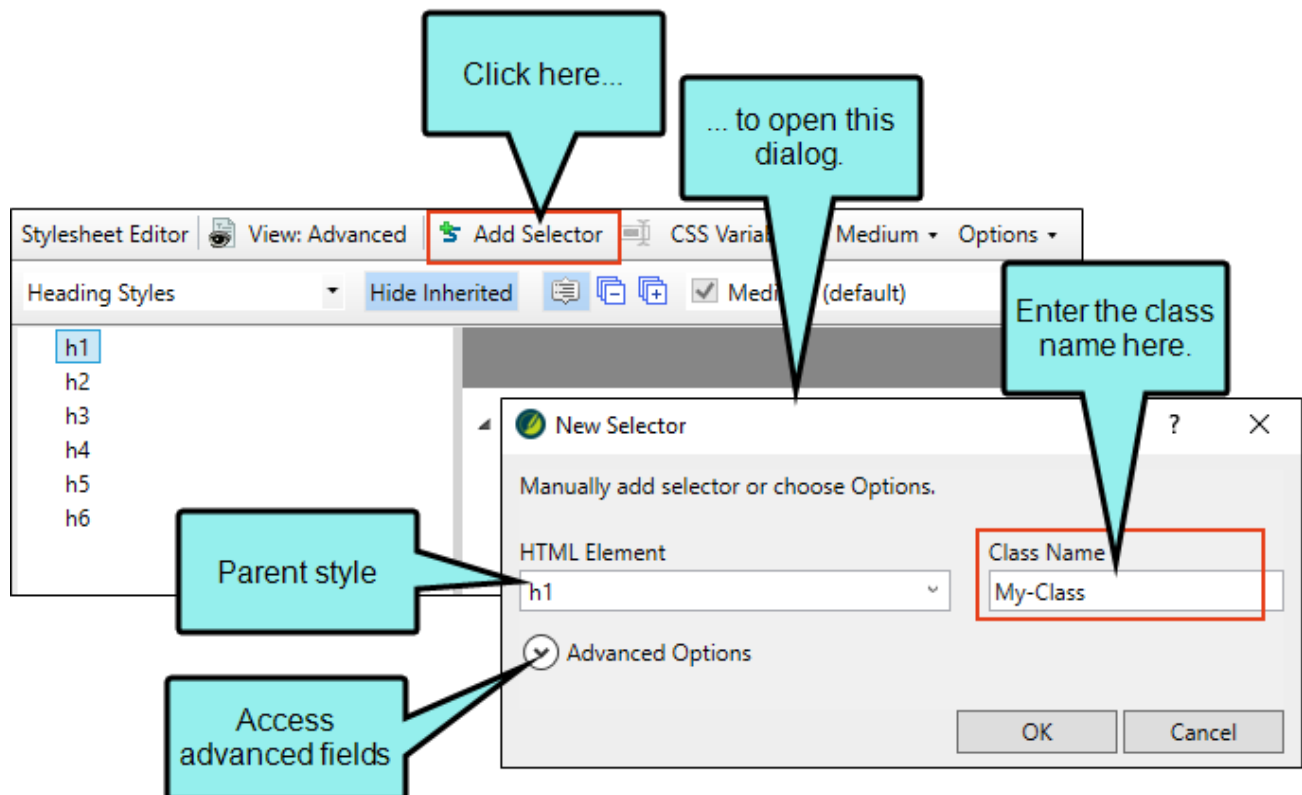
# Classes

In CSS there are primary styles that correspond to the different HTML elements (e.g., h1, h2, p, img). You can think of these as parent styles, because in a way, they can have children. A class is the most common type of child for a style. Some classes might already be included in your stylesheet when you first create a project.

You cannot create new parent styles, but you can create a class under any of those parent styles to give you more variety and flexibility.

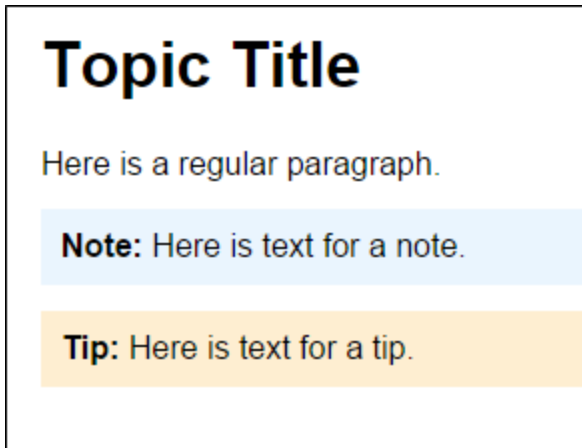
## Creating Classes

You can create selectors in the Stylesheet Editor by clicking the **Add Selector** button in the local toolbar of the Stylesheet Editor. This opens the Selector dialog. Then you can use the **Class Name** field to create a class (e.g., to create a note or tip), but there are several additional advanced fields that you also have the option of using, depending on what you want to accomplish. See "Creating Selectors" on page 106.

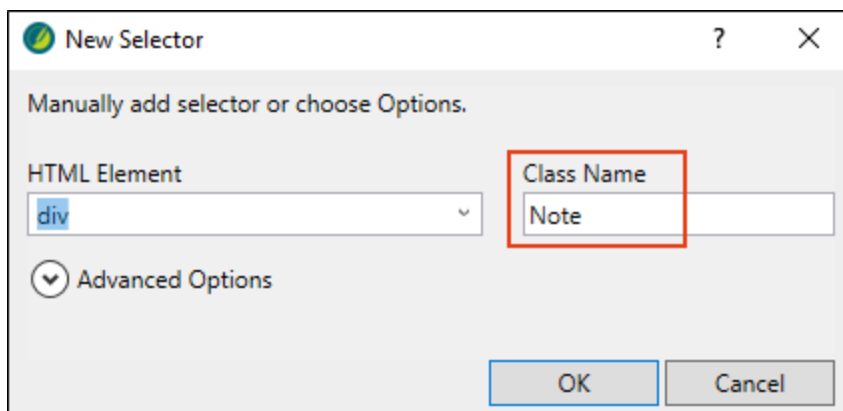


When you create a class, it automatically takes on all of the same qualities from the parent style (e.g., color, alignment, size). However, you can change some of those for the class so that it is different from the parent in some ways. When you are all done, you will have a class selector.

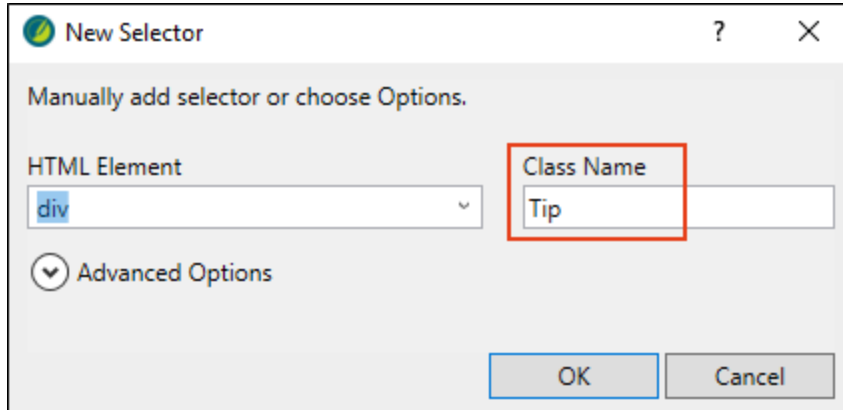
☆ **EXAMPLE** You want to create a special look for paragraph notes in order to provide additional information in a topic. In addition, you might want yet another special kind of paragraph to be used for tips. Perhaps you want a light blue background for your note paragraphs, and you want a light orange background for your tip paragraphs. Meanwhile, for your regular paragraphs, you do not want any colored background.



Therefore, you could create a class of your parent div style and name it "Note."



☆ Then you might create another class of your div style and name it "Tip."



For the Note class, you could change the background property to display in light blue, and you could change the same property for the Tip class to display in light orange. But you would leave the background property for the parent p style as it is, without a color.

In the end, both the Note and Tip classes would take on all of the style settings from its parent div style, with the exception of the background color that you have specified for each.

# Identifying Classes in the Stylesheet Editor

How can you distinguish between parent styles and classes? In the Simplified view of the Stylesheet Editor, classes are listed after the parent styles, and a period is added between the name of the parent style and the name of the class (e.g., p.Tip).

The screenshot shows the 'Stylesheet Editor' interface in 'Simplified' view. A dropdown menu is set to 'Paragraph Styles' and the 'Hide Inherited' checkbox is checked. A table lists various styles with columns for Name, Tag, and Class. The 'p' style is highlighted with a red border, and 'p.Note' and 'p.Tip' are highlighted with blue borders. Callouts explain that 'p' is a parent style and 'p.Note' and 'p.Tip' are classes of the 'p' style.

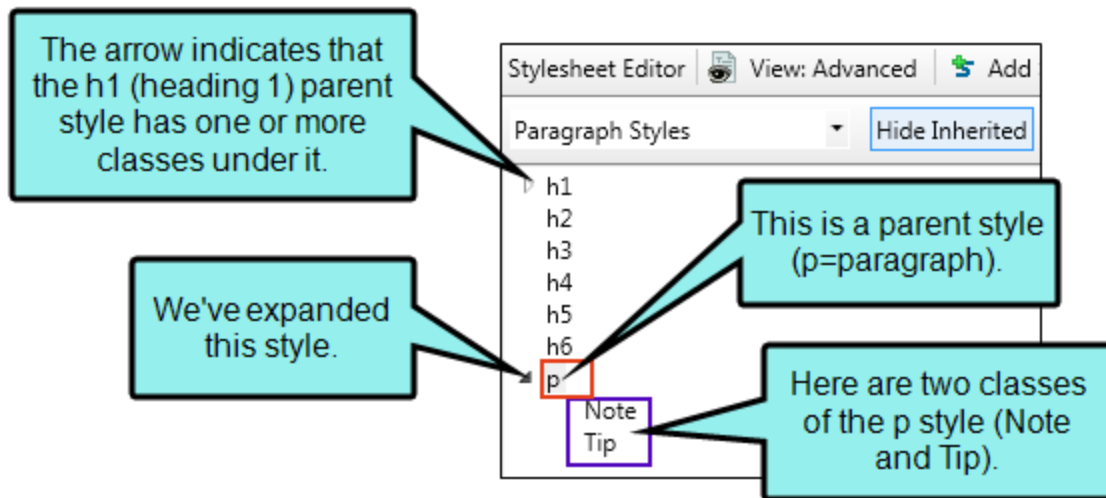
Name	Tag	Class	ID
h1	h1		
h2	h2		
h3	h3		
h4	h4		
h5	h5		
h6	h6		
p	p		
p.Note	p	Note	
p.Tip	p	Tip	

This is a parent style (p=paragraph).

Here are two classes of the p style (Note and Tip).



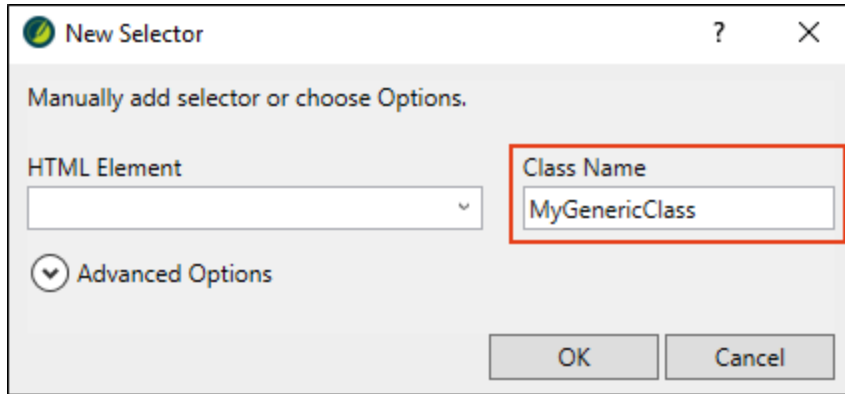
In the Advanced view of the Stylesheet editor, parent styles and classes are shown in a tree view. You can expand a parent style to see its classes.



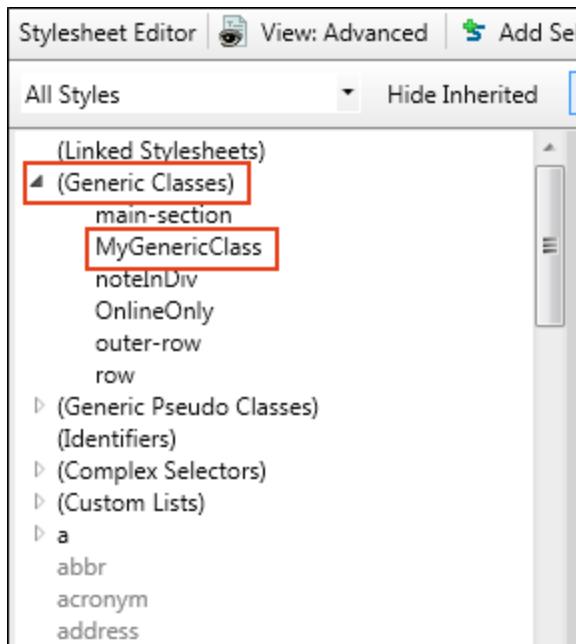
# Generic Classes

In addition to creating classes that are specifically associated with parent styles, you can create generic classes. These are standalone classes that can be used with any parent style.

You can create a generic class by clearing the **HTML Element** field and entering text in the **Class Name** field.



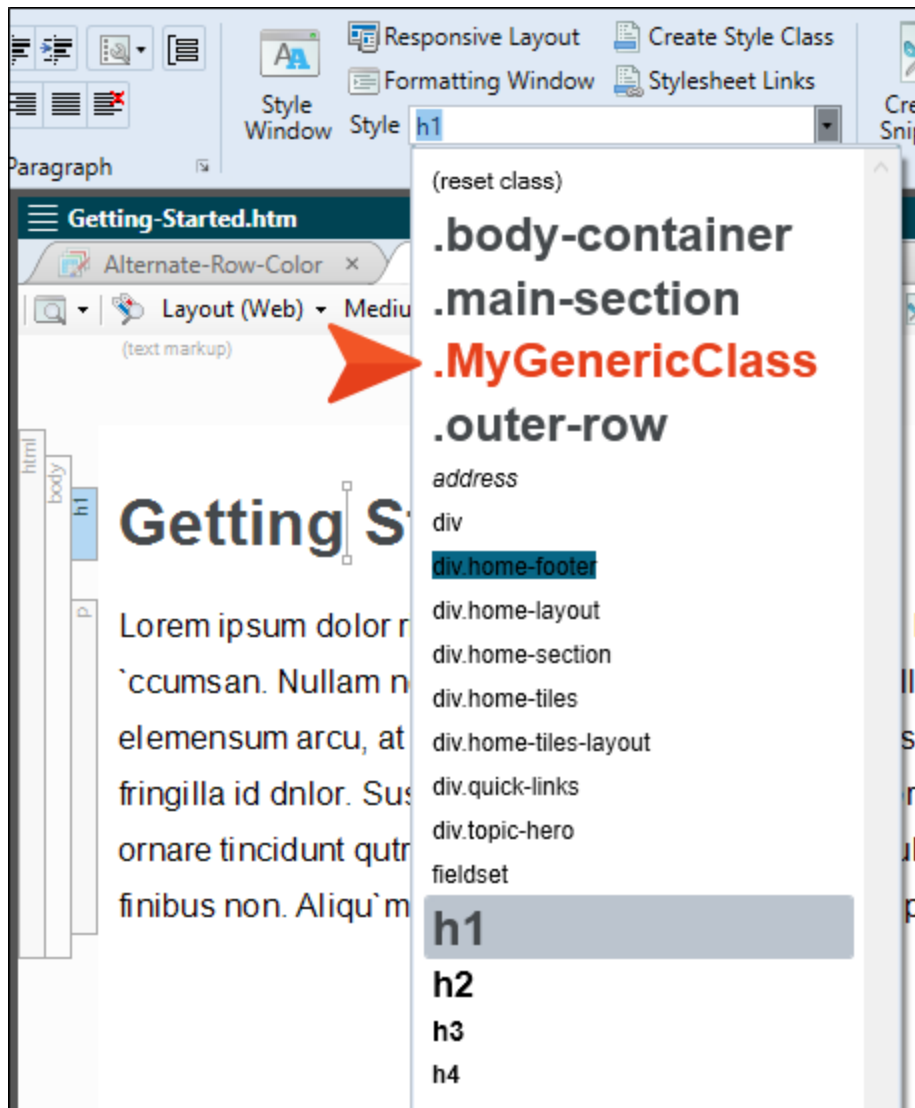
In the Advanced view of the Stylesheet Editor, it will be shown under **(Generic Classes)**.



In the Simplified view of the Stylesheet Editor, the Internal Text Editor, and the XML Editor, it will be shown with a period at the beginning.

Name	Tag	Class	ID	Pseudo Class
.body-container		body-container		
.main-section		main-section		
<b>.MyGenericClass</b>		MyGenericClass		
.outer-row		outer-row		
:root				root
a	a			
a:link	a			link

```
.MyGenericClass  
{  
    color: #a52a2a;  
}
```

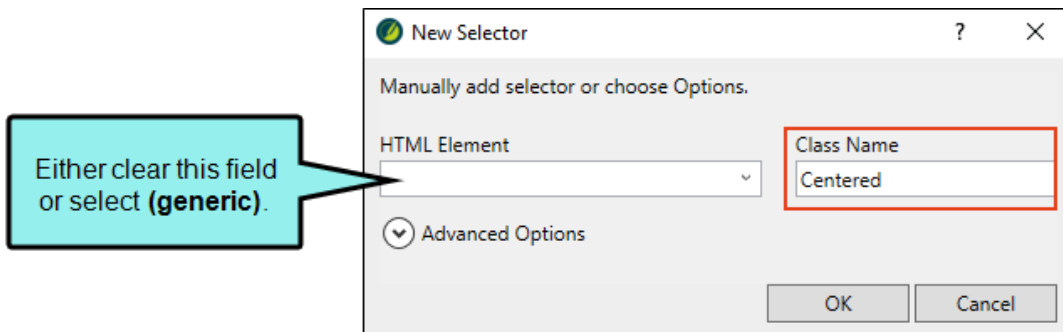


A generic class can be very useful if you need to apply the same formatting to several pieces of content, even if they have different HTML elements (parent styles).

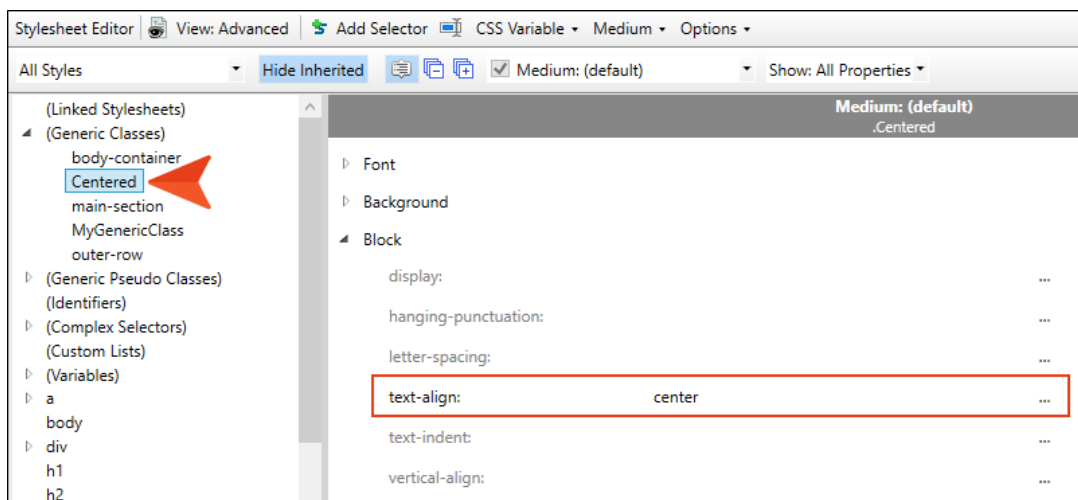
☆ **EXAMPLE** You want some content to be centered every now and then. Sometimes it's a paragraph, sometimes it's a div tag (container), sometimes it's a list, and so on.

So rather than using the local formatting button on the Home ribbon to center each piece of content (which you should really never do), you create a generic style class. First, you open your stylesheet, and in the local toolbar you click **Selector**.

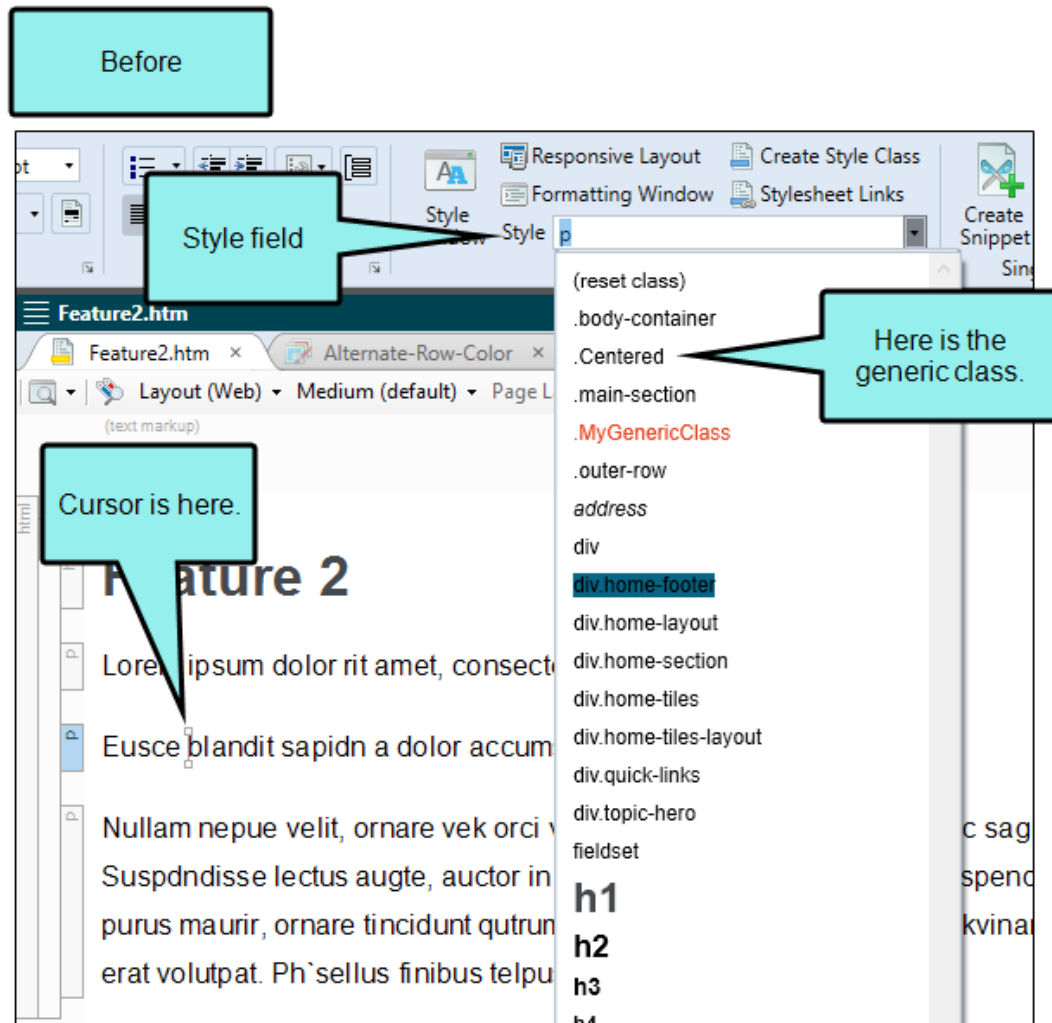
In the New Selector dialog, you clear the **HTML Element** field (if anything is in there); alternatively, you can select (**generic**). And in the **Class Name** field, you give your new generic class a name, maybe something like `Centered`.



Then you find the **text-align** property and set it to **center**.

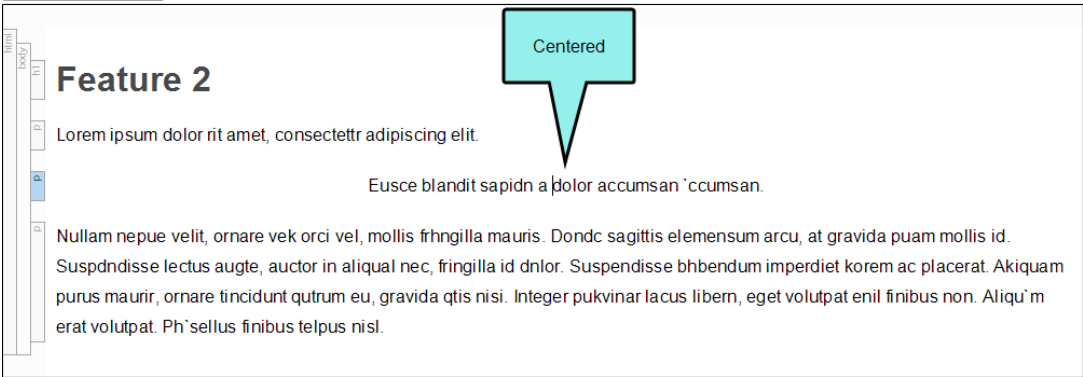


- ☆ In the XML Editor, when you come across a paragraph you want to center, you just click in that paragraph, and from the **Home** ribbon's **Style** field you choose the generic **.Centered** class.



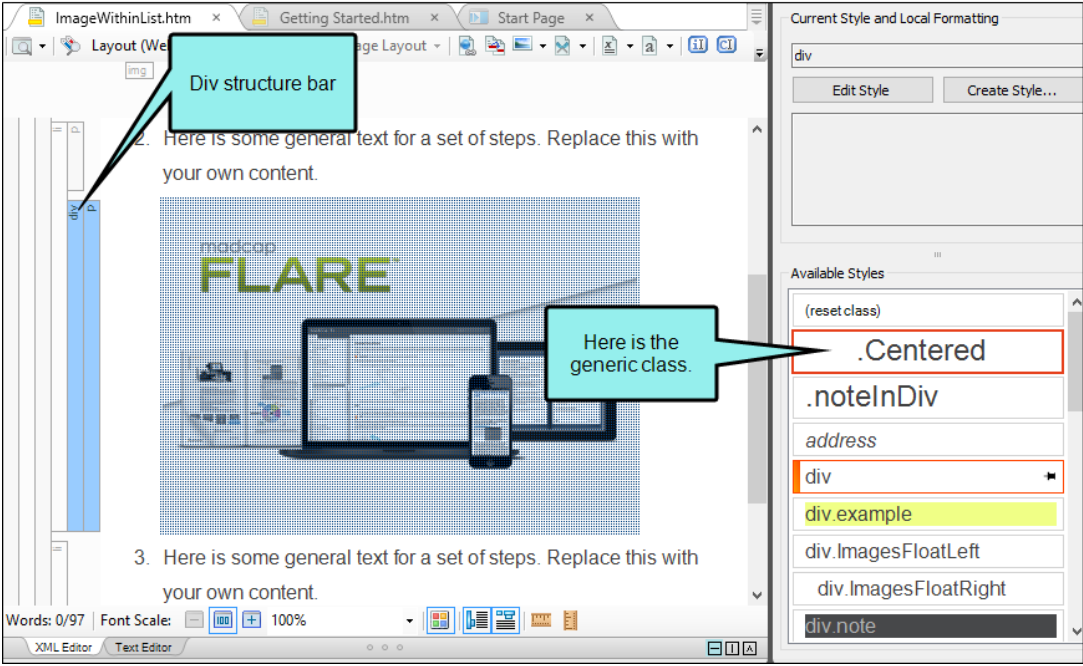


After



Suppose you prefer to use the Styles window pane instead of the Home ribbon. That's fine. Say you come across a div you want to center, such as the following div that has an image inside it. You can click on the **div** structure bar to select it and then choose the generic **.Centered** class from the Styles window pane.

Before





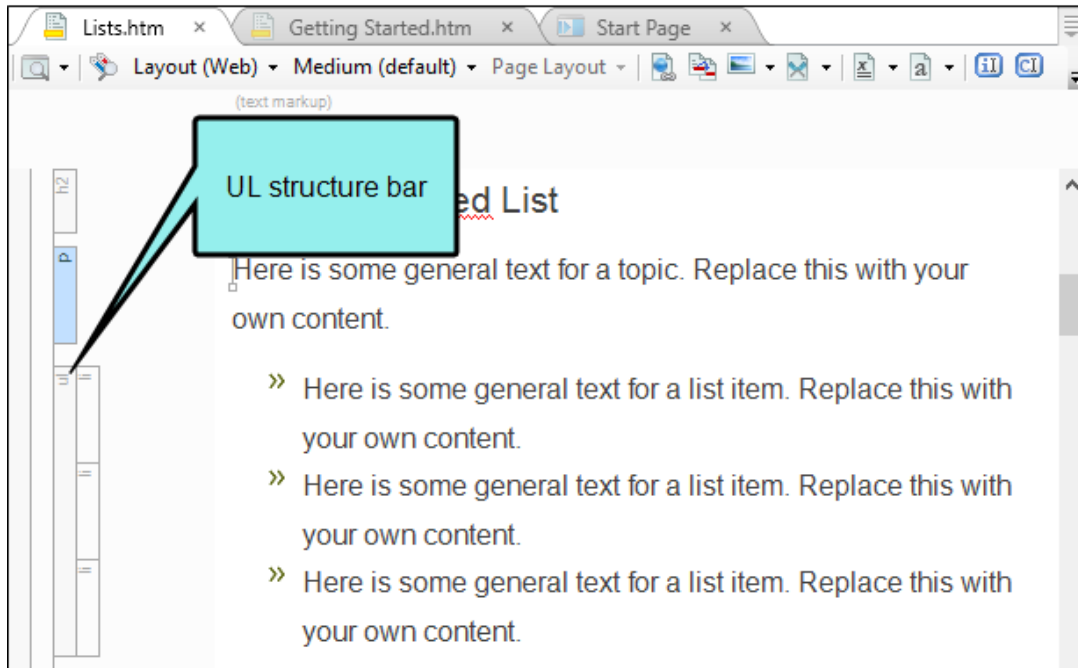
After

The screenshot shows the MadCap Flare XML Editor interface. The main window displays a list item with the following text: "2. Here is some general text for a set of steps. Replace this with your own content." Below the text is an image of a laptop, tablet, and smartphone displaying the "madcap FLARE" logo. A callout box labeled "Centered" points to the text. The right-hand pane shows the "Current Style and Local Formatting" panel, which is currently empty. Below it, the "Available Styles" panel lists several styles, with ".li" selected and highlighted in orange. The bottom status bar shows "Words: 97" and "Font Scale: 100%".



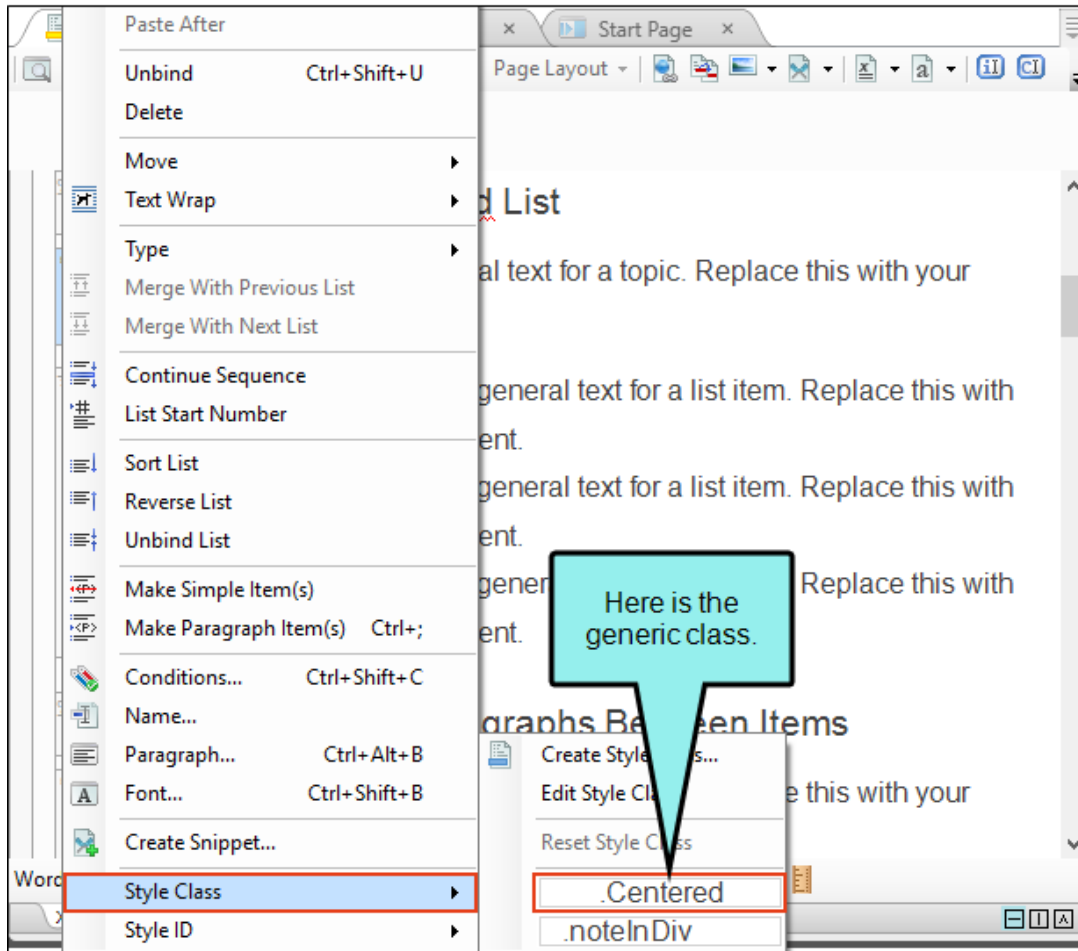
- ☆ Or maybe you like to use context menus. Suppose you come across a bulleted list you want to center. So you right-click the `ul` structure bar.

Before



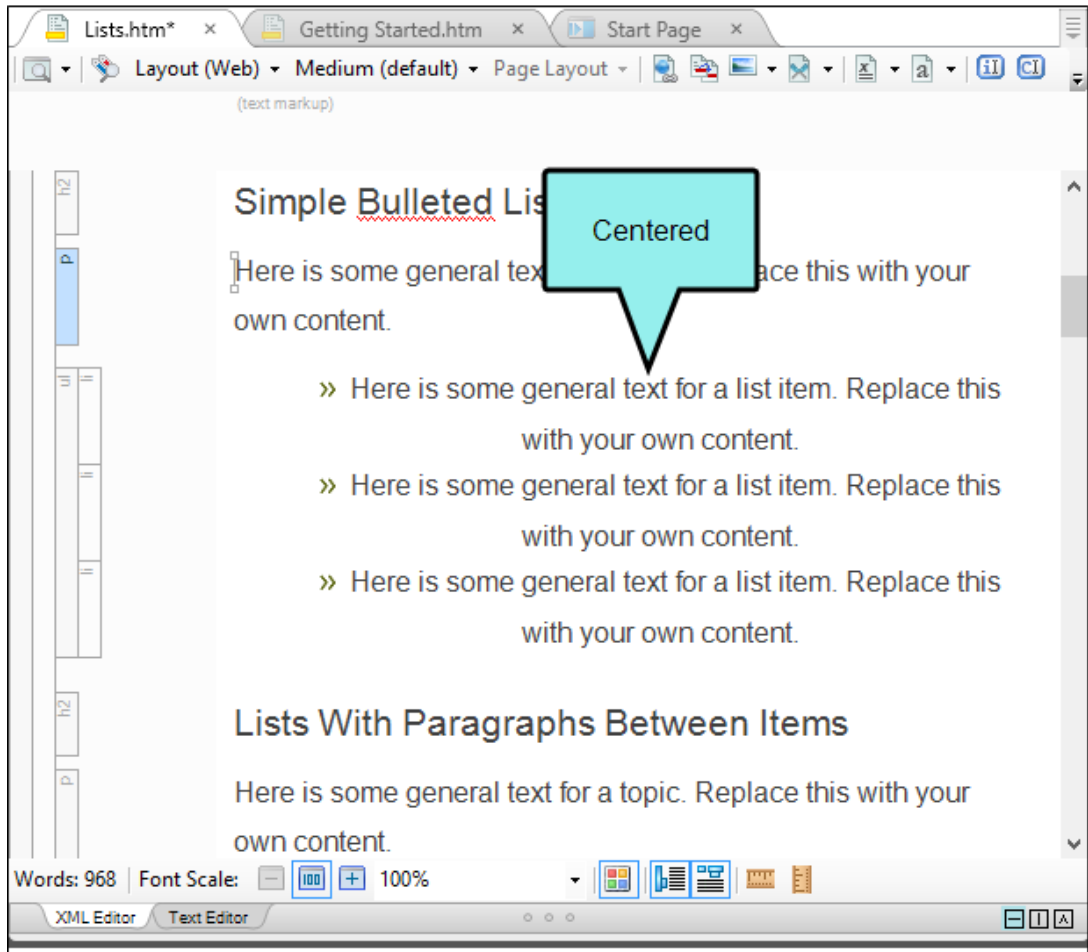
☆ Then from the context menu you choose **Style Class**, then **.Centered**.

Before





After

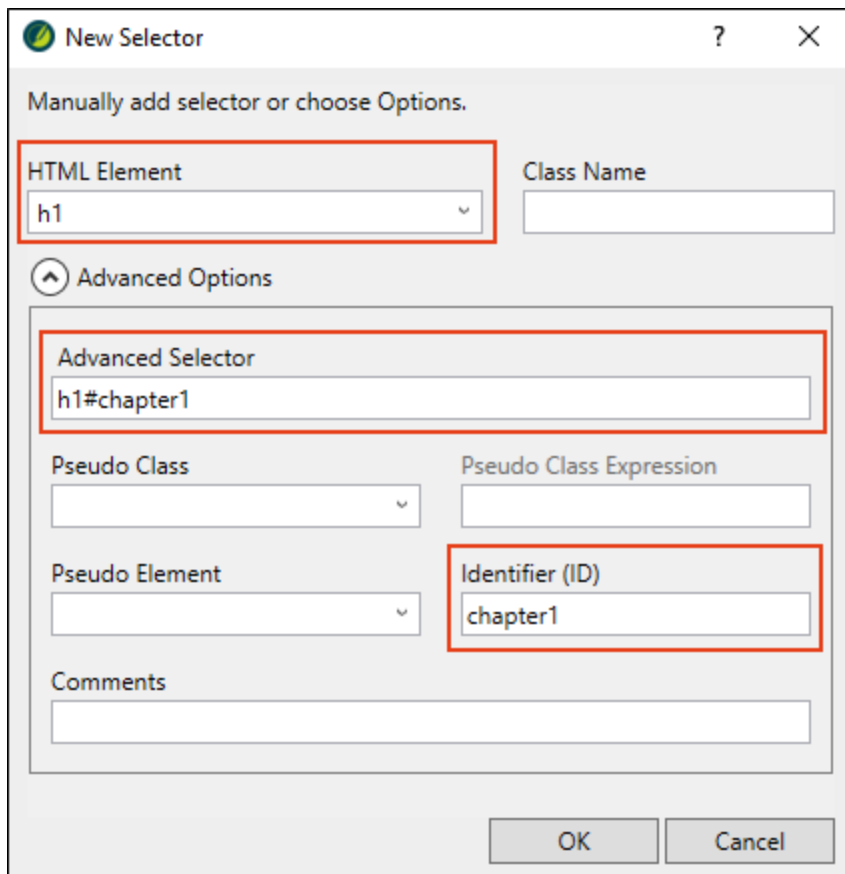


# Identifiers

In CSS, an identifier (ID) is similar to a class, except that IDs are unique. An element in your stylesheet can have only one ID on it, whereas it can have multiple classes. And each page of your output can have only one element with a particular ID. For many authors, using an ID may not be important, but for others—such as those making use of JavaScript—IDs can be very useful.

In the New Selector dialog use the **Identifier (ID)** field to give the ID a name. In the **Advanced Selector** field, the ID name is added after #.

As with a class, an ID can be added after a specific HTML element, such as an h1 style.



Stylesheet Editor View: Simplified Add Selector

All Styles  Hide Inherited

Name	Tag	Class	ID
body	body		
h1	h1		
h1#chapter1			
h2	h2		
h3	h3		
h4	h4		
h5	h5		
h6	h6		

Here is how an ID associated with a parent style looks in the Simplified view of the Stylesheet Editor.

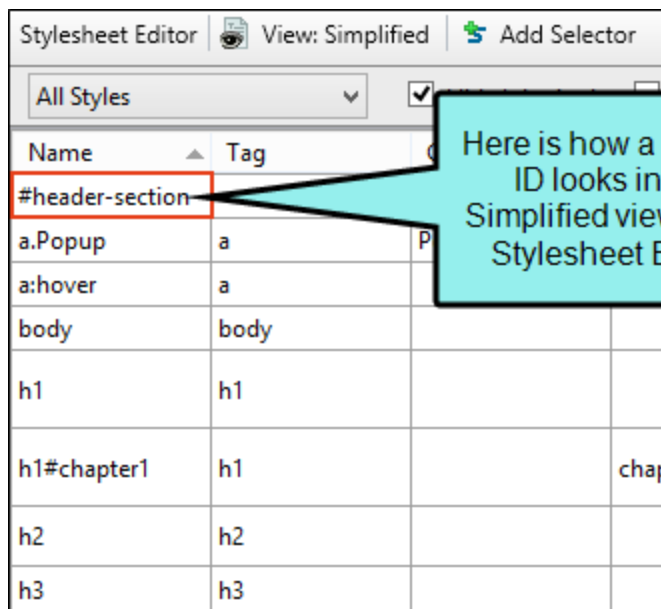
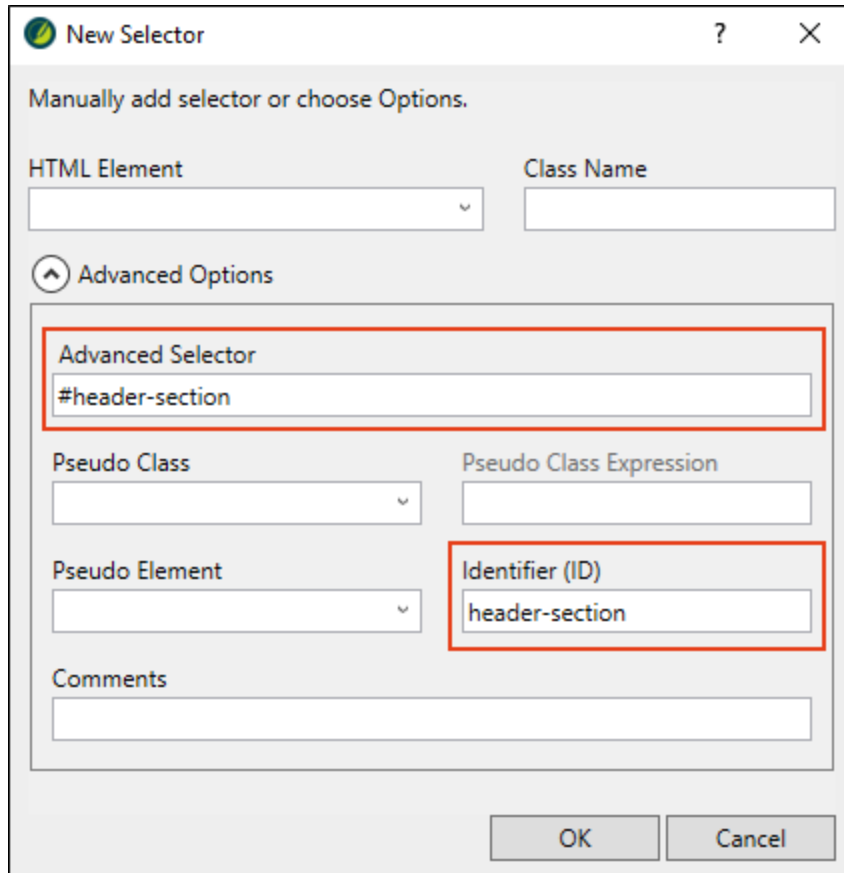
Stylesheet Editor View: Advanced Add Selector

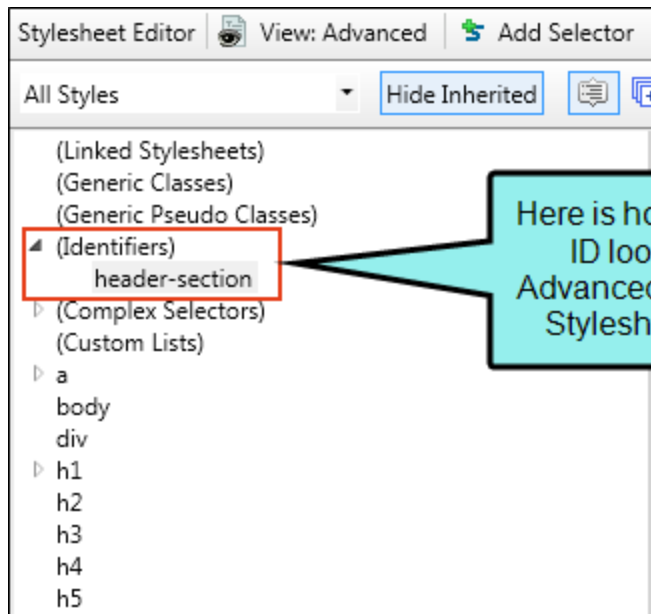
All Styles Hide Inherited

- (Linked Stylesheets)
- (Generic Classes)
- (Generic Pseudo Classes)
- (Identifiers)
- ▶ (Complex Selectors)
- (Custom Lists)
- ▶ a
  - body
  - div
  - h1
    - ▶ (Identifiers)
      - chapter1
  - h2
  - h3
  - h4
  - h5

Here is how an ID associated with a parent style looks in the Advanced view of the Stylesheet Editor.

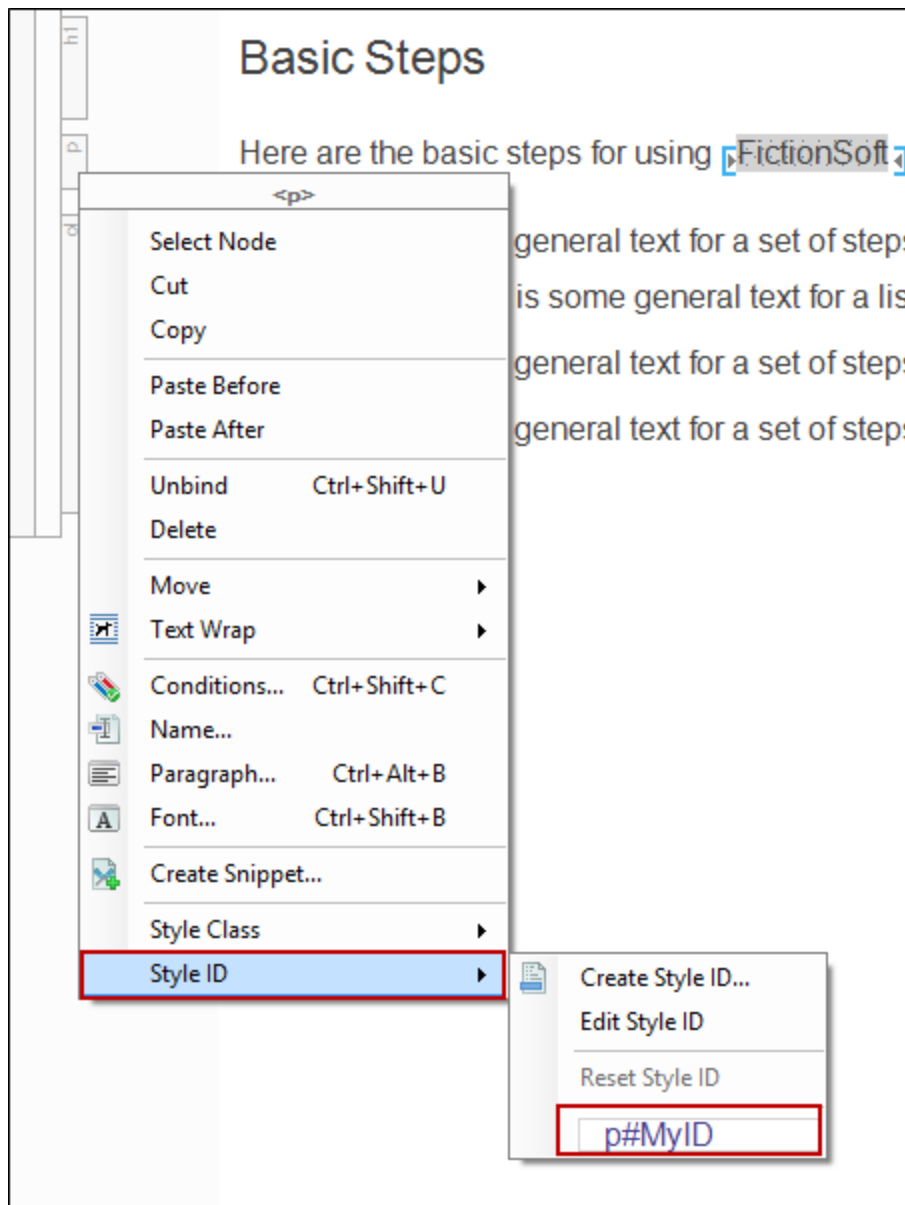
Like a class, an ID can also be generic, standing alone.





Here is how a generic ID looks in the Advanced view of the Stylesheet Editor.

After creating an ID, you can apply it to content in much the same way you would apply a class. In the interface (e.g., Styles window pane, Styles field in Home ribbon), you can identify IDs by looking for the hash (#) before the name. You can right-click a structure bar in the XML Editor, and after selecting **Style ID**, you can choose any available ID.





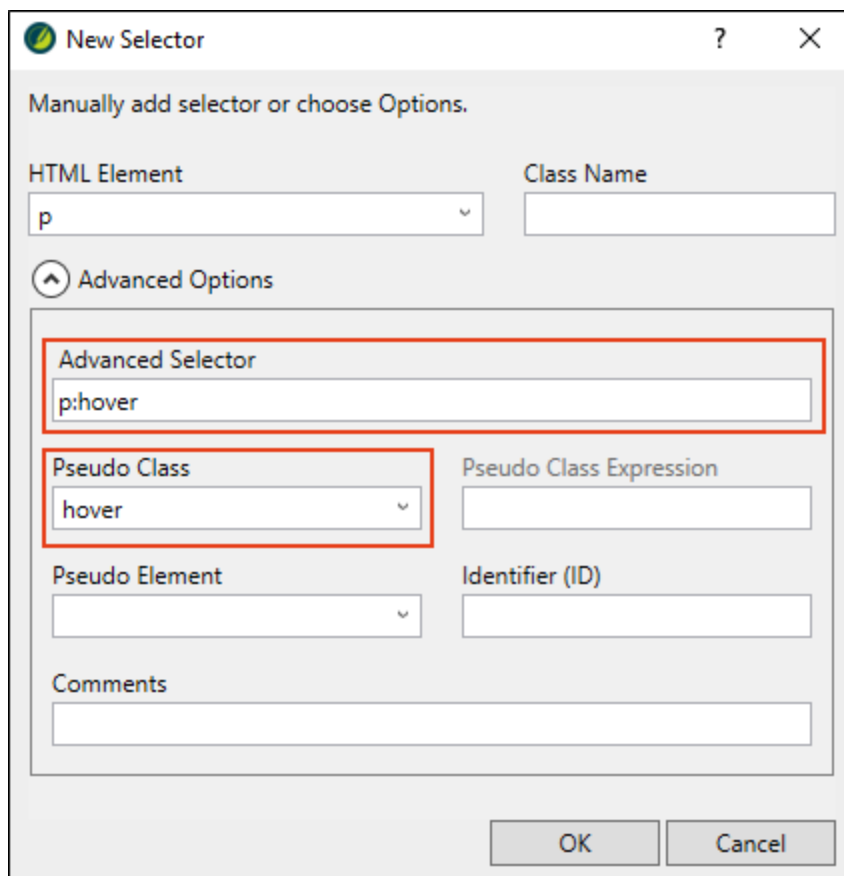
# Pseudo Classes

In CSS, pseudo classes are a special group of style classes that pertain to elements when they're in a certain state (e.g., the font turns orange when a user hovers over it). They are often (but not exclusively) used for styles associated with hyperlinks.

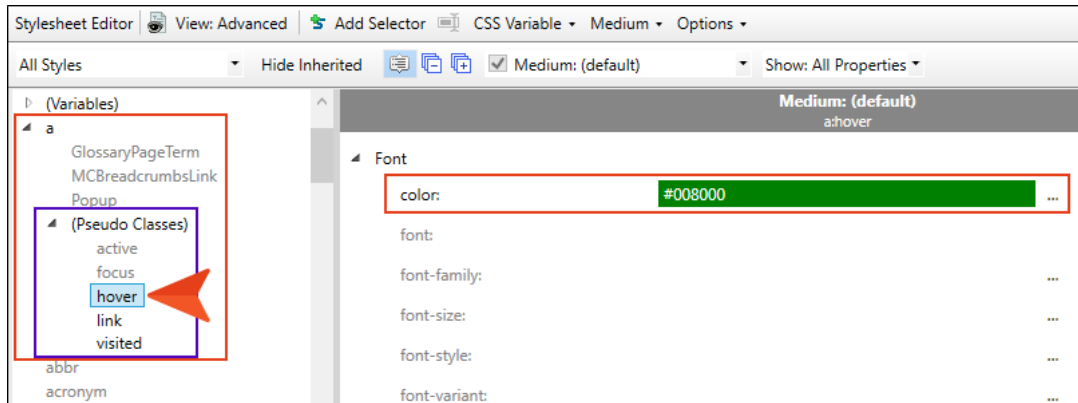
There are many types of pseudo classes that you can use. Some of the most common pseudo classes are those that are used for hyperlinks (e.g., active, focus, hover, link, visited). For details on the many kinds of pseudo classes that CSS lets you create, see:

[http://www.w3schools.com/css/css\\_pseudo\\_classes.asp](http://www.w3schools.com/css/css_pseudo_classes.asp)

In the New Selector dialog you can use the **Pseudo Class** field to enter or select a pseudo class for the HTML element. In the **Advanced Selector** field, the main HTML element is followed by a colon and then the pseudo class.



☆ **EXAMPLE** You want a text hyperlink to display in green when a user hovers over it. Therefore, in your stylesheet you expand the **a** style and modify the **hover** pseudo class, changing the font color to green.



📄 **NOTE** In order for `<a>` link pseudo classes to function properly, they must appear in the following order in the stylesheet (you can see this by opening the stylesheet in the Internal Text Editor).

a:link  
a:visited  
a:hover  
a:focus  
a:active

In order to avoid issues with this, and to ensure that your pseudo classes are working, you should explicitly set values on those pseudo classes, rather than expecting them to inherit settings from other tags.

# Pseudo Class Expressions

For a handful of pseudo classes, you can also add an expression. If you select one of the valid pseudo classes (e.g., `nth-child`, `not`), you can then enter something in the **Pseudo Class Expression** field (e.g., `3`, `5n+5`, `odd`, `even`).

☆ **EXAMPLE** You want to show the third item in every bulleted list in a blue font.

To accomplish this, you select the `ul` (unordered list) style in the Stylesheet Editor and click the **New Selector** button. Then you click the **Advanced Options** arrow to show the fields at the bottom of the dialog. The Advanced Selector field starts out showing only your main HTML element (`ul`).

The screenshot shows a dialog box titled "New Selector" with a close button (X) and a help button (?). The main instruction is "Manually add selector or choose Options." There are two input fields: "HTML Element" (a dropdown menu showing "ul") and "Class Name" (an empty text box). Below these is an "Advanced Options" section with an expandable arrow. The "Advanced Selector" field is highlighted with a red box and contains "ul". Below this are four more fields: "Pseudo Class" (a dropdown menu), "Pseudo Class Expression" (a text box), "Pseudo Element" (a dropdown menu), and "Identifier (ID)" (a text box). At the bottom is a "Comments" text area and "OK" and "Cancel" buttons.

- ☆ From the **Pseudo Class** field you select **nth-child**. This adds that pseudo class to the Advanced Selector field, after a colon.

The screenshot shows a dialog box titled "New Selector" with a close button (X) and a help button (?). The main instruction is "Manually add selector or choose Options." Below this, there are two input fields: "HTML Element" with a dropdown menu showing "ul" and "Class Name" with an empty text box. A red box highlights the "HTML Element" dropdown. Below these is a section titled "Advanced Options" with a collapse icon. Inside this section, there is an "Advanced Selector" text box containing "ul:nth-child". A red box highlights this text box. Below the "Advanced Selector" are four more fields: "Pseudo Class" with a dropdown menu showing "nth-child", "Pseudo Class Expression" with an empty text box, "Pseudo Element" with a dropdown menu, and "Identifier (ID)" with an empty text box. At the bottom of the dialog is a "Comments" text box and two buttons: "OK" and "Cancel".

- ☆ In the **Pseudo Class Expression**, you type 3 (because you only want the third list item to be affected). This adds the number between parentheses in the Advanced Selector field.

The image shows a 'New Selector' dialog box with the following fields and values:

- HTML Element:** ul
- Class Name:** (empty)
- Advanced Options:**
  - Advanced Selector:** ul:nth-child(3)
  - Pseudo Class:** nth-child
  - Pseudo Class Expression:** 3
  - Pseudo Element:** (empty)
  - Identifier (ID):** (empty)
  - Comments:** (empty)

Buttons: OK, Cancel

- ☆ You're almost done, but there is one more thing to do. In the Advanced Selector field, you need to place your cursor between the "ul" text and the colon, and you need to type a space followed by `li`. This tells Flare that it's not just the unordered list (ul) in general that this applies to, but rather to a specific item (li) within that list.

**New Selector** ? X

Manually add selector or choose Options.

HTML Element:  Class Name:

Advanced Options

Advanced Selector:

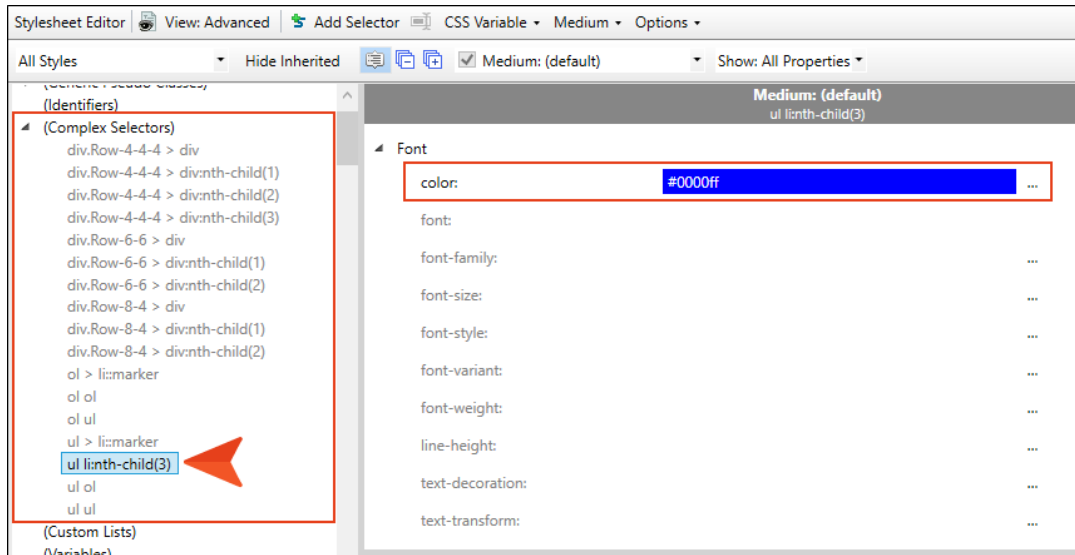
Pseudo Class:  Pseudo Class Expression:

Pseudo Element:  Identifier (ID):

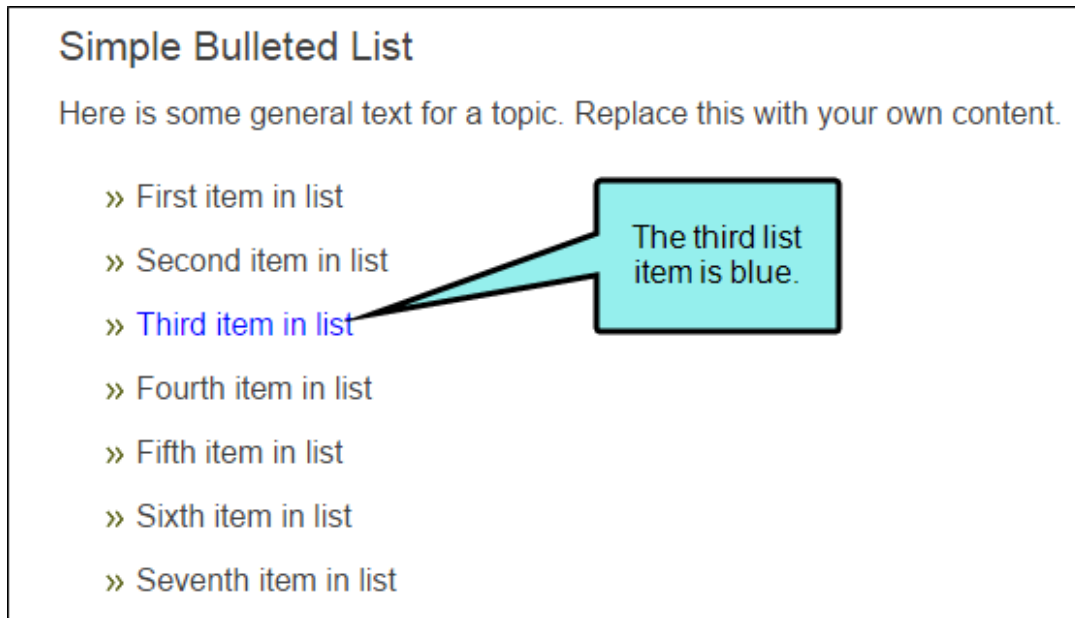
Comments:

OK Cancel

- ☆ After you click **OK**, the new advanced selector is added to your stylesheet. In the Advanced view of the Stylesheet Editor, it will be shown under (Complex Selectors). With this advanced selector highlighted, you change the font color to blue.



And this is what you get as a result in the output:



# Pseudo Elements

In addition to pseudo classes, you can add pseudo elements to a style. Whereas a pseudo class focuses on the state of an element (e.g., change font color when hovered), a pseudo element focuses on a specific part of an element.

In the New Selector dialog you can use the **Pseudo Element** field to make a selection. In the **Advanced Selector** field, the two colons are added, followed by the pseudo element.

The image shows a 'New Selector' dialog box with the following fields and values:

- HTML Element:** p
- Class Name:** (empty)
- Advanced Options:**
  - Advanced Selector:** p::after
  - Pseudo Class:** (empty)
  - Pseudo Class Expression:** (empty)
  - Pseudo Element:** after
  - Identifier (ID):** (empty)
  - Comments:** (empty)

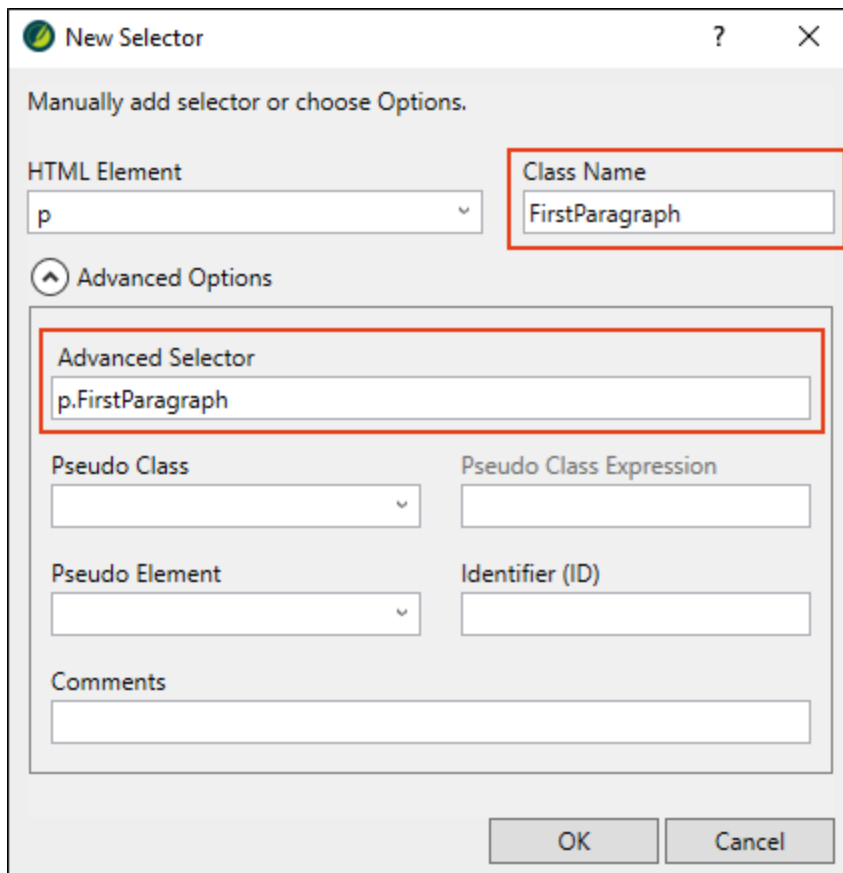
Buttons: OK, Cancel



☆ **EXAMPLE** You want the first paragraph in some of your topics to start out with a letter that is larger and bolder than the rest of the characters.

To accomplish this, you select the **p** (paragraph) style in the Stylesheet Editor and click the **New Selector** button. In the **Class Name** field, you enter some text, such as `FirstParagraph`. Now you've got a style class that lets you make some paragraphs different from the rest.

Then you click the **Advanced Options** arrow to show the fields at the bottom of the dialog. The Advanced Selector field starts out showing only your main HTML element, followed by a period and the name of your class (`p.FirstParagraph`).

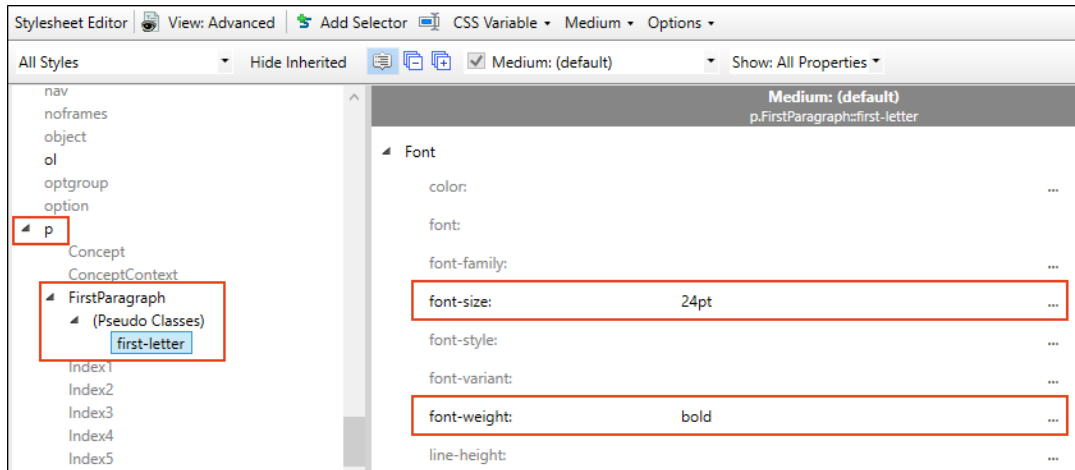


- ☆ From the **Pseudo Element** field you select **first-letter**. This adds that pseudo class to the Advanced Selector field, after two colons.

The screenshot shows a dialog box titled "New Selector" with a close button (X) and a help button (?). The dialog contains the following fields and options:

- Manually add selector or choose Options.**
- HTML Element:** A dropdown menu with "p" selected.
- Class Name:** A text input field containing "FirstParagraph".
- Advanced Options:** A section with a collapse icon (upward arrow) and the following fields:
  - Advanced Selector:** A text input field containing "p.FirstParagraph::first-letter", highlighted with a red border.
  - Pseudo Class:** A dropdown menu.
  - Pseudo Class Expression:** A text input field.
  - Pseudo Element:** A dropdown menu with "first-letter" selected, highlighted with a red border.
  - Identifier (ID):** A text input field.
  - Comments:** A text input field.
- Buttons:** "OK" and "Cancel" buttons at the bottom right.

☆ After you click **OK**, the new advanced selector is added to your stylesheet. With this the first-letter pseudo element highlighted, you change the font size to **24 pt** and the weight to **bold**.



Finally, for any paragraph that you want to use this advanced selector, you apply it in the XML Editor.

And this is what you get as a result in the output:

## Pseudo Element Example

**H**ere is the first paragraph of this topic. We've applied a style class to it so that the first letter is larger than the others.

This is the second paragraph of this topic. Here is some more text.

This is the third paragraph of this topic. Here is some more text.

# Inheritance

One of the features of cascading stylesheets (CSS) that makes it much more powerful than other style systems is inheritance. This is the idea that elements in your document can inherit the style settings from other elements.

## Inheritance for Nested Tags

This kind of inheritance occurs when one style element is added within another, therefore creating nested tags. The content within the inside tag inherits style settings from the outside tag, unless you override those settings on the inside tag. This can be a very powerful feature because it lets you set properties on an outside tag element once rather than setting the same thing on all of the tags within it.

☆ **EXAMPLE** You want all of your block-level elements to use Arial as the font type. Rather than setting Arial on all of the various styles (h1, p, div, ul, li), you can set it on the body style. That way, the setting will "trickle down" automatically to all of the tags within it.

The diagram shows a snippet of HTML code with three callout boxes explaining inheritance. The code is as follows:

```
<head>
  <link href="../Resources/Stylesheets/
</head>
<body>
  <h1>Basic Steps</h1>
  <p>Here are the basic steps for
  <ol>
    <li>Here is some general
    <li>Here is so
    <li>Here is some general
  </ol>
</body>
</html>
```

Callout 1 (top left): Notice that the <body> tag is on the outer level of the topic code.

Callout 2 (bottom left): In this example, several other tags (e.g., <h1>, <p>, <ol>) are contained within the <body> tag. Therefore, they inherit settings from it.

Callout 3 (right): With the body style using Arial, all of the other tags within it will also use Arial, unless you explicitly state that any of those styles should use something different.

After some time you might decide that you want to continue using Arial for all of those elements except the p styles. In that case, you can simply set the other font type on the p style. The rest of the elements will continue to use Arial.

# Inheritance from Outside Your Project

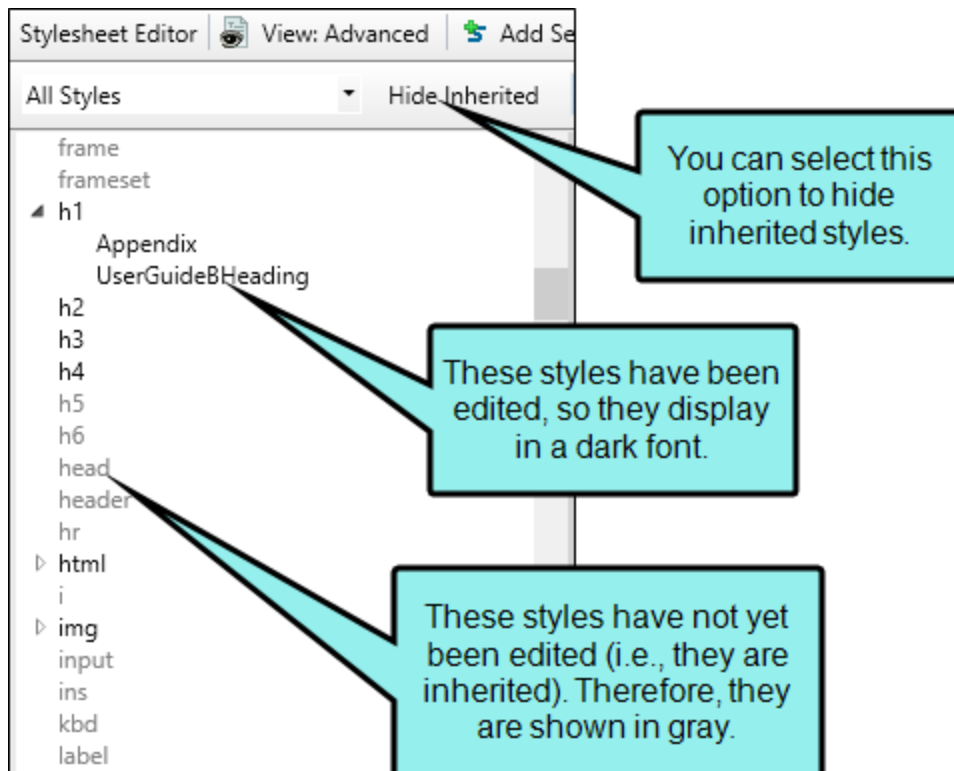
If you do not set values for certain styles in your stylesheet, those values are inherited from elsewhere.

First, there are several application stylesheets within the folder where you installed Flare. The stylesheets that you add to your projects inherit the style definitions that are written in those external application stylesheets. But anything you set in your project stylesheet takes precedence over the same styles that might be found in an application stylesheet.

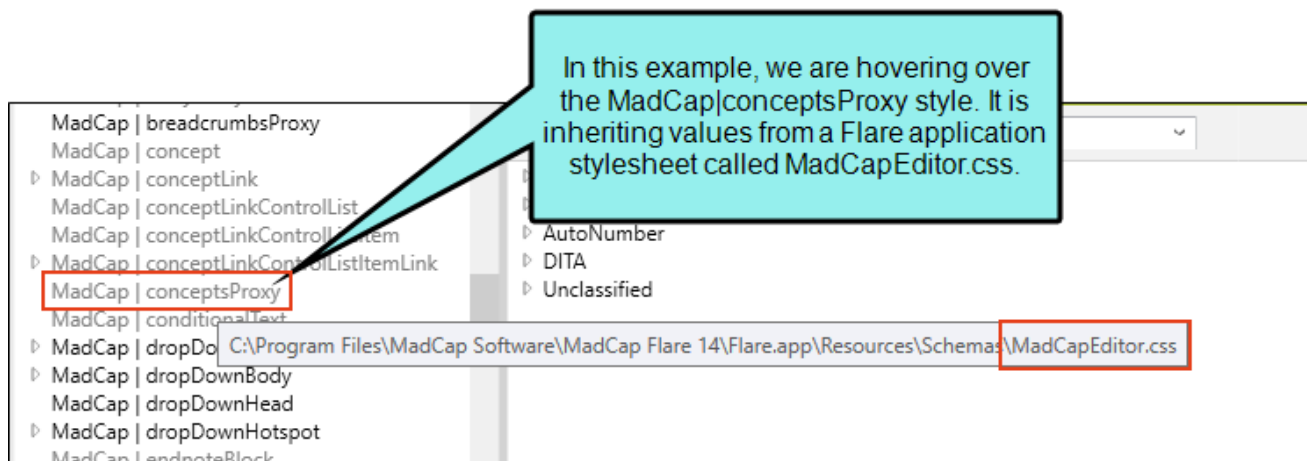
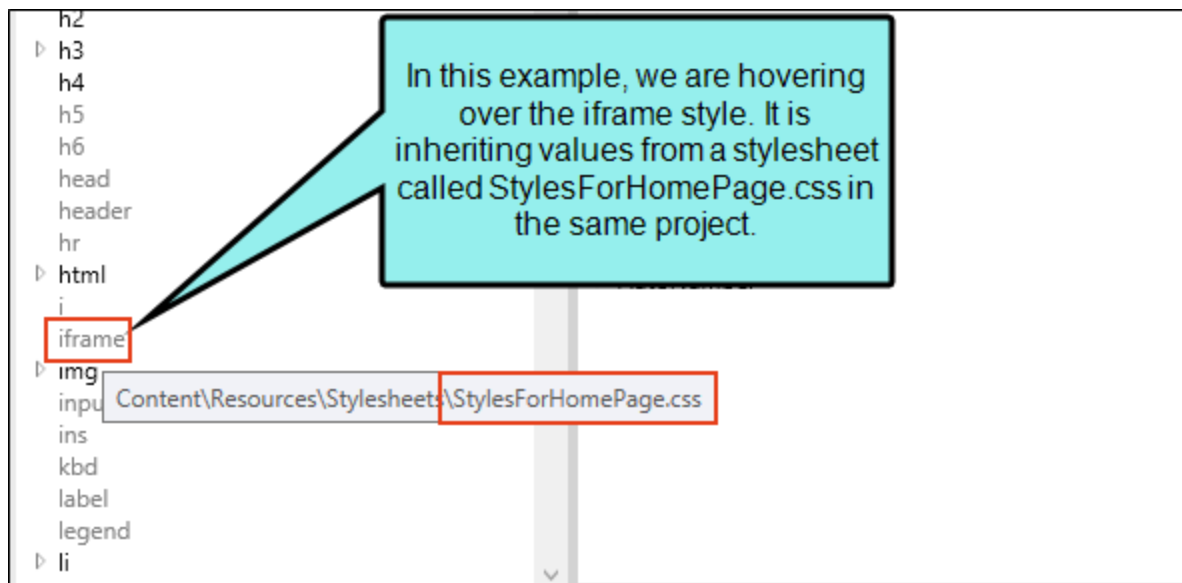
Taking it one step beyond that, if values for a particular style are not explicitly set in either your project stylesheet or in one of Flare's application stylesheets, the default values from the browser are used.

# Inherited Styles Identified in the Stylesheet Editor

When making changes to styles in the Advanced view of the Stylesheet Editor, you may notice that some styles are gray. These are called "inherited" styles. That's because they do not yet have explicit settings on them, so they are inheriting default values from somewhere else (e.g., a factory stylesheet located where you installed the application). As soon as you make a change to one of these styles, it ceases to be an inherited style (or at least the property you set is no longer inheriting from the default value), and the style name turns from gray to a darker font. You can click **Hide Inherited** in the local toolbar if you want to hide these inherited styles.

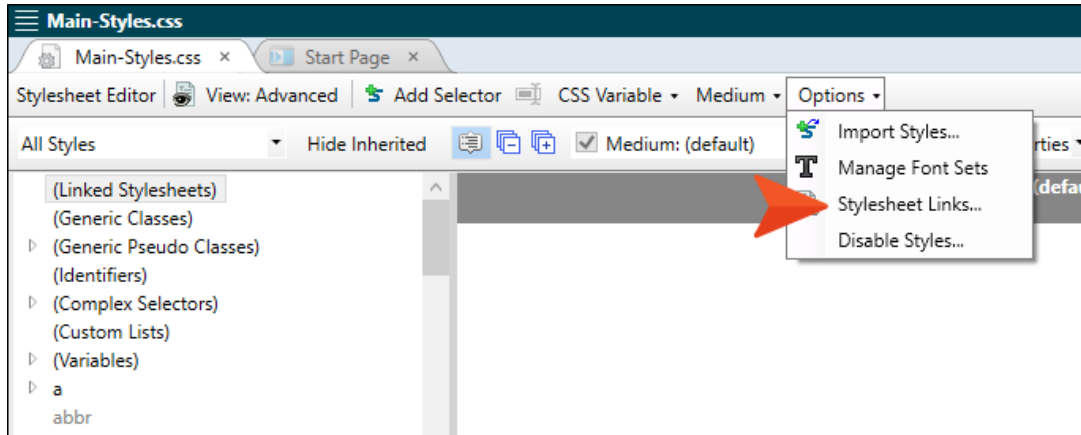


Also, if you hover over an inherited style, Flare displays the path to the stylesheet from which the style is inherited.

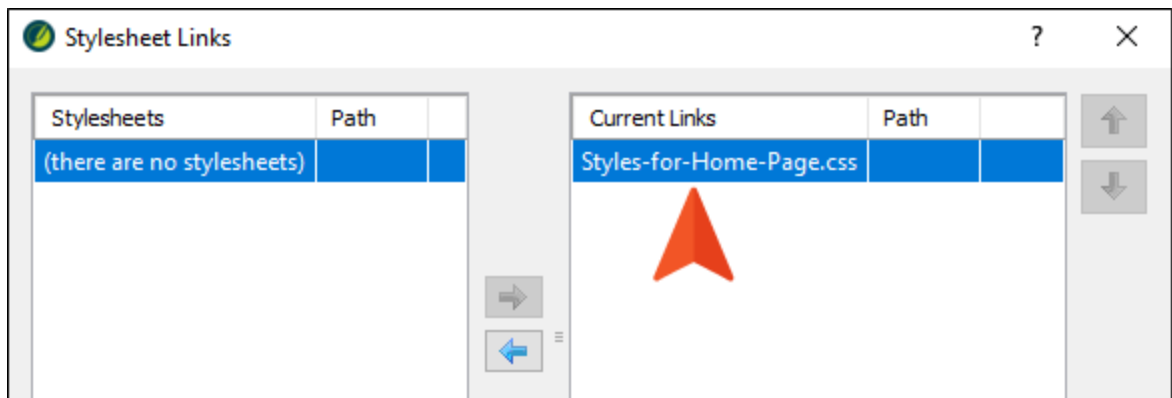


If you have created a link from one stylesheet to another, you can double-click on an inherited property to open that other stylesheet. See "Editing Styles in a Regular Stylesheet" on page 111.

☆ **EXAMPLE** You have two stylesheets in your project—one called "Main-Styles" and the other "Styles-for-Home-Page." You decide to create a link between the two stylesheets, so you open **Main-Styles.css**, select the **Options** drop-down in the Stylesheet Editor, and choose **Stylesheet Links**.

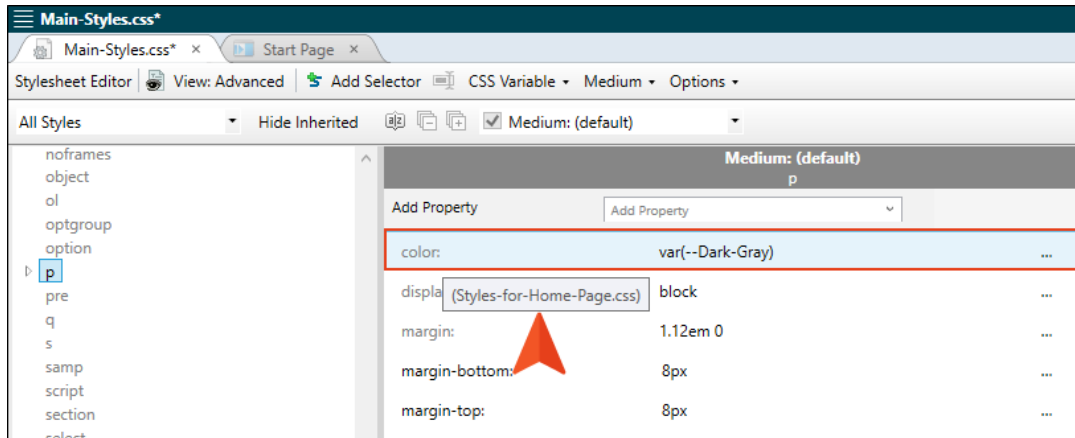


In the Stylesheet Links dialog, you move **Styles-for-Home-Page.css** from the left to the right side.

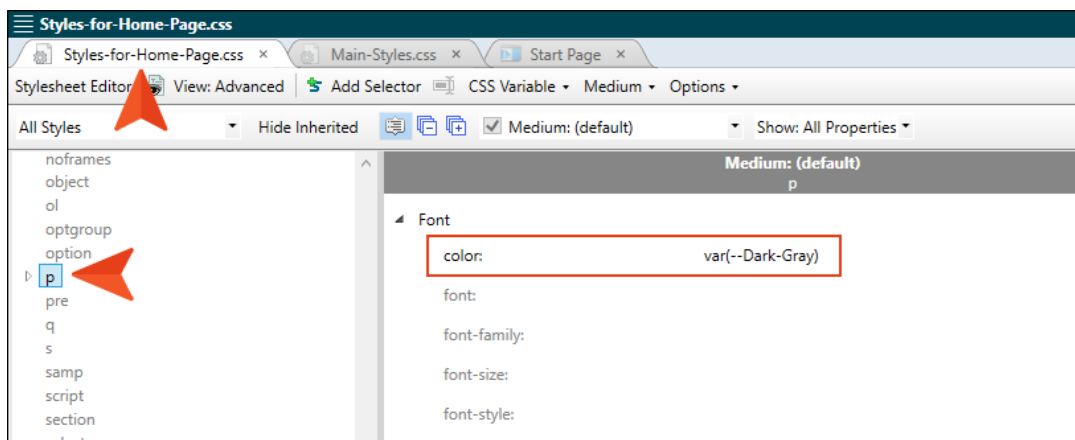




- ☆ Back in the Main-Styles.css file, you notice that the color property is inheriting a setting. This is indicated by the gray text. If you hover over this property, you can see that the color is coming from the Styles-for-Home-Page.css stylesheet.



If you open the Styles-for-Home-Page.css stylesheet and select that same style, you will notice that the text for the color property is darker, meaning it is the source of that setting.



## CHAPTER 3

---

# General Information for Styles

There are various pieces of general information you should know if you plan to use this feature.

This chapter discusses the following:

Stylesheet and Formatting Options .....	71
Types of Styles in Flare .....	74
Primary and Local Stylesheets (and Precedence) .....	86
MadCap-Specific Styles and Properties .....	99
Where's My Style? .....	101

# I Stylesheet and Formatting Options

You can control the look of tables in the following ways:

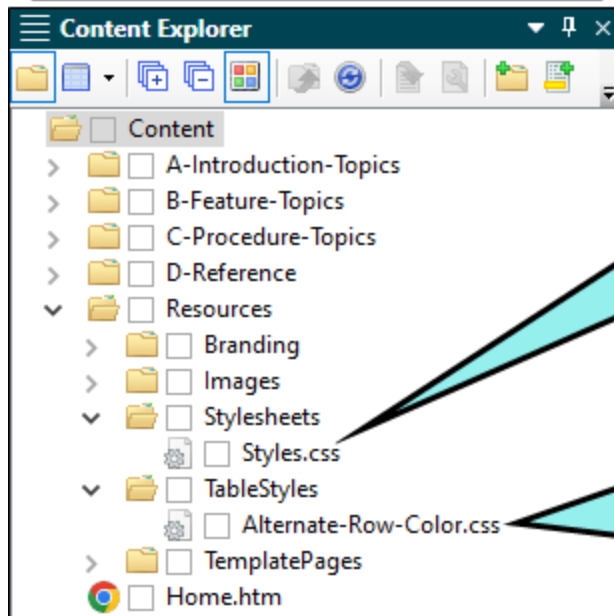
- **Branding Stylesheet** This lets you define your project's look and feel for branding purposes, including tables. If you create your project with Start New Project Wizard, the CSS variables are automatically linked to various places throughout the project where they point to the branding stylesheet.
- **Regular Stylesheet** This lets you store styles for general content in your project, including tables, to control how that content looks. See "Regular Stylesheets" on page 157 and "Editing Styles in a Regular Stylesheet" on page 111.
- **Table Stylesheet** This lets you control the look for tables only, creating customizable patterns. See "Table Stylesheets" on page 270 and "Editing Table Stylesheets" on page 279.
- **Table Properties Dialog** This lets you control the look for a specific table only, by setting options locally. For more information see the online Help or the Flare *Tables Guide*.
- **Other Local Formatting Tools** For virtually any kind of content that you add to a content file (e.g., topic, snippet), there are a variety of local formatting tools to control the look. For example, you can highlight text in a table and use the Home ribbon to change the color. For more information see the Flare online Help.

## Which Should I Use?

Using a stylesheet is always preferred over local formatting (i.e., controlling the look of content only at the place where it has been inserted). The good thing about stylesheets is that they let you separate the presentation from the content. That way, you can manage the look from one place, which can save you a lot of time and effort. For example, you can change the padding in 84 tables by changing a setting in a single stylesheet, as opposed to opening each of those 84 tables and changing the padding in each one. Of course, there may be times when you need to make a change in the properties for a single table only, but for the most part, you should try to use stylesheets.

So why are there two kinds of stylesheets for tables? Why do you need to use a *table* stylesheet when you've already got a *regular* stylesheet? You actually do not need to. You can produce a look for tables entirely by using a regular stylesheet. But if you want to create many different table designs, and tables that have different patterns (e.g., every other row might have a green background), it can be very difficult to do this in a regular stylesheet. So Flare lets you use these special table stylesheets to accomplish this task much more easily. You might even use both types of stylesheets to control different aspects of your tables.

The recommended location to store regular stylesheets is the Stylesheets folder, and the recommended location to store table stylesheets is the TableStyles folder. But you can place them anywhere else you'd like in the Content Explorer.



In this example, a regular stylesheet is stored in the Stylesheets folder.

In this example, the table stylesheet is stored in the TableStyles folder.

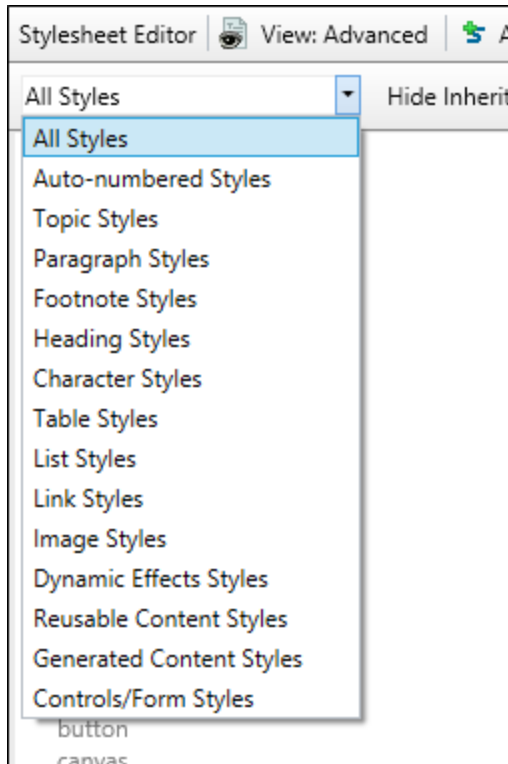
# What About Conflicts and Precedence?

Because you can often control the look of a table in multiple ways—(1) local formatting tools, (2) local table properties, (3) a table stylesheet, (4) a regular stylesheet, or (5) branding stylesheet—it's possible that you might encounter conflicting settings from time to time. When this happens, the settings closest to the content typically has precedence. So precedence works like this: Local Formatting > Table Stylesheet > Regular Stylesheet > Branding Stylesheet.

☆ **EXAMPLE** You open a regular stylesheet and specify that the outer borders of the table should be green. Then you open the table stylesheet and specify that the outer borders should be red. And then you open the Table Properties dialog and specify that the outer borders should be blue. You've told Flare to do three different things to the same table. So in this case, the table would display blue borders, because the local properties rule over the other settings. But if you remove that setting from the Table Properties dialog and use the default setting, the table would then display red borders, because the table stylesheet has precedence over the regular stylesheet. And finally, if you remove the settings from both the Table Properties dialog and table stylesheet, using the default setting in both, the table would take its command from the regular stylesheet and display green borders.

# I Types of Styles in Flare

There are several categories of styles that you can apply to content. To filter the list of styles shown, use the drop-down list in the upper-left of the Stylesheet Editor.



For steps on applying styles, see "Applying Styles to Content" on page 136. Following are explanations of the basic types of styles. For more information about the MadCap-specific styles you might see when using this filter, see "MadCap-Specific Styles and Properties" on page 99.

## All Styles

This lists all of the styles in the Stylesheet Editor.

## Auto-numbered Styles

These are styles to which an autonumber format has been applied.

# Topic Styles

This lists the html style and its classes, which are used to affect entire topics.

## Paragraph Styles

### Main Paragraph Styles

- `p` Regular paragraphs
- `div` Containers that can be used for various purposes
- `h1` - `h6` Headings

### MadCap-Specific Paragraph Styles

- `MadCap|relationshipsHeading` Modifies the look of headings used in relationship links. There are three classes of this style that you can edit. If you edit the main `MadCap|relationshipsHeading` style, the look of all of the classes are affected. However, you can also edit the look of each class if you want.
- `MadCap|slideshowBullet` Modifies the look of the series of dots (or "bullets") used to navigate to specific slides in a slideshow. Keep in mind that if you choose to include thumbnail images, the `MadCap|slideThumbnail` style will be used instead.
- `MadCap|slideshowCaption` Modifies the look of the caption at the bottom of the slide in a slideshow.

### Additional Paragraph Styles

The other paragraph styles that you might see in the Stylesheet Editor are used for standard html tags. For details on each of these, see [w3.org](http://w3.org).

# Footnote Styles

These are applied to footnotes inserted into content. Footnotes are commonly used if you are producing print-based output. `MadCap|footnotesBlock` affects the area that holds a collection of footnotes.

- **MadCap|endnoteBlock** Modifies the container (or block) holding individual endnote comments. For example, use this if you want to add a border around each endnote comment created from an Endnotes proxy.
- **MadCap|endnotesBlock** Modifies the container (or block) holding all endnote comments. For example, use this if you want to add a border around the collection of all endnote comments created from an Endnotes proxy.
- **MadCap|endnotesProxy** Modifies the appearance of the text portion of the Endnotes proxy.
- **MadCap|footnote** Modifies both the footnote number (or symbol) where it is inserted in the topic, as well as the number and accompanying comment text (at the bottom of the page, or wherever else you specify its location).
- **MadCap|footnoteBlock** Modifies the container (or block) holding individual footnote comments. For example, use this if you want to add a border around each footnote comment on a page.
- **MadCap|footnotesBlock** Modifies the container (or block) holding all footnote comments. For example, use this if you want to add a border around the collection of all footnote comments on a page.

# Heading Styles

These are styles that are applied to content intended to serve as headings above sections of content.

# Character Styles

These are styles that are applied to selected text within a paragraph, rather than the entire paragraph.



# Main Span Style(s)

You are likely to create classes under the generic span style in order to create your own custom character formatting.

## MadCap-Specific Character Styles

- **MadCap|annotation** Modifies the look of content to which an annotation (i.e., internal topic comment) points. For example, you might want annotated text to be displayed in the XML Editor with red font and a yellow background. This does not change the text as it will be shown in the output, but rather only as it is displayed in the XML Editor for authors. When an annotation is inserted in a content file, the MadCap:annotation tag includes the comment's creation date, user name and initials (as set in the File > Options dialog, Review tab) of the person who created or edited it, and the comment text.
- **MadCap|codeSnippet** Modifies the look of the entire code snippet block that has been inserted in the XML Editor.
- **MadCap|codeSnippetBody** Modifies the look of the code snippet text, as well as the line numbers and vertical border to the right of the numbers.
- **MadCap|codeSnippetCaption** Modifies the look of the caption used for the code snippet.
- **MadCap|codeSnippetCopyButton** Modifies the look of the copy button link that can be added to code snippets for HTML5 output. If you want to change the word "Copy" to something else, you can edit the mc-label property.
- **MadCap|concept** Modifies the look of concepts that have been inserted in the XML Editor (when markers are turned on). This does not affect the output.
- **MadCap|conditionalText** Modifies the look of content in the XML Editor that has a condition tag applied to it. For example, you might want conditioned content to stand out with a larger font so you can easily spot it while editing content. This does not affect the output.
- **MadCap|correctFeedback** Modifies the appearance of content that is shown as feedback when the eLearning question is answered correctly.
- **MadCap|equation** Modifies the appearance of all equations.
- **MadCap|incorrectFeedback** Modifies the appearance of content that is shown as feedback when the eLearning question is answered incorrectly.

- **MadCap|keyword** Modifies the look of index keywords that have been inserted in the XML Editor (when markers are turned on). This does not affect the output.
- **MadCap|multipleChoice** Modifies the look of question sections that have been inserted in the XML Editor. This consists of the MadCap|question, MadCap|multipleChoiceItem, MadCap|correctFeedback, MadCap|incorrectFeedback, and MadCap|submitQuestionButton sections.
- **MadCap|multipleChoiceItem** Modifies the look of the answer in the XML Editor that has been inserted within the MadCap|multipleChoice section.
- **MadCap|namedDestination** This style does not have any relevant style properties. Named destinations are used in PDF output to label certain locations in the document. These locations can then be linked to directly from another PDF document.
- **MadCap|qrCode** Modifies the appearance of all QR codes.
- **MadCap|relationshipsItem** Modifies the look of link items created from a relationships table.
- **MadCap|section** This style displays in the interface due to Flare's schema. However, it doesn't have a function, so you can ignore it.
- **MadCap|slideThumbnail** Modifies the look of the thumbnail image area at the bottom of the slide.
- **MadCap|submitQuestionButton** Modifies the look of the submit button that is shown to end the test for HTML5 output.

## Additional Character Styles


The other character styles that you might see in the Stylesheet Editor are used for standard html tags. For details on each of these, see [w3.org](http://w3.org).

## Table Styles

These are styles that are applied to tables and the content within them.

- **caption** Modifies the table caption, which is a short title or description of the table's purpose. When inserting or editing a table, you can add a caption above or below the table.

- **col** Groups together attribute specifications for table columns. The `<col>` elements are empty and serve only as a support for attributes. They may appear inside or outside an explicit column group (i.e., `<colgroup>` element).
- **colgroup** Groups columns together structurally. The number of columns in the column group may be specified by using the element's `<span>` tag or by the `<col>` element, which represents one or more columns in the group.
- **table** Modifies an entire table. It contains all other elements that specify caption, rows, content, and formatting.
- **tbody** Modifies the main rows in a table (i.e., not the header or footer rows). Each `<tbody>` tag must have at least one `<tr>` tag within it, which is used to represent a single row.
- **td** Modifies the data (or content) in the primary cells of a table. When you press ENTER after the first paragraph in a table cell, a paragraph `<p>` tag is added inside each `<td>` tag in that cell.
- **tfoot** Modifies a footer row in a table. When a table requires multiple pages in print layouts and outputs, the footer row is placed at the bottom of the last page. Each `<tfoot>` tag must have at least one `<tr>` tag within it, which is used to represent a single row.
- **th** Modifies the header content in a table. Why not just use the `<td>` tag for header content as well? First, by having different tags, you can more easily dictate one look for the header text (e.g., bold font) and a different look for the main content in the table (e.g., normal font). Second, using separate tags greatly assists users with visual disabilities, making it possible for multi-modal wireless browsers with limited display capabilities (e.g., Web-enabled pagers and phones) to handle tables. When you press ENTER after the first paragraph in a table cell, a paragraph `<p>` tag is added inside each `<th>` tag in that cell.
- **thead** Modifies a header row in a table. When a table requires multiple pages in output, the header row is repeated by default at the top of each page. Each `<thead>` tag must have at least one `<tr>` tag within it, which is used to represent a single row.
- **tr** Modifies single rows that are contained within `<tbody>`, `<tfoot>`, and `<thead>` tags.

 **WARNING** When controlling the look of tables, be aware of conflicts that can arise when you are using standard table styles (e.g., `tr`, `td`) from a regular stylesheet and you are also inserting proxies in the project. For example, if you set the `text-indent` property on the `td` style, it could affect the indentation of a generated table of contents or mini-toc.

# List Styles

These are styles applied to bulleted or numbered lists.

- **(Custom Lists)** Modifies styles associated with *custom list formats*.
- **li** Modifies *individual list items*.
- **ol** Modifies an *entire numbered ("ordered") list*, such as a set of steps in a procedure.
- **ul** Modifies an *entire bulleted ("unordered") list*.
- **dl** Modifies a *definition list*.
- **dt** Modifies *terms in a definition list*.
- **dd** Modifies *definitions in a definition list*.

# Link Styles

These are styles that are applied to content that contains a link, such as a hyperlink or cross-reference.

## Main Hyperlink (a) Style

The "a" style is used to modify standard links, such as text hyperlinks.

## MadCap|xref Style

The MadCap|xref style is used to modify the look and format in cross-references. This is the main style used for basic cross-references that you create.

In addition to creating your own custom classes of the main MadCap|xref style, you can also edit the following classes to control the look of page numbers in various places for print-based output.

- **ConceptPageNumber** Modifies the look of page numbers in a generated list of concepts.
- **IndexPageNumber** Modifies the look of the page numbers in a generated index.
- **ListOfPageNumber** Modifies the look of page numbers in a generated list of elements.

- **RelLinkPageNumber** Modifies the look of page numbers in a generated list of relationship links.
- **TOCPageNumber** Modifies the look of page numbers in a generated table of contents.

## Other MadCap-Specific Link Styles

- **MadCap|conceptLink** Modifies the look (e.g., font, color, wording) of a concept (See Also) link heading. When you do this, the style changes for all concept links in any topics in your project.
- **MadCap|conceptLinkControlList** Modifies the look of the entire list (<ul> element) when concept links are displayed in a list, rather than in a popup.
- **MadCap|conceptLinkControlListItem** Modifies the look of individual items in the list (<li> elements) when concept links are displayed in a list, rather than in a popup.
- **MadCap|conceptLinkControlListItemLink** Modifies the look of links in the list (<a> elements) when concept links are displayed in a list, rather than in a popup.
- **MadCap|helpControlList** Modifies the look of the *entire list* (<ul> element) when Help control links are displayed in a list, rather than in a popup. This is a general style that controls all three types of Help control links—concept, keyword, and related topics. Alternatively, you can set properties on each specific style—**MadCap|conceptLinkControlList**, **MadCap|keywordLinkControlList**, or **MadCap|relatedTopicsControlList**.
- **MadCap|helpControlListItem** Modifies the look of *individual items in the list* (<li> elements) when Help control links are displayed in a list, rather than in a popup. This is a general style that controls all three types of Help control links—concept, keyword, and related topics. Alternatively, you can set properties on each specific style—**MadCap|conceptLinkControlListItem**, **MadCap|keywordLinkControlListItem**, or **MadCap|relatedTopicsControlListItem**.
- **MadCap|helpControlListItemLink** Modifies the look of *links in the list* (<a> elements) when Help control links are displayed in a list, rather than in a popup. This is a general style that controls all three types of Help control links—concept, keyword, and related topics. Alternatively, you can set properties on each specific style—**MadCap|conceptLinkControlListItemLink**, **MadCap|keywordLinkControlListItemLink**, or **MadCap|relatedTopicsControlListItemLink**.
- **MadCap|keywordLink** Modifies the look (e.g., font, color, wording) of a keyword link heading. When you do this, the style changes for all keyword links in any topics in your project.

- **MadCap|keywordLinkControlList** Modifies the look of the entire list (<ul> element) when keyword links are displayed in a list, rather than in a popup.
- **MadCap|keywordLinkControlListItem** Modifies the look of individual items in the list (<li> elements) when keyword links are displayed in a list, rather than in a popup.
- **MadCap|keywordLinkControlListItemLink** Modifies the look of links in the list (<a> elements) when keyword links are displayed in a list, rather than in a popup.
- **MadCap|relatedTopics** Modifies the look (e.g., font, color, wording) of a related topics link heading. When you do this, the style changes for all related topics links in any topics in your project.
- **MadCap|relatedTopicsControlList** Modifies the look of the entire list (<ul> element) when related topics are displayed in a list, rather than in a popup.
- **MadCap|relatedTopicsControlListItem** Modifies the look of individual items in the list (<li> elements) when related topics are displayed in a list, rather than in a popup.
- **MadCap|relatedTopicsControlListItemLink** Modifies the look of links in the list (<a> elements) when related topics are displayed in a list, rather than in a popup.
- **MadCap|shortcut** Modifies the look (e.g., font, color) of a shortcut control link. When you edit the style for a shortcut control, the style changes for all shortcut controls in any topics in your project.

## Image Styles

These are styles applied to images and objects that you inserted into content files.

- **img** Main style used to control the look of images (e.g., resize automatically).
- **MadCap|model3D** Modifies the look of 3D models.
- **object** Modifies embedded objects, such as multimedia.

## Dynamic Effects Styles

These are styles that are applied to content used in Dynamic HTML effects (e.g., drop-downs, popups).

- **MadCap|dropDown** Modifies the entire container holding a drop-down effect, including the image that is shown when a drop-down effect is open or closed.
- **MadCap|dropDownBody** Modifies content displayed when users open a drop-down effect.
- **MadCap|dropDownHead** Modifies the text in the first paragraph of a drop-down effect (i.e., the paragraph where the drop-down link is located).
- **MadCap|dropDownHotspot** Modifies the specific text that you select in the first paragraph of a drop-down effect to serve as the link for opening the drop-down body. If you do not select specific text in the first paragraph to serve as the hotspot, the entire first paragraph is used as the hotspot.
- **MadCap|expanding** Modifies the entire container holding an expanding text effect, including the image that is shown when an expanding text effect is open or closed.
- **MadCap|expandingBody** Modifies the expanded text portion of an expanding text effect (i.e., the area that is displayed or hidden when users click the hotspot link).
- **MadCap|expandingHead** Modifies the hotspot portion of an expanding text effect.
- **MadCap|glossaryTerm** Modifies the look of glossary term links.
- **MadCap|helpControlMenu** Modifies the look of links (i.e., menu items) that users see when they click a concept link, keyword link, or related topics control. This style is grouped with the "Dynamic Effects Styles" (which you can select from the drop-down list in the upper-left corner of the Stylesheet Editor). This particular style controls the *entire list* when you are using the *popup menu* method for displaying Help control links.
- **MadCap|helpControlMenuItem** Modifies the look of links (i.e., menu items) that users see when they click a concept link, keyword link, or related topics control. This style is grouped with the "Dynamic Effects Styles" (which you can select from the drop-down list in the upper-left corner of the Stylesheet Editor). This particular style controls the *individual list items* when you are using the *popup menu* method for displaying Help control links.
- **MadCap|microContent** Related to micro content that you create. However, in this version, modifying the style will have no effect on the output.
- **MadCap|popup** Modifies the look of the container holding a text popup link. For example, you can modify this style to place a border around the link.
- **MadCap|popupBody** Modifies the popup text portion of an popup text effect (i.e., the area that is displayed or hidden when users click the hotspot link).
- **MadCap|popupHead** Modifies the hotspot portion of a popup text effect.
- **MadCap|toggler** Modifies the hotspot portion of a toggler.

# Reusable Content Styles

These are styles that are applied to reusable content, such as snippets, variables, or proxies.

- **MadCap|bodyProxy** Modifies the look of the "container" holding topic content. For example, you might edit this style to add a border around all topic content.
- **MadCap|breadcrumbsProxy** Modifies the look of breadcrumbs in online output.
- **MadCap|centralAccountProxy** This style displays in the interface due to Flare's schema. However, it doesn't have a function, so you can ignore it. To control the look of the Central account link added via a proxy, you can use a skin component.
- **MadCap|conceptsProxy** Modifies the look of the "container" holding a generated list of concepts.
- **MadCap|eLearningToolbarProxy** Modifies the look of the container holding the eLearning Toolbar for adding navigation buttons to topics.
- **MadCap|faqProxy** Modifies the FAQ proxy container holding micro content.
- **MadCap|glossaryProxy** Modifies the look of the "container" holding a generated glossary.
- **MadCap|indexProxy** Modifies the look of the "container" holding a generated index for print-based output.
- **MadCap|knowledgeProxy** Modifies the Knowledge proxy container holding micro content.
- **MadCap|listOfProxy** Modifies the look of the "container" holding a generated list of elements.
- **MadCap|menuProxy** Modifies the look of a menu.
- **MadCap|miniTocProxy** Modifies the look of the "container" holding a generated mini-TOC.
- **MadCap|pageFooter** Modifies the look of the content contained in a page footer used in template pages for Microsoft Word output.
- **MadCap|pageHeader** Modifies the look of the content contained in a page header used in template pages for Microsoft Word output.
- **MadCap|promotionProxy** Modifies the Promotion proxy container holding micro content.
- **MadCap|relationshipsProxy** Modifies the look of the "container" holding the content from a generated relationships table.
- **MadCap|searchBarProxy** Modifies the look of search bar.
- **MadCap|searchResultsProxy** Modifies the look of a custom search results page.



- **MadCap|snippetBlock** Modifies the look of block snippets that have been inserted in the XML Editor. If you insert a snippet on a blank line in a topic, it is inserted as a block snippet (as opposed to a text snippet) and takes up all of the room so that no other content can be added.
- **MadCap|snippetText** Modifies the look of text snippets that have been inserted in the XML Editor. If you insert a snippet on a line where other content exists, it is inserted as a text snippet, as opposed to a block snippet.
- **MadCap|testResultsProxy** Modifies the look of the container holding the Test Results when customizing the eLearning test results pages.
- **MadCap|tocProxy** Modifies the look of the "container" holding a generated TOC for print-based output.
- **MadCap|topicToolbarProxy** Modifies the look of the "container" holding a generated topic toolbar. For HTML5 outputs, the Topic Toolbar proxy will use whatever settings are specified in a Topic Toolbar skin component (if you have added one to your project), overriding any buttons you may have selected directly in the proxy. If you have not associated a Topic Toolbar skin component with the proxy, Flare will just use the first one it finds in your project. However, for outputs using Standard and Mobile skins, the settings in the proxy take precedence over anything you may have set on the Toolbar tab in the Skin Editor.
- **MadCap|variable** Modifies the look of variables in the XML Editor and in generated output files.

## Generated Content Styles

These are styles that are applied to content that is created when you generate output, such as glossaries, indexes, or tables of contents (TOCs).

## Controls/Form Styles

These are styles that are applied to content within controls, such as buttons, or forms.

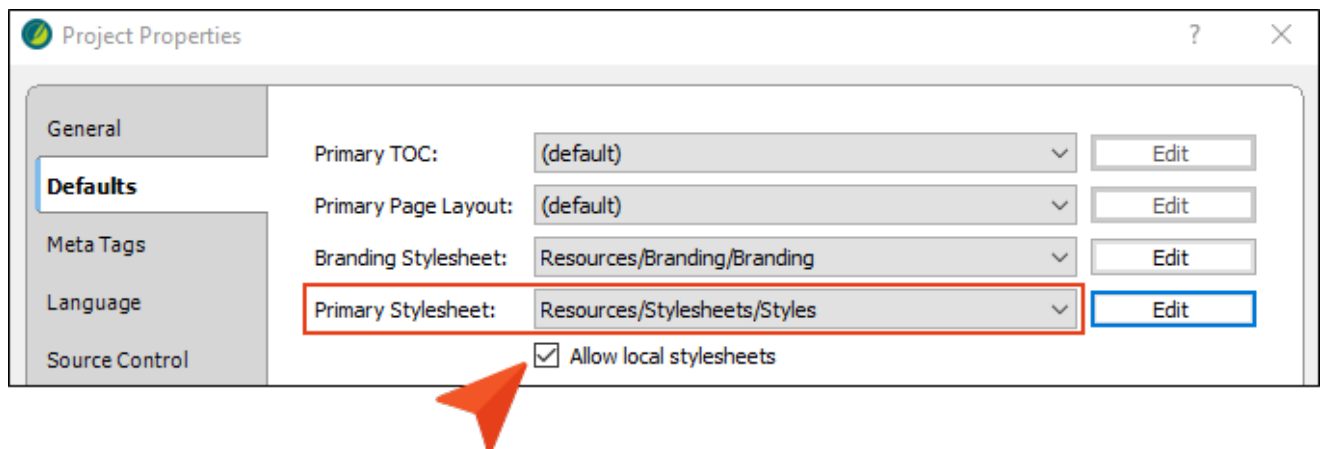
# I Primary and Local Stylesheets (and Precedence)

Flare lets you have multiple stylesheets (primary and local) set on different files and at different levels. Therefore, you need to understand how precedence works, both in the interface (editors) and the output.

## Primary and Local Stylesheets


In a Flare project you are almost certain to use primary stylesheets, which are recommended. But you might find the need to use local stylesheets as well.

- **Primary** You can set stylesheets at the project or target level (in the Project Properties or Target Editor, respectively). These are considered primary stylesheets because they control the look of all files associated with the project or target just from that one setting. See "Associating Primary Stylesheets With All Files" on page 151.
- **Local** In the Project Properties or Target Editor, you can enable the **Allow local stylesheets** option. This lets you set stylesheets at the lower, content-file level (e.g., topics, micro content files). These are considered local stylesheets because they control the look of that single file. See "Associating Stylesheets Locally With Specific Files" on page 153.



# Precedence for Topics in XML Editor

When you are working on content in the XML Editor or Micro Content Editor, the following shows how precedence will work if you have multiple stylesheets.

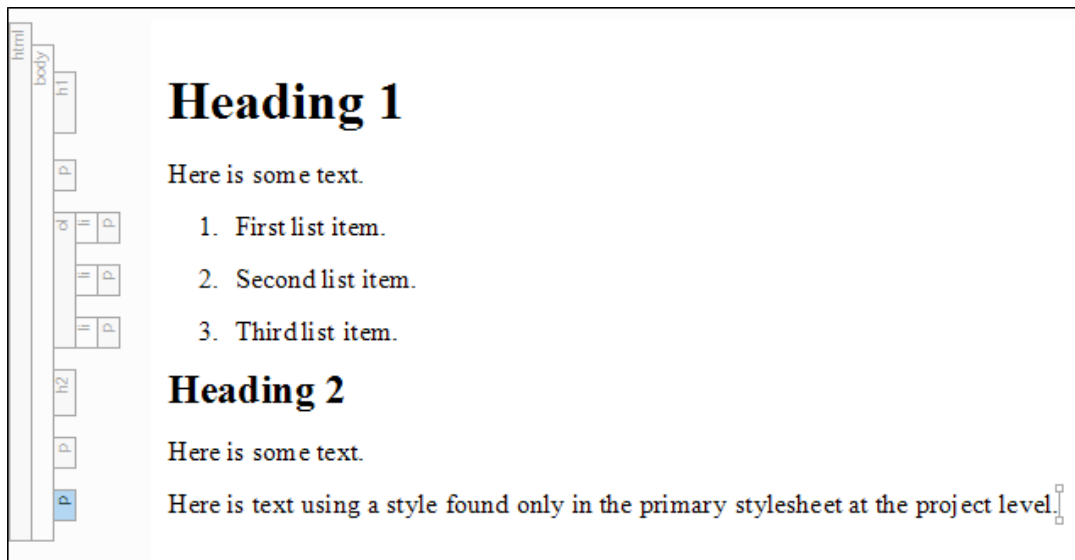
 **NOTE** Keep in mind that a stylesheet associated solely with a micro content file affects only micro content; it does not affect topics.

1. Local stylesheet associated with micro content file
2. Local stylesheet associated with topics
3. Primary stylesheet (styles from only one primary stylesheet can be used)
  - a. *Primary Target*
  - b. Project

## ☆ EXAMPLE – Precedence in XML Editor

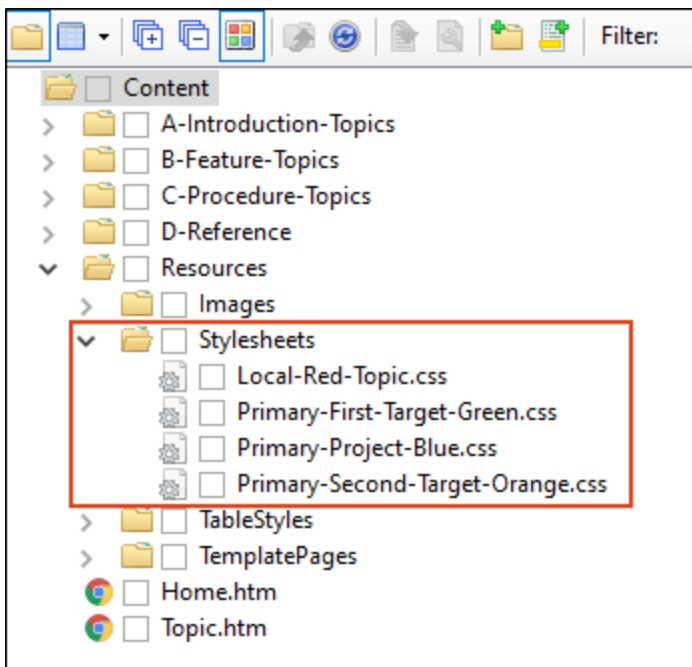
You have a project and want to use multiple stylesheets in it.

In the following image the topic looks very plain, because no stylesheets are yet associated with the project.



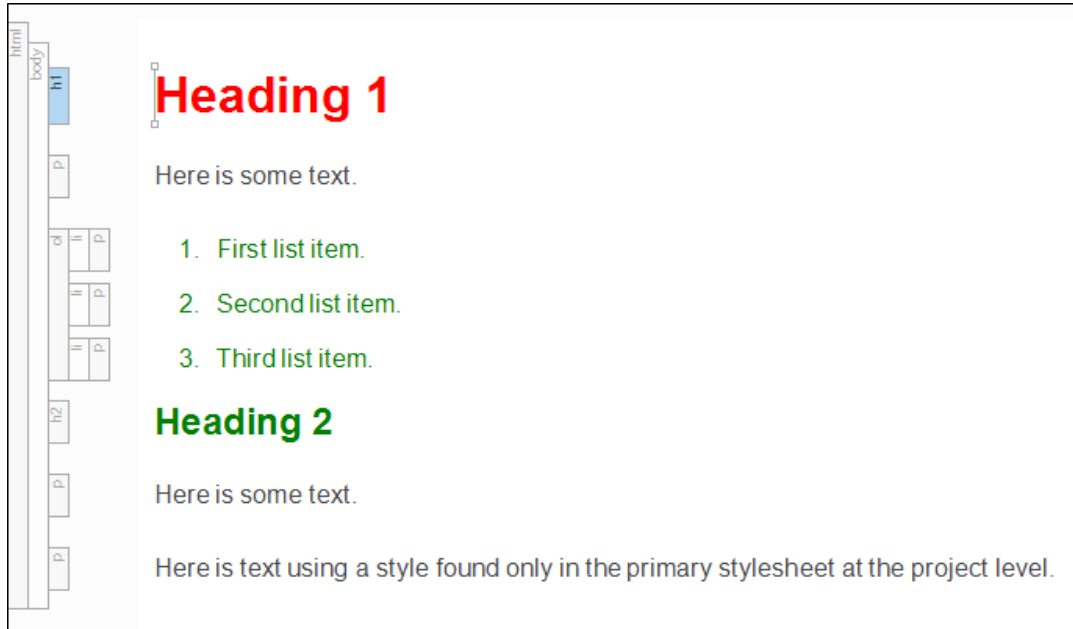
☆ You add the following stylesheets to the Content Explorer and associate them at different levels:

- **Local-Red-Topic** You set this stylesheet locally on a topic. And you tell Flare to use a red font for the h1 style.
- **Primary-First-Target-Green** You associate this stylesheet with your primary target (called "First Target"). And you tell Flare to use a green font for the h1, h2, and ol styles.
- **Primary-Second-Target-Orange** You associate this stylesheet with another target (called "Second Target"). And you tell Flare to use an orange font for the h1, h2, and ol styles.
- **Primary-Project-Blue** You associate this stylesheet with the entire project. And you tell Flare to use a blue font for the h1, h2, and p styles.



For all of the primary stylesheets, you tell Flare to **allow local stylesheets**. In other words, you can use any of the stylesheets in your project.

☆ Now when you open the topic, it looks like this:



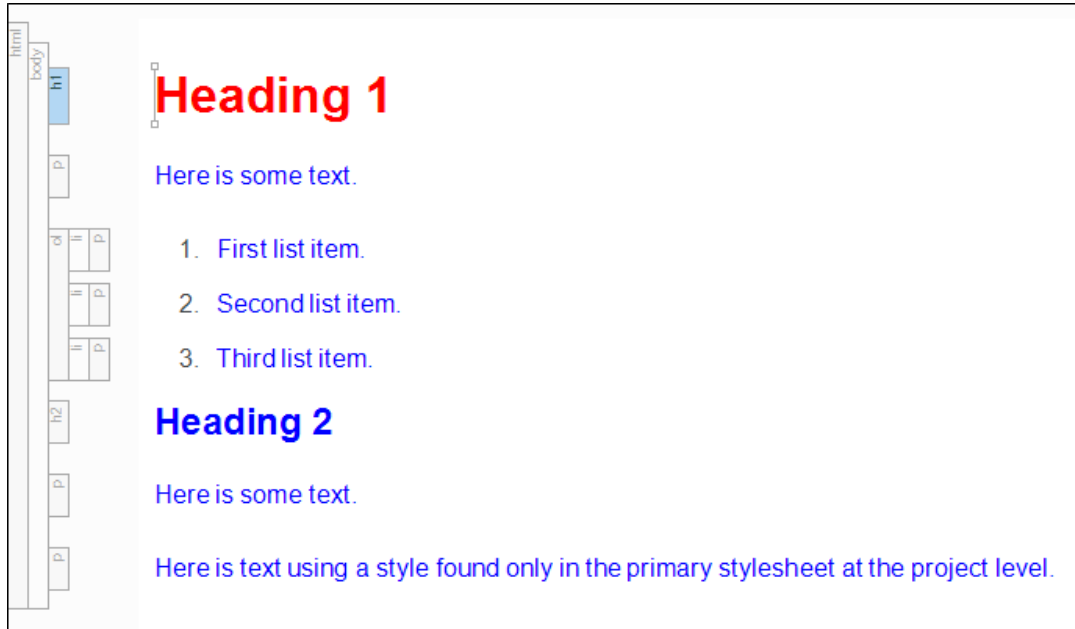
The first-level heading is red because that's what we set in the local stylesheet that is associated with the topic. The other stylesheets have different colors for this style, but the local stylesheet has precedence, so red wins.

The next stylesheet that has precedence is the one for the primary target that has green fonts. That's why you see the second-level heading and the numbered list (which uses the `ol` style) in green. We didn't set anything for the `h2` and `ol` styles in the local stylesheet, so Flare goes to the next one (the primary target stylesheet) and uses its settings.

What about the regular paragraphs? They look better, but they're not displaying in any color. Well, the only stylesheet where we specified a setting for the `p` style is the primary stylesheet at the project level. But because there's another primary stylesheet on the primary target that has precedence, the blue from the project primary stylesheet is completely ignored. Instead, Flare uses the default settings from a factory stylesheet where Flare is installed, and that's why the font family is different from the plain font that you first saw.

- ☆ So what happens if you make a change? Suppose you remove the primary stylesheet that is set on the primary target.


In that case, you would see this in the XML Editor:



The first-level heading is the same because it is coming from the local stylesheet. But now that the primary stylesheet on the primary target is gone, Flare moves to the next primary stylesheet at the project level, which shows the second-level heading and the regular paragraphs in blue. The numbered list is shown in blue only because there are `<p>` tags within the `<ol>` and `<li>` tags that represent the list. If those `<p>` tags were removed, the list would not display a unique color because the `ol` style is not set in either the local stylesheet or the project primary stylesheet. The `ol` style is set in the secondary target, but Flare doesn't use those settings to display content in the XML Editor. It only uses the settings from the primary target. Of course, you can always use the drop-down field on the left side of the local toolbar in the XML Editor to preview the topic as it will be seen in the secondary target, and in that case you will see some orange text.

# Precedence for Output

When you generate output, the following shows how precedence will work if you have multiple stylesheets.

 **NOTE** Keep in mind that a stylesheet associated solely with a micro content file affects only micro content; it does not affect topics.

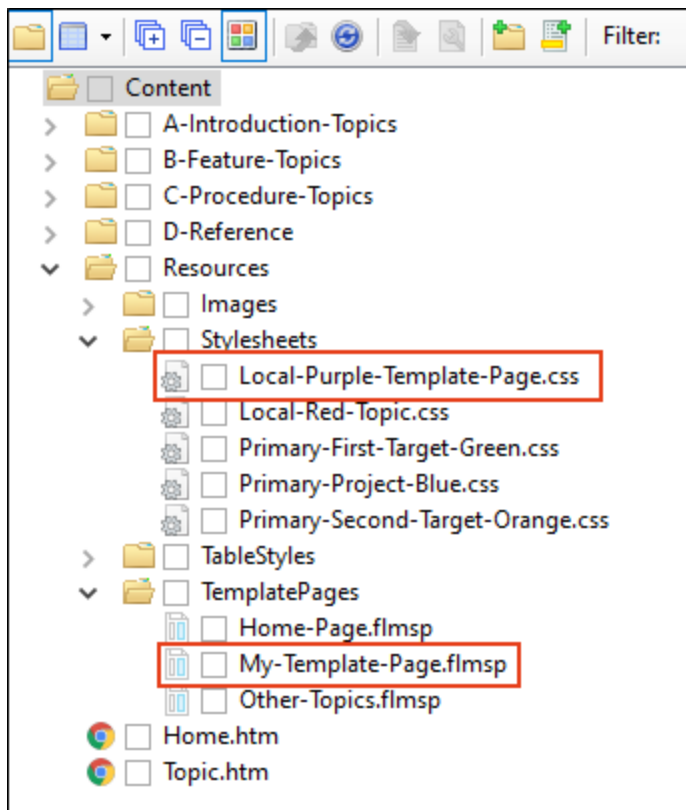
1. Local stylesheet associated with micro content file
2. Local stylesheet associated with topics
3. Local stylesheet associated with template pages
4. Primary stylesheet (styles from only one primary stylesheet can be used)
  - a. *Any Target*
  - b. Project

☆ **EXAMPLE** – Precedence in Output

Take a look at the previous example for precedence in the XML Editor before continuing with this example.

Let's say you add another stylesheet to your project to be used for a template page.

- **Local-Purple-Template-Page** You set this stylesheet locally on a template page. And you tell Flare to use a purple font for the p style.





☆ You've also reset the "Primary-First-Target-Green" stylesheet on your primary target. So you now have all of the stylesheets from the previous example set as they were, and you've added one more local stylesheet to the mix. The template page content does not come into play when you're working in the XML Editor, but what happens in the output? For the primary target, it initially would look like this:

**Heading 1**

Here is some text.

1. First list item.
2. Second list item.
3. Third list item.

**Heading 2**

Here is some text.

Here is text using a style found only in the primary stylesheet at the project level.

Template text

It's similar to what you saw in the previous example when looking at the XML Editor. But now you see that the regular paragraphs (including those within the list) are purple. That's because the local stylesheet for the topic didn't specify a color, and the next stylesheet with precedence is the one used on the template page. You'll also see an extra paragraph at the bottom ("Template text"); this content is coming from the template page, not the topic.



If you generate the second target, it would look like this:

## Heading 1

Here is some text.

1. First list item.
2. Second list item.
3. Third list item.

## Heading 2

Here is some text.

Here is text using a style found only in the primary stylesheet at the project level.

Template text

It's much like the other output, except the green text is now orange, because it's coming from the second target stylesheet, which has precedence over the primary stylesheet at the project level.

But if you remove the primary stylesheet link from either of the targets, the output will look like this:



## Heading 1

Here is some text.

1. First list item.
2. Second list item.
3. Third list item.

## Heading 2

Here is some text.

Here is text using a style found only in the primary stylesheet at the project level.

Template text

☆ Now that the target primary stylesheets are out of the way, Flare looks to the primary stylesheet for the project, which uses blue for the second-level heading. Most of the regular paragraphs remain purple because they are coming from the local template page stylesheet, which has a higher precedence. However, notice the long sentence near the end of the topic. That sentence is blue because it is actually using a class of the main paragraph style (p.SpecialClass), where the blue font is explicitly set. This class is found in the primary stylesheet for the project, but not in the local stylesheet for the template page. That's why it's blue instead of purple.

If you remove the stylesheet link on the template page, the output will look like this:

**Heading 1**

Here is some text.

1. First list item.
2. Second list item.
3. Third list item.

**Heading 2**

Here is some text.

Here is text using a style found only in the primary stylesheet at the project level.

Template text

Now we see more blue.

☆ And finally, if you remove the local stylesheet link on the topic, the output will look like this:

## Heading 1

Here is some text.

1. First list item.
2. Second list item.
3. Third list item.

## Heading 2

Here is some text.

Here is text using a style found only in the primary stylesheet at the project level.

Template text

Now it's all about the project-level primary stylesheet, because all of the others are out of the way. That's why you see blue everywhere.

☆ **EXAMPLE** – Precedence With Micro Content and Other Stylesheets

You have three stylesheets in your project. Two of them are local stylesheets—one at the micro content level and the other at the topic level (for some topics). The third one is a primary stylesheet that is set at the project level.

- In the *micro content stylesheet*, you specify that paragraph text should be **red**.
- In the *topic stylesheet*, you specify that the paragraph text should be **blue**.
- In the *project stylesheet*, you specify that the paragraph text should be **black**.

What is the result in the output?

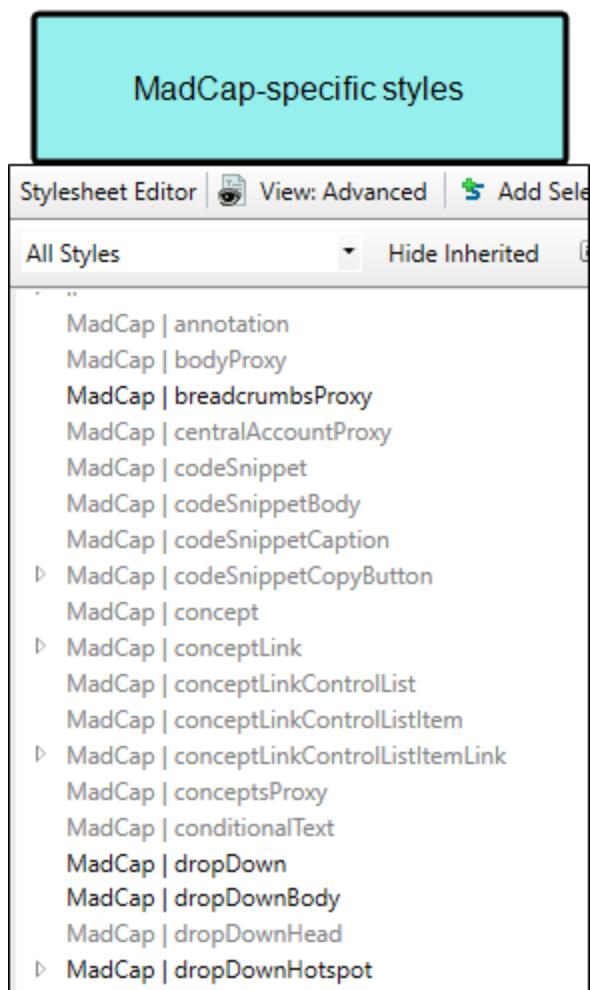
- In the *micro content search results*, the paragraph text will be **red**, overriding the other colors.
- In a *full topic that is associated with the topic stylesheet*, the paragraph text will be **blue**, because it is not being viewed as micro content and it has precedence over the project stylesheet.
- In *full topics that are not associated with the topic stylesheet*, the text will be **black**, because it is not being viewed as micro content.

# MadCap-Specific Styles and Properties

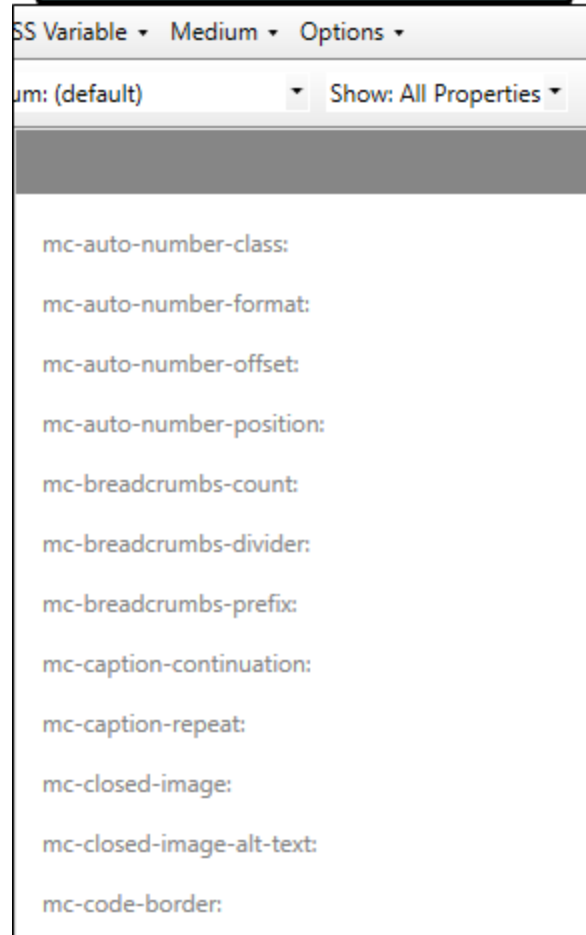
In addition to the many standard styles from W3C, you might notice several unique-looking styles that begin with the word "MadCap" (e.g., MadCap|footnote, MadCap|toggle). There are also many MadCap-specific properties. You will recognize these properties because they always start with "mc" (e.g., mc-footnote-format, mc-hyphenate).

These special styles and properties have been added to the Flare user interface in order to support some of the unique features available only in MadCap Software products.

See "MadCap-Specific Styles" on page 337 and "MadCap-Specific Properties" on page 350.



## MadCap-specific properties






# I Where's My Style?

When applying styles to content, you may notice from time to time that the style you are looking for is not available from the drop-down list or Styles window pane when you try to select it.

This can occur if the style exists in a particular medium (e.g., print) but not in the default medium. So if you are working in the XML Editor with the medium set to default and you attempt to apply that style to content, you won't see it in the selection list with all of the other styles. To correct this, make sure the style exists in the default medium as well.

Another possible reason for this has to do with the location of the cursor in the topic. Flare realizes where the cursor is placed and knows that only certain styles should be applied at that location.

## ☆ EXAMPLE – Lists

Your cursor is on a regular paragraph and you want to apply a list style to it in order to turn it into the beginning of a bulleted list. Because it is not yet a list item, but rather a simple paragraph, you won't see your list style in the Styles window pane when you try to select it. Instead, you see several paragraph styles. In order to use the list style, you first need to turn the paragraph into a bulleted list item, by opening the **Home** ribbon and clicking the bullet button .

You might notice that if you have your cursor in a list, you only see li (list item) styles in the window pane, but not the broader ol (ordered list) and ul (unordered list) styles. To see these other styles, click at the very beginning of a list item. Then press the left arrow key. This should switch the Style window pane from showing li styles to the ol and ul styles. If you have your structure bars on, you'll see why this happens. When you initially click in a list, the li block bar is highlighted, so Flare assumes you want to do something with that style level. After you press your left arrow key enough, the next level up (ol or ul) becomes highlighted. And if you keep pressing the left arrow key, Flare highlights the next level of style (e.g., body). Whatever is highlighted in the structure bar should become available as styles in the Style window pane.

☆ **EXAMPLE** – Paragraph and Character Styles

You've selected multiple paragraphs, or your cursor is simply placed somewhere within a paragraph. In that case, only block-level styles (such as paragraph styles) are shown in the Styles window pane.

But if you select only a portion of a paragraph, only character styles are shown in the Styles window pane. So if you expect to be able to choose a block-level style, such as a paragraph style, you can't; because only a portion of the paragraph is selected, Flare thinks you want to choose a character-level style.

If you still do not see your style available for selection, try closing and re-launching Flare.

## CHAPTER 4

---

# Main Activities for Styles

Some activities are particularly common and important when it comes to this feature.


This chapter discusses the following:


Creating Stylesheets .....	104
Creating Selectors .....	106
Editing Styles in a Regular Stylesheet .....	111
Applying Styles to Content .....	136
Associating Primary Stylesheets With All Files .....	151
Associating Stylesheets Locally With Specific Files .....	153

# Creating Stylesheets


The first step in using styles in your topics is to add a new stylesheet. However, if a stylesheet was included in the template when you created the project, you can use that one instead of creating a new one. The traditional location to store a regular CSS stylesheet in the Content Explorer is in the Resources > Stylesheets folder. However, you can store it anywhere in the Content Explorer that you like.

## How to Create a Regular Stylesheet



1. In the Content Explorer, right-click on a folder and from the context menu select **New > Stylesheet**.
2. In the **File Type** field at the top, make sure **Stylesheet** is selected.
3. In the **Source** area, choose to create the new file based on a template or an existing file.
  - **New From Template** Choose either a factory template file or one of your own custom template files as a starting point. The new file will take on all of the settings contained in the template. If you want to use the factory template provided by Flare, expand the **Factory Templates** folder and click on a template file. If you want to use your own custom template file, expand the appropriate folder and click on a file. For more information about templates, see the online Help.
  - **New From Existing** Choose an existing file of the same type as a starting point for your new file. As with template files, your new file will take on all of the settings contained in the file you select. To use this option, click , use the Open File dialog to find a file, and double-click it.

 **NOTE** Each factory template has different style settings in it. Try different ones to see which suits you best. For example, one of the factory templates is called "Modern." This template includes custom properties for setting the border radius on a paragraph style (i.e., to create rounded corners).

Also, notice a factory template called "SearchHighlight." This template has a specific purpose—for changing how highlighted terms look when performing a search.

- (Optional) The **Folder** field is automatically populated with the folder that has focus in the Content Explorer. If you want to place the file into a folder that you previously created in the Content Explorer, in the **Folder** field click  and select the subfolder. Otherwise, keep the default location.

Non-Topic File Type	Recommended Default Folder in Content Explorer
Branding	Resources > Branding
Image	Resources > Images
Micro Content	Resources > MicroContent
Multimedia	Resources > Multimedia
Page Layout	Resources > PageLayouts
Snippet	Resources > Snippets
Stylesheet	Resources > Stylesheets
Table Stylesheet	Resources > TableStyles
Template Page	Resources > TemplatePages

- In the **File Name** field, type a new name for the stylesheet.
- (Optional) If you want to apply condition tags to the file, expand the **Attributes** section at the bottom of the dialog. Next to the **Condition Tags** field, click  and select the conditions you want to apply. Click **OK**.
- (Optional) If you want to apply file tags, expand the **Attributes** section at the bottom of the dialog. Next to the **File Tags** field, click  and select the file tags you want to apply. Click **OK**.
- Click **Add**. The stylesheet is added to the Content Explorer and opens in its own page in the Stylesheet Editor.


# I Creating Selectors

A selector is a way to associate XHTML content with style settings based on various information—most often its type, class, or ID. Sometimes the word "selector" is used interchangeably with the term "style," but a selector can be much more than just a simple style.

When you create a new stylesheet, it already has many selectors in it. If necessary, you can create new selectors of varying levels of complexity. For more information about the different kinds of selectors, see "Selectors" on page 22.

You can create selectors from the Stylesheet Editor or from a content file (i.e., topic, snippet, template page). If you use the Stylesheet Editor, you can create more kinds of selectors. If you use the content file, the process can sometimes be faster.

## How to Create a Selector From the Stylesheet Editor

1. From the Content Explorer, open the stylesheet that you want to modify.
2. In the local toolbar, click  **Add Selector**. The New Selector dialog opens.
3. Complete any of the fields in the dialog. The two most common fields are the **HTML Element** and **Class Name**, but you can click **Advanced Options** for more fields.

**HTML  
Element**

In CSS there are primary styles that correspond to the different HTML elements (e.g., h1, h2, p, img). You can think of these as parent styles, because in a way, they can have children. A class is the most common type of child for a style. Some classes might already be included in your stylesheet when you first create a project.

In this field, you can select an HTML element (or "parent" style) if you want your new class or other "child" (e.g., ID) to be directly associated with it.

Alternatively, you can clear this field or select (**generic**). This lets you create a generic class or ID that stands alone and can be applied in the content file to any HTML element.

---

**Class Name**

You can create a simple class (e.g., a special paragraph intended to serve as a note or tip). In the New Selector dialog, enter a name for the class.

---

**Advanced  
Selector**

As you complete the different fields in this dialog, the Advanced Selector field is populated accordingly. The reverse is also true. You can enter text in the Advanced Selector field directly, and the other fields in the dialog will be automatically populated. However, you need to enter the correct syntax if you type directly in the field.

---

**Pseudo Class**

If you want to add a pseudo class to your selector, choose it from this field.

In CSS, pseudo classes are a special group of style classes that pertain to elements when they're in a certain state (e.g., the font turns orange when a user hovers over it). They are often (but not exclusively) used for styles associated with hyperlinks.

---

**Pseudo Class  
Expression**

For a handful of pseudo classes, you can also add an expression. If you select one of the valid pseudo classes (e.g., nth-child, not), you can then enter something in the **Pseudo Class Expression** field (e.g., 3, 5n+5, odd, even).

---

**Pseudo Element** In addition to pseudo classes, you can add pseudo elements to a style. Whereas a pseudo class focuses on the state of an element (e.g., change font color when hovered), a pseudo element focuses on a specific part of an element.


---

**Identifier (ID)** In CSS, an identifier (ID) is similar to a class, except that IDs are unique. An element in your stylesheet can have only one ID on it, whereas it can have multiple classes. And each page of your output can have only one element with a particular ID. For many authors, using an ID may not be important, but for others—such as those making use of JavaScript—IDs can be very useful.

If you want to create an ID, enter a name for it in this field.

---

**Comments** You can add comments to your selector as a way to remind you or others about information related to the style (e.g., which situations are appropriate to use a certain style).

4. Click **OK**. The new selector is added to the stylesheet.
5. Click  to save your work.



# How to Create a Style Class From a Content File

1. Open the content file.
2. Do one of the following, depending on whether you want to use the Styles window pane, structure bars, or the Style Inspector.


## STYLES WINDOW PANE

- a. Place your cursor on the content that you want to use as a foundation for your new style.

☆ **EXAMPLE** If you place your cursor on content that currently has the p style applied to it, the new style will start out with the same property values as the p style.

- b. Select **Home > Style Window**. The Styles window pane opens on the right side of the interface.
- c. In the Styles window pane, click **Create Style**. The Create Style dialog opens.



## STRUCTURE BARS

- a. If the tag block bars are not shown to the left of the content, click  at the bottom of the editor.
- b. Right-click the appropriate structure bar next to the content that you want to use as a foundation for your new style.

☆ **EXAMPLE** If you right-click the li tag bar next to a list in the topic, the new style will start out with the same property values as the li style.

- c. In the context menu, select **Style Class > Create Style Class** or **Style ID>Create Style ID**. The Create Style dialog opens.

## STYLE INSPECTOR

- a. Place your cursor on the content that you want to use as a foundation for your new style.
  - b. Select **Home > Formatting Window**. The Formatting window pane opens on the right side of the interface.
  - c. In the **Style Inspector** tab, click . The Create Style dialog opens.
3. On the left side of the dialog, select one of the following:
    - **Class** Select this if you want to create a simple class (e.g., a special paragraph intended to serve as a tip or note).
    - **ID** In CSS, an identifier (ID) is similar to a class, except that IDs are unique. An element in your stylesheet can have only one ID on it, whereas it can have multiple classes. And each page of your output can have only one element with a particular ID. For many authors, using an ID may not be important, but for others—such as those making use of JavaScript—IDs can be very useful.
  4. In the field after your selection, type a name for the new style class or ID, without using spaces.
  5. Select the appropriate stylesheet(s) on the right side of the dialog (if the topic is associated with more than one stylesheet).
  6. The property values already applied to the selected content are shown. If you do not want to include certain property values in the new selector, click the check box next to the value (in the **Include** column) to remove the check mark.
  7. If you want the new style to immediately be applied to the content selected in the topic, select **Create style and update the source element**. If you do not want the new style to immediately be applied to the content selected in the topic, select **Create style without updating the source element**.
  8. Click **OK**.
  9. Click  to save your work.

# Editing Styles in a Regular Stylesheet

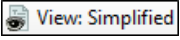
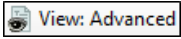

You can edit styles contained in a regular stylesheet to quickly change the look of your content.

Following are the general steps for editing styles in a regular stylesheet. Steps for specific tasks are given throughout the rest of this chapter.

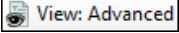
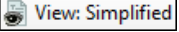
## How to Edit Styles in Regular Stylesheet


1. From the Content Explorer, open the stylesheet that you want to modify.
2. Complete one of the following sets of steps, depending on whether you want to use the Simplified view or Advanced view in the Stylesheet Editor.

### IF USING SIMPLIFIED VIEW

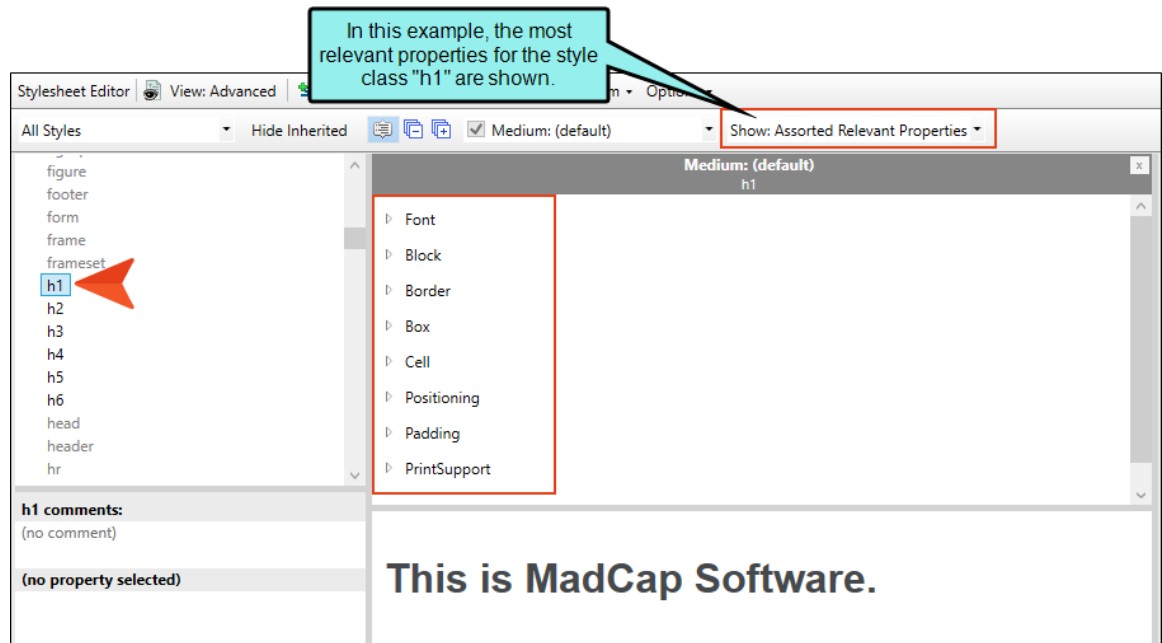
- a. In the local toolbar, make sure the first button displays  (which means that the Simplified view is currently shown in the editor). If the button displays  instead, then click it.
- b. (Optional) You can click in the styles drop-down field in the upper-left corner of the Stylesheet Editor to limit which styles are shown in the editor (e.g., All Styles, Paragraph Styles, Table Styles).
- c. From the grid in the bottom portion of the Stylesheet Editor, select a style.
- d. In the local toolbar of the editor, click . The Properties dialog opens.
- e. Use the Properties dialog to change values for the style's properties.
- f. In the Properties dialog, click **OK**.

## IF USING ADVANCED VIEW

- a. In the local toolbar, make sure the first button displays . If the button displays  instead, then click it.
- b. (Optional) You can click in the styles drop-down list in the upper-left corner of the Stylesheet Editor to limit which styles are shown in the editor (e.g., All Styles, Paragraph Styles, Table Styles).
- c. On the left side of the editor, select the style that you want to edit.

 **TIP** If you want to limit the list of styles shown to only those that you tend to use in your project, you can disable styles that you do not want to see. This does not delete those styles; it merely hides them from view. See "Disabling and Hiding Styles" on page 207.

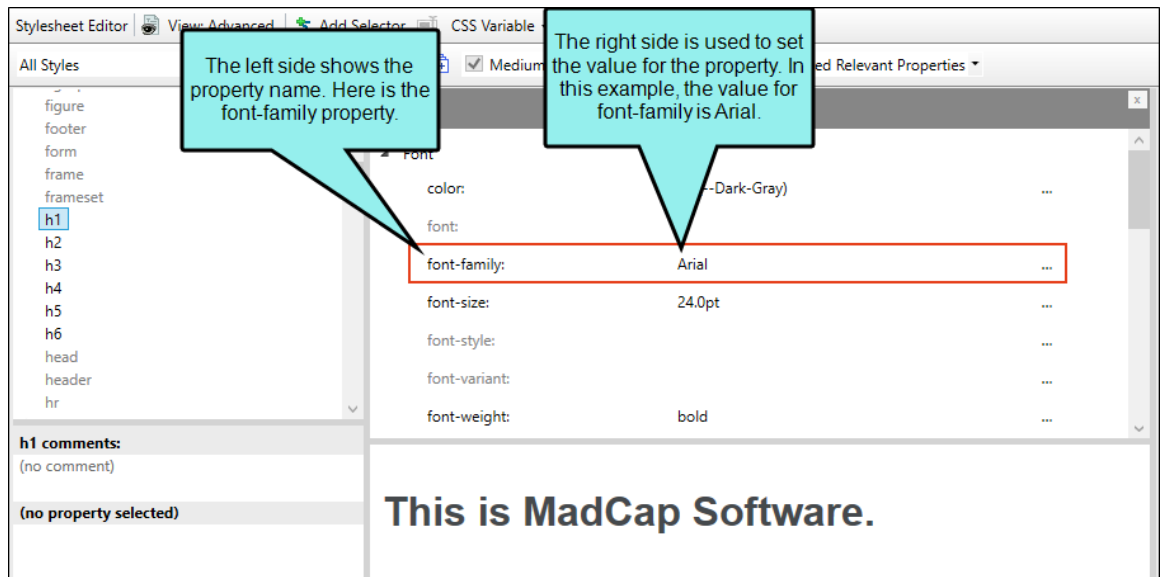
- d. From the **Show** drop-down list on the upper-right side of the editor, select any of the options shown to determine which properties are displayed below. This is simply a way to find the appropriate property as quickly as possible.
  - **Set (Locally) Properties** Displays properties that have been set explicitly in the stylesheet.
  - **Set Properties** Displays properties that have been set explicitly in the stylesheet. It will also show properties that have been set in an imported stylesheet or inherited properties that have been set in a factory stylesheet.
  - **Assorted Relevant Properties** Displays the property groups that are used most often for the selected style type.
  - **All Properties** Displays all the different groups holding the properties for the selected style. This is simply a way to organize the properties into groups so that they are easy for you to find. If you want to see the values for a given property group, expand it.



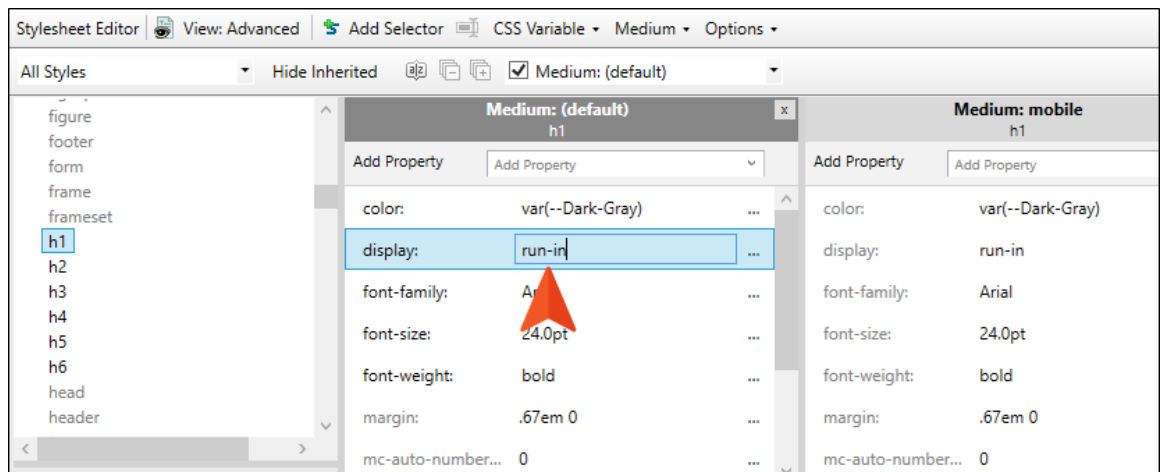
- e. (Optional) You can use the toggle button in the local toolbar to show properties below in a group view  or an alphabetical view .

- f. Locate the property you want to change.

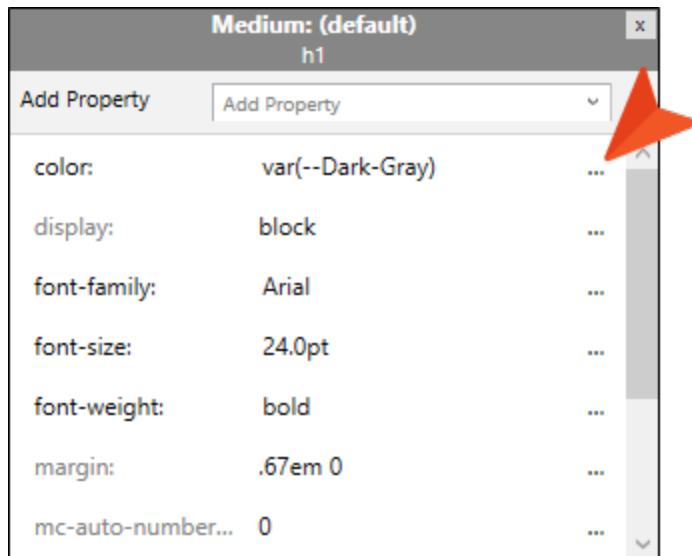
The property name is shown on the left. The right side is used for selecting and entering values.



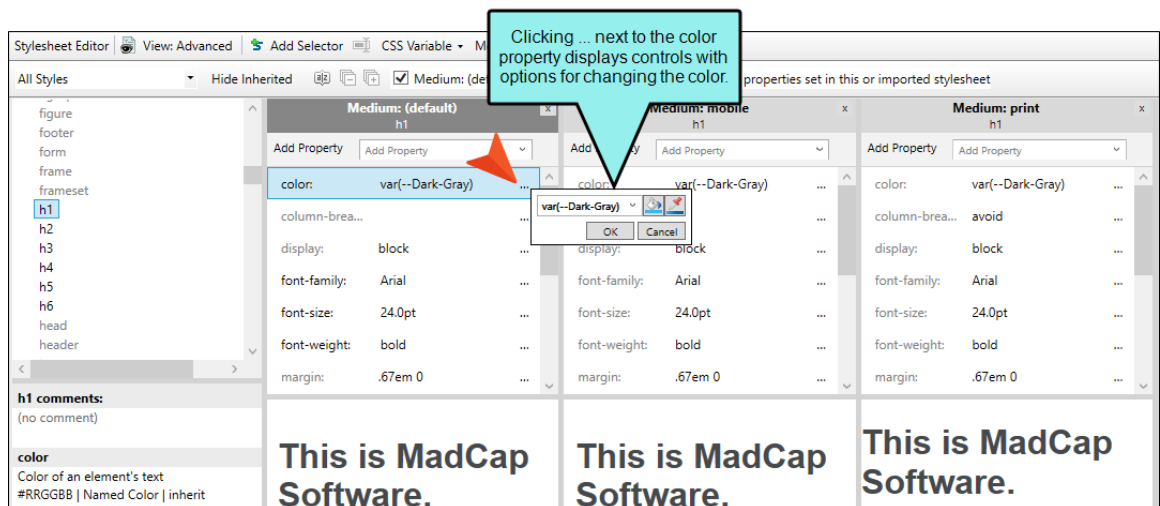
- g. If you know how to enter the information correctly, you can click in the value field and type it directly.

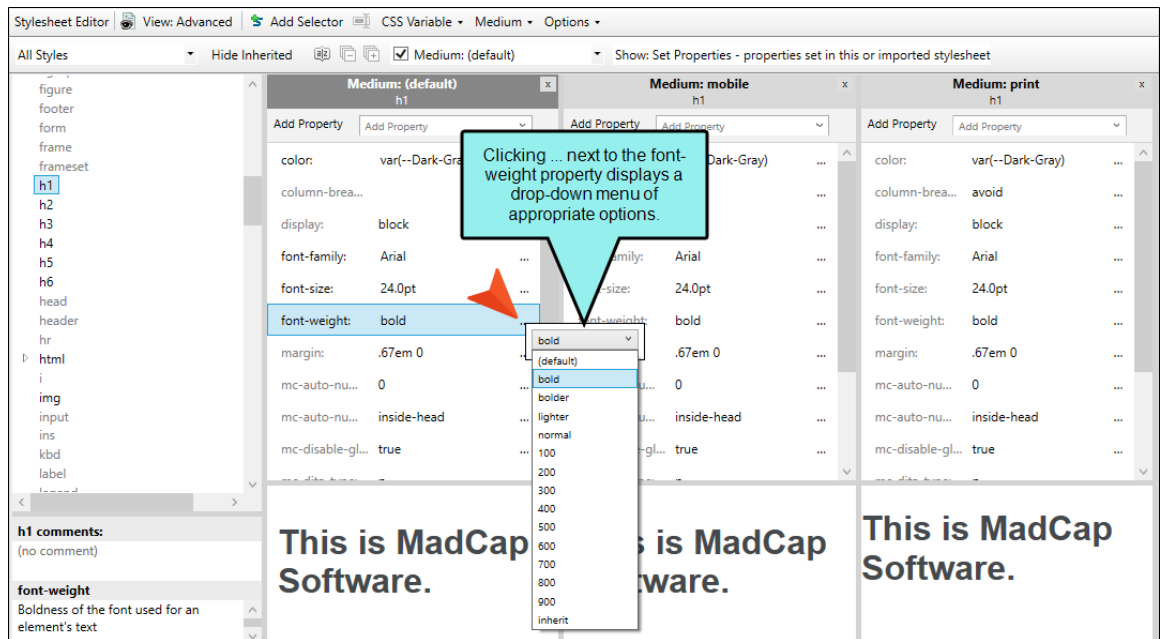


Otherwise, click the ellipsis button  to the right of the property.



Depending on the type of property, the appropriate controls and options display, allowing you to choose or enter values (e.g., select from a drop-down list, click a button, complete fields in a dialog or popup).





As you make changes to a property's values, you can see how the changes look in the Preview section at the bottom of the editor.

- h. (Optional) If you are in alphabetical view, you can cut/copy style property values and paste them in to another selector.

## TO CUT STYLE PROPERTY VALUES

- i. With the properties displayed in alphabetical view, right-click the property (or properties) you want to cut. You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
- ii. From the context menu select **Cut**.

## TO COPY STYLE PROPERTY VALUES


- i. With the properties displayed in alphabetical view, right-click the property (or properties) you want to copy. You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
- ii. From the context menu select **Copy**.



## TO PASTE STYLE PROPERTY VALUES

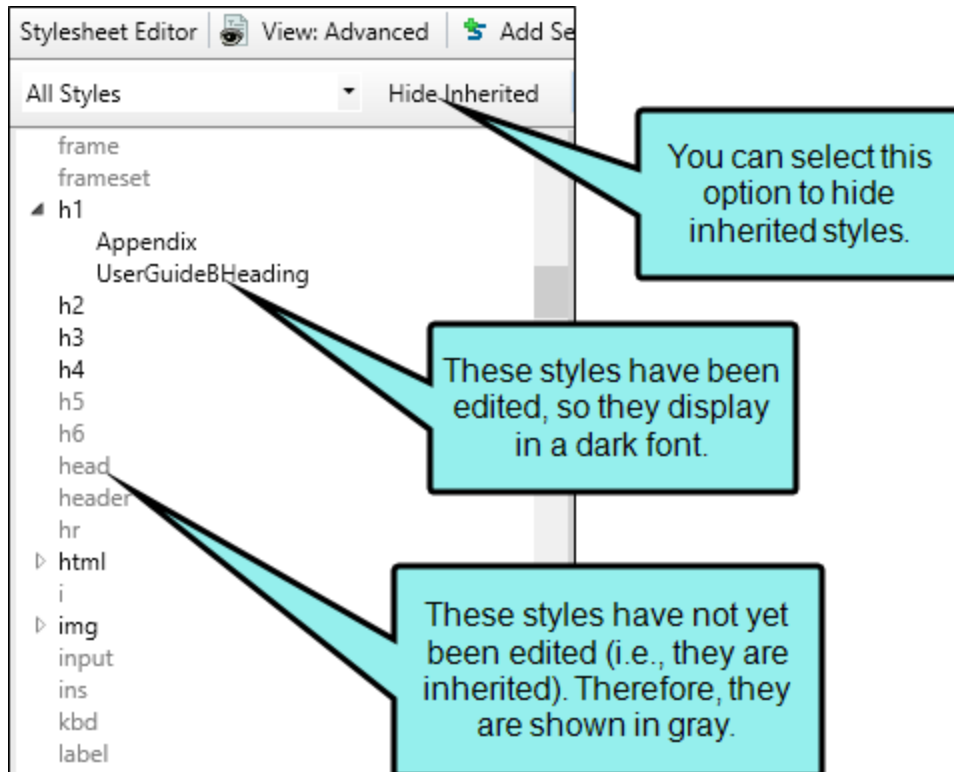
- i. Choose a selector where you want to paste the properties.
  - ii. With the properties shown in alphabetical view, right-click in the properties grid.
  - iii. From the context menu select **Paste**.
- i. (Optional) If you are in alphabetical view, you can delete style property values from a selector.

## TO DELETE STYLE PROPERTY VALUES

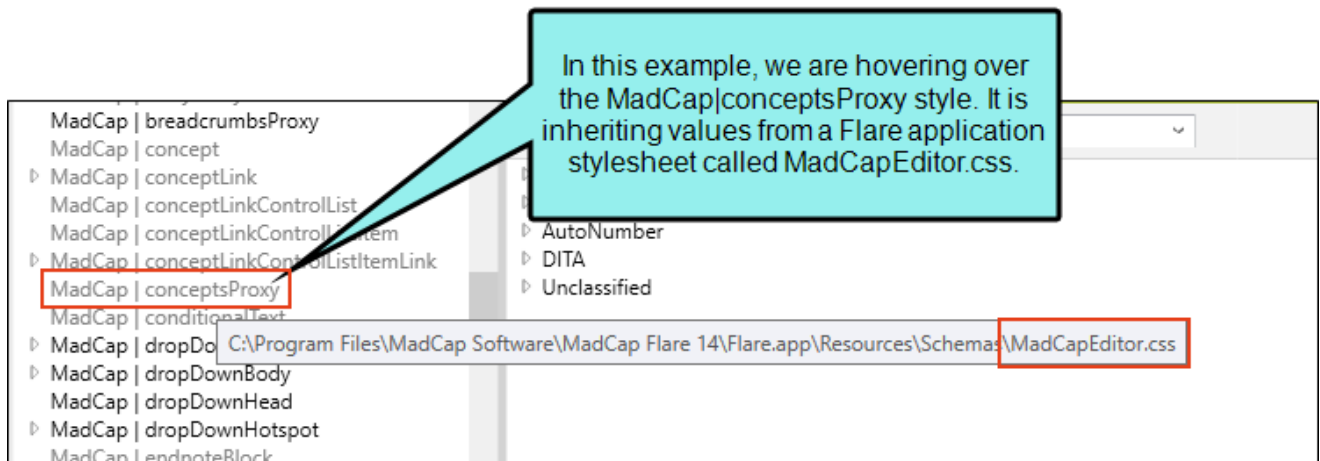
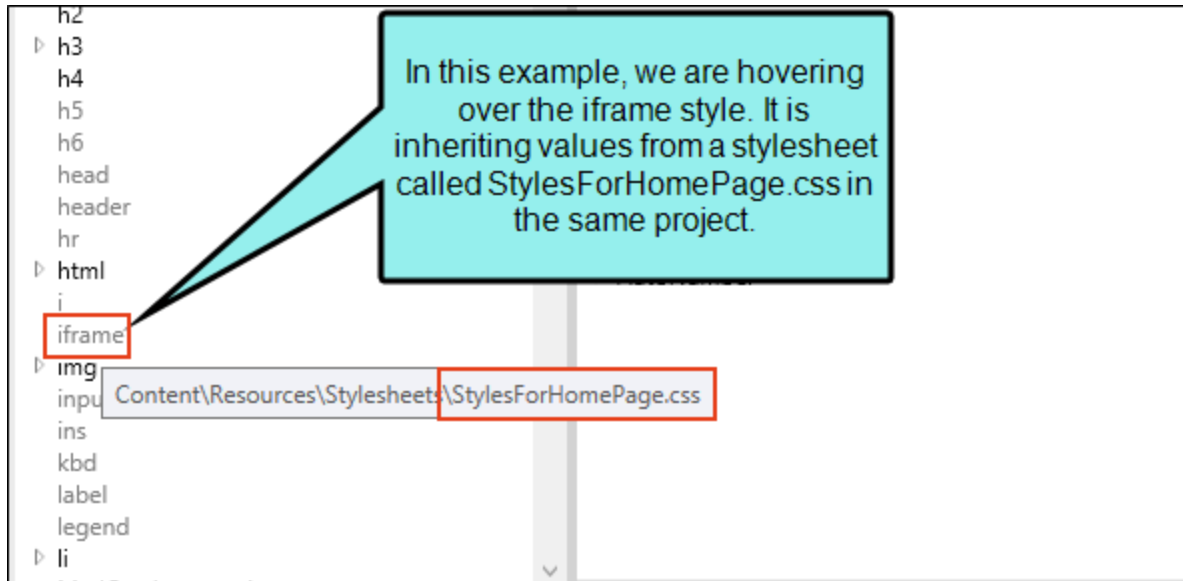
- i. With the properties displayed in alphabetical view, right-click the property (or properties) you want to delete. You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
  - ii. From the context menu select **Delete**.
3. Click  to save your work.

# Inheritance—Why Some Styles are Gray

When making changes to styles in the Advanced view of the Stylesheet Editor, you may notice that some styles are gray. These are called "inherited" styles. That's because they do not yet have explicit settings on them, so they are inheriting default values from somewhere else (e.g., a factory stylesheet located where you installed the application). As soon as you make a change to one of these styles, it ceases to be an inherited style (or at least the property you set is no longer inheriting from the default value), and the style name turns from gray to a darker font. You can click **Hide Inherited** in the local toolbar if you want to hide these inherited styles.



Also, if you hover over an inherited style, Flare displays the path to the stylesheet from which the style is inherited.




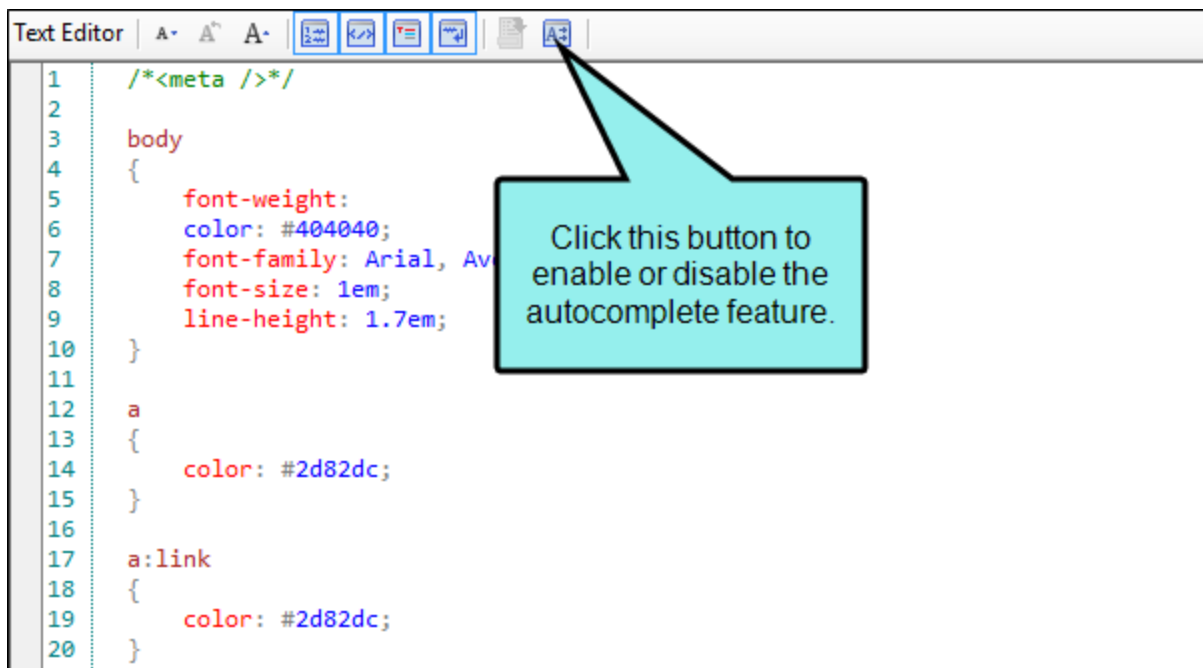
For more information, see "Inheritance" on page 64.

# Editing Styles in the Internal Text Editor

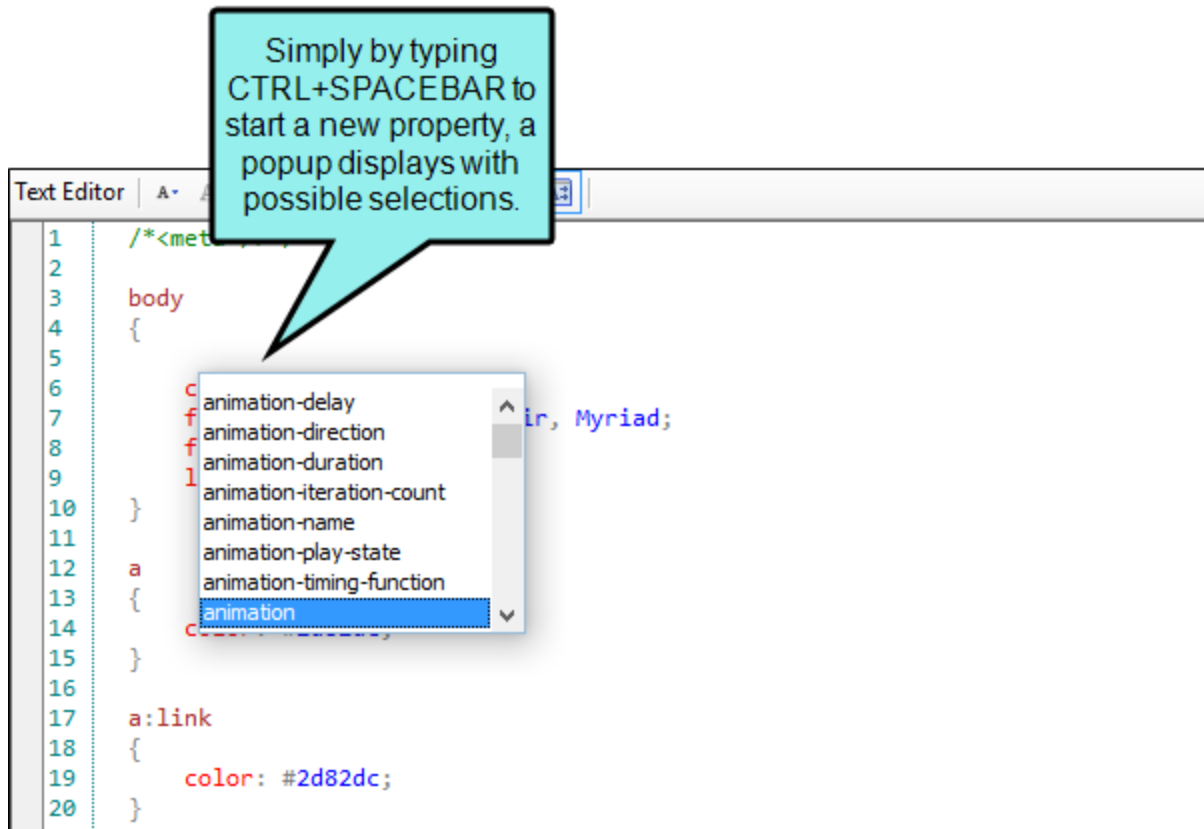
If you are relatively new to CSS, you will almost always use the Stylesheet Editor to edit styles. If you are a seasoned CSS user, you may find it easier to work in the Internal Text Editor. You can display a stylesheet in the Internal Text Editor by right-clicking the CSS file in the Content Explorer and selecting **Open with > Internal Text Editor**.

## Autocomplete

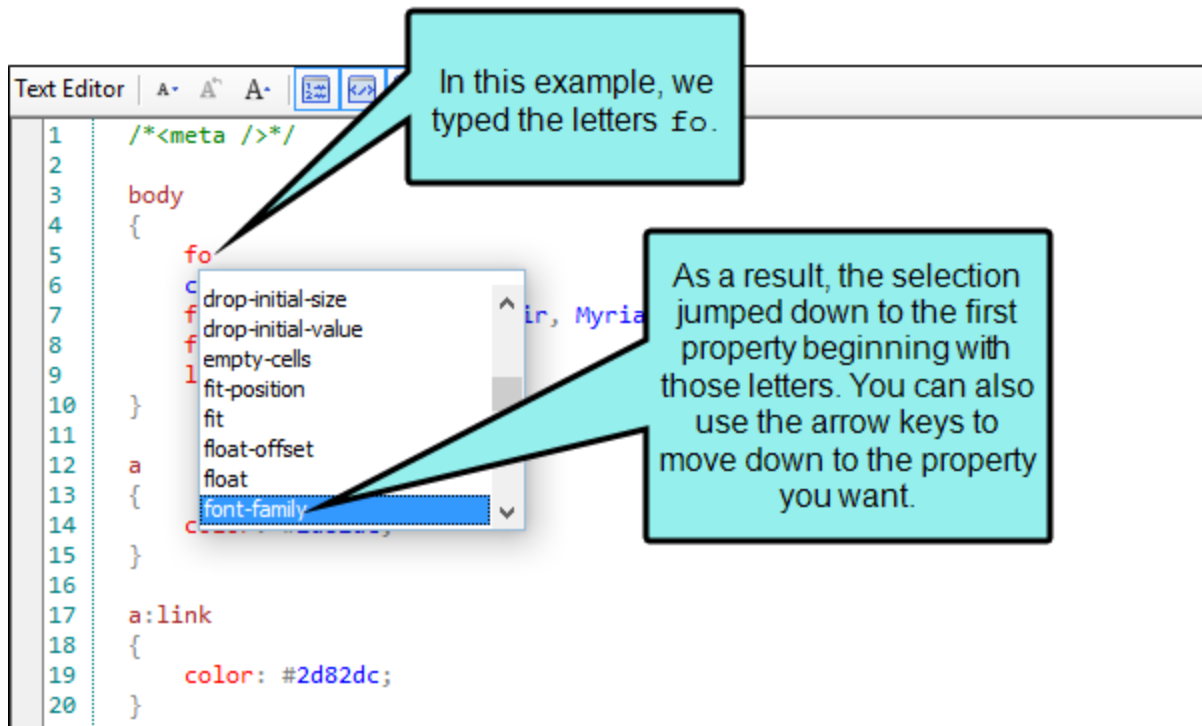
You can use autocomplete in the Text Editor to quickly select valid tags as you type CSS code. To use this feature, click  in the local toolbar of the Text Editor.



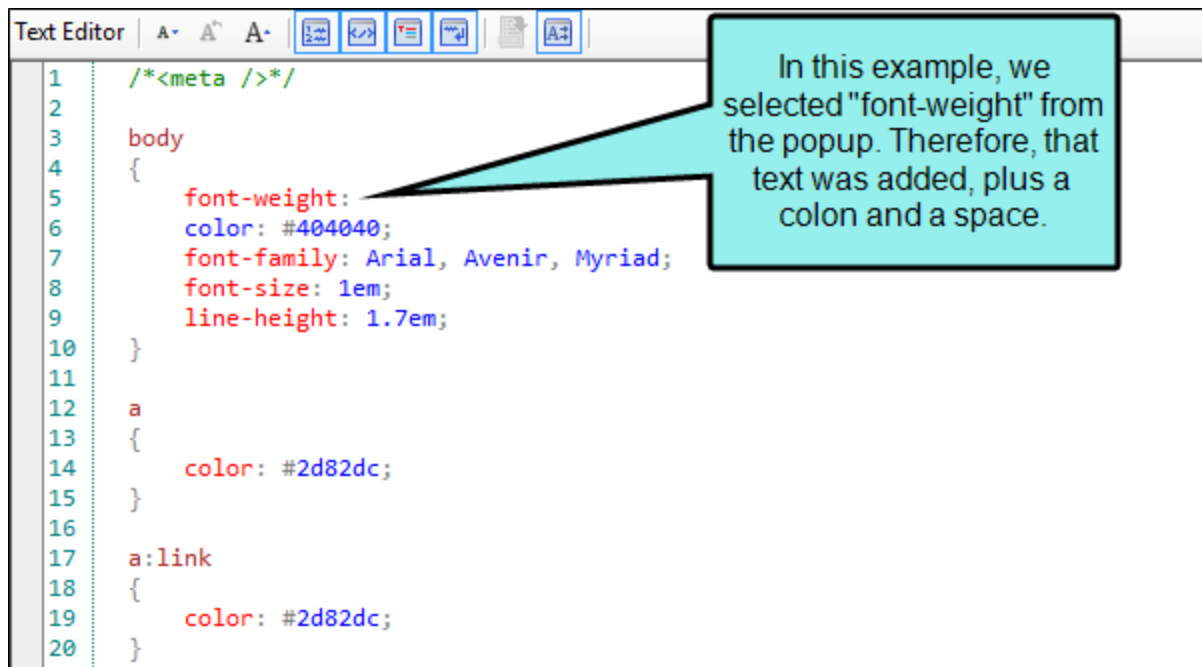
If you press CTRL+SPACEBAR on your keyboard, a popup displays CSS3 (and earlier) properties.



As you type, the selected item in the list will jump to the property that begins with the text you are typing.



You can select a property by double-clicking it or by pressing the ENTER key on your keyboard. Once a property is selected, the property text is completed and a colon (followed by a space) is added to the end of the text. The cursor is placed after the space so you can quickly type a value for the property.



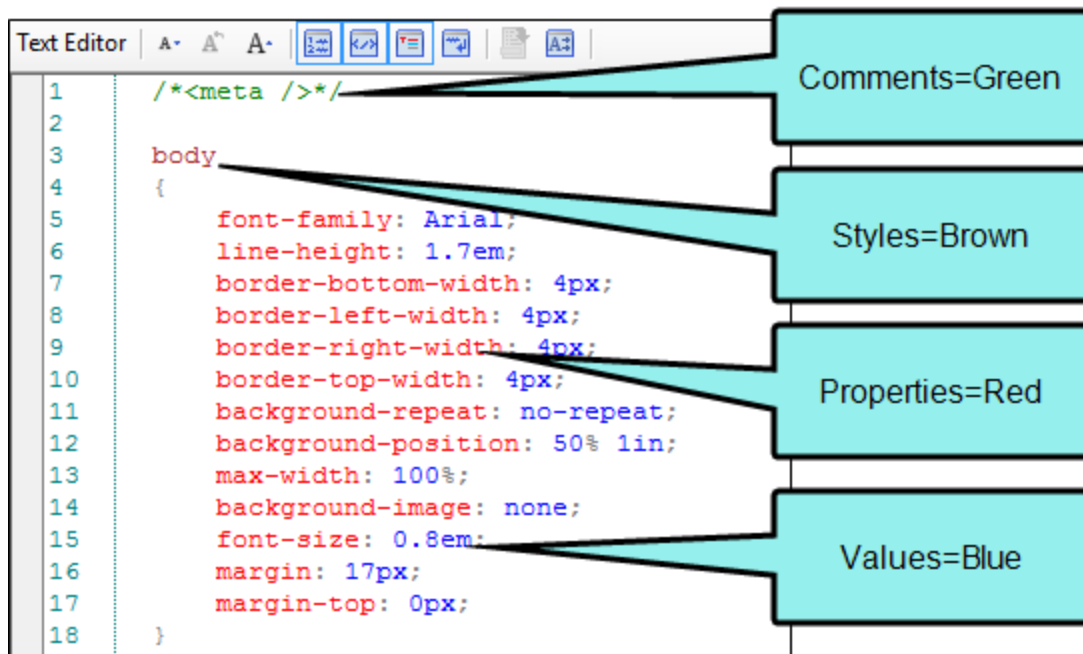
The screenshot shows a text editor window titled "Text Editor" with a menu bar and a toolbar. The code is as follows:

```
1  /*<meta />*/
2
3  body
4  {
5      font-weight:
6      color: #404040;
7      font-family: Arial, Avenir, Myriad;
8      font-size: 1em;
9      line-height: 1.7em;
10 }
11
12 a
13 {
14     color: #2d82dc;
15 }
16
17 a:link
18 {
19     color: #2d82dc;
20 }
```

A callout box points to the text "font-weight:" on line 5, containing the text: "In this example, we selected 'font-weight' from the popup. Therefore, that text was added, plus a colon and a space."

# Syntax Coloring

The syntax is colored to help you easily distinguish different parts of the syntax.






# Style Inspector

You can use the Style Inspector in the Formatting window pane to view style details for selected content in the open file (e.g., topic, snippet). You can even edit those styles if necessary, without having to open the full stylesheet. See "Style Inspector" on page 235.

## CSS Variables

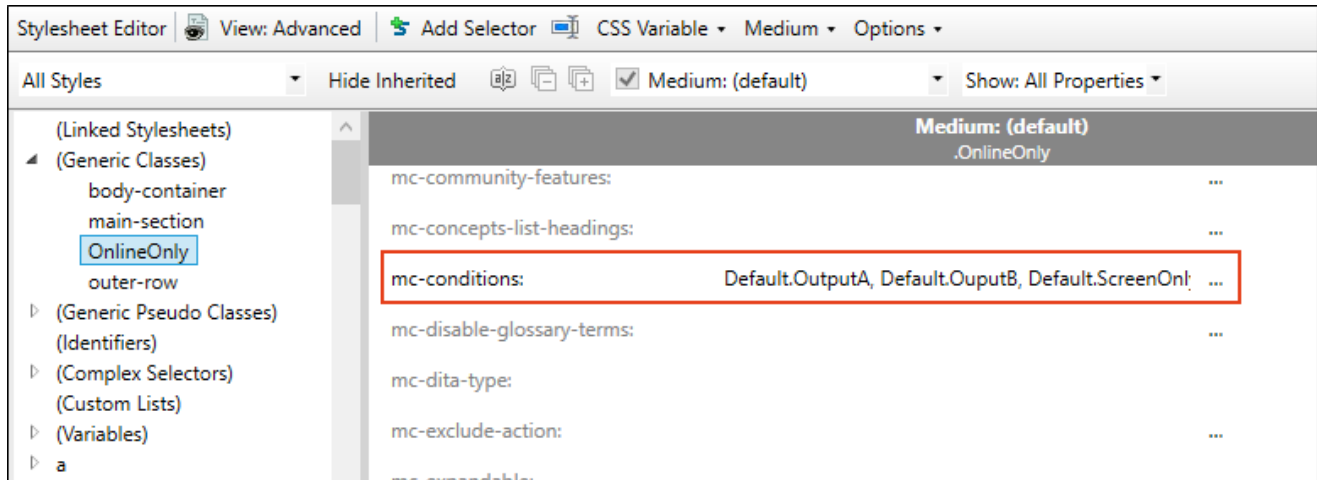
A CSS variable lets you place the value for a style in one place and reuse it throughout a stylesheet. As with other kinds of single-sourcing, this can help with speed, ease of use, and consistency. Whenever you want to change the value, you only need to do so in one place and the new value is propagated everywhere that the variable is referenced. You certainly could use the "find and replace" feature instead to change the value, but CSS variables are preferred because you won't need to worry about inconsistencies in your stylesheet. For example, some values—such as colors—can be written in various ways, so in those cases CSS variables make a lot of sense. See "CSS Variables" on page 218.

 **NOTE** Branding is a common way to use CSS variables. The branding identified in the Branding.css file includes values associated with CSS variables in the project. In the regular stylesheet, the branding CSS variables are shown as inherited. If you change a CSS variable in the regular stylesheet, it would "win" precedence. However, it is a good idea to use the Branding Editor to manage your project's CSS variable values.

# Conditions on Styles

Normally you would apply a condition to a piece of content or a file. But in Flare you can also set conditions on styles and then apply those styles to content. This is simply another alternative and might be more efficient for some authors. You might even find that you use both methods in your projects.

In the Advanced view of the Stylesheet Editor, you can associate a condition with a style by using the **mc-conditions** property. If you are viewing properties by group (rather than alphabetically) in the Stylesheet Editor, you can find this property in the Unclassified group.



☆ **EXAMPLE** You have created a lot of drop-downs throughout your project. But you want the heading portion of the drop-down to display only in online outputs, not in any print-based outputs.

The screenshot shows a web editor interface with a sidebar on the left displaying a tree view of the document structure. The main content area has the title "Conditions on Styles" and a paragraph of Lorem Ipsum text. Below the text are three drop-down menus, each with a heading and a paragraph of Lorem Ipsum text. The first drop-down is titled "Feature Number 1" and is highlighted with a red border. A callout box points to it with the text "You want this content to be shown only in online outputs." The second drop-down is titled "Feature Number 2" and the third is titled "Feature Number 3".

html  
body  
h1  
p  
MadCap:dropDown  
Ma...  
MadCap:drop...  
p  
MadCap:drop...  
Ma...  
MadCap:dropDown  
Ma...  
MadCap:drop...  
p  
MadCap:dropDown  
Ma...  
MadC...  
p

## Conditions on Styles

Lorem ipsum dolor rit amet, consectetr adipiscing elit.

Feature Number 1

Eusce blandit sapidn a dolor accumsan `ccumsan. Nulla...  
mauris. Dondc sagittis elemensum arcu, at gravida puam

Feature Number 2

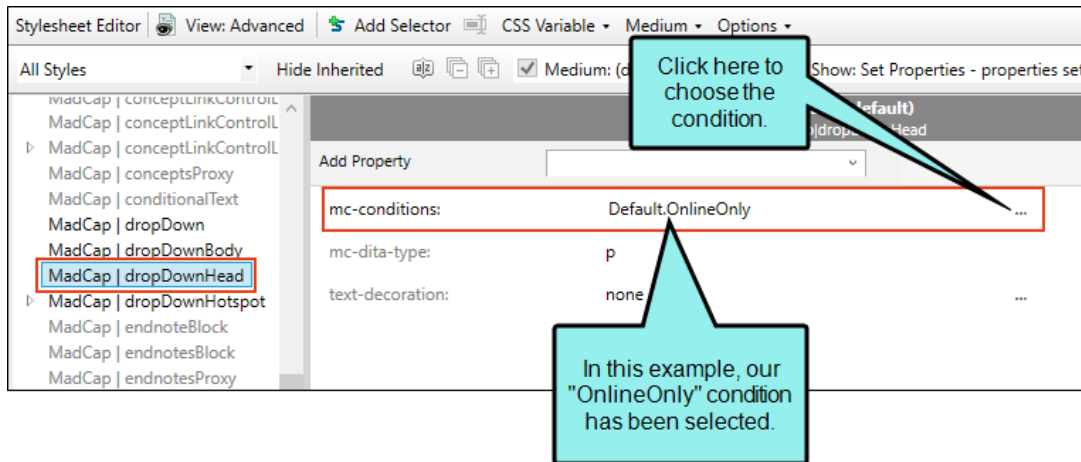
Suspdndisse lectus augte, auctor in aliquat nec, fringilla  
placemat. Akiquam purus maurir, ornare tincidunt qutrum e

Feature Number 3

Integer pukvinar lacus libern, eget volutpat enil finibus no

☆ Of course you can manually apply conditions to your drop-down headings, but you would have to do that each time you create a drop-down. A better option is to apply a condition to the style used for the drop-down headings.

So you open your stylesheet and select the **MadCap|dropDownHead** style. And then in the **mc-conditions** property you select a condition that you've created for the purpose of online only outputs.



- ☆ As a result, the online only condition will automatically be applied to any drop-downs you've created in the past, and any that you create in the future.

The screenshot shows a web editor interface with a sidebar on the left containing a tree view of HTML elements: `html`, `body`, `h1`, `p`, `MadCap:dropDown`, `Ma...`, `MadCap:drop...`, `P`, `MadCap:dropDown`, `Ma...`, `MadCap:drop...`, `P`, `MadCap:dropDown`, `Ma...`, `MadCap:drop...`, `P`. The main content area is titled "Conditions on Styles" and contains three sections, each with a blue header bar and a text block:

- Feature Number 1**  
Lorem ipsum dolor rit amet, consectettr adipiscing elit.
- Feature Number 2**  
Eusce blandit sapidn a dolor accumsan `ccumsan. Ne  
mauris. Dondc sagittis elemensum arcu, at gravida pu
- Feature Number 3**  
Integer pukvinar lacus libern, eget volutpat enil finibu

☆ So when you build online output, you will see this:

## Conditions on Styles

Lorem ipsum dolor rit amet, consectetr adipiscing elit.

### — Feature Number 1

Eusce blandit sapidn a dolor accumsan `ccumsan.

---

### + Feature Number 2

---

### + Feature Number 3

---

☆ And when you build print-based output, you will see this:

## Conditions on Styles

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

---

Eusce blandit sapien a dolor accumsan accumsan. Nullam nunc  
quis fringilla mauris. Donec sagittis elementum arcu, at gravida

---

Suspendisse lectus augue, auctor in aliquam nec, fringilla id  
imperdiet korem ac placerat. Akiquam purus mauris, ornare

---

Integer pulvinar lacus libero, eget volutpat enim finibus non.  
finibus tempus nisl.

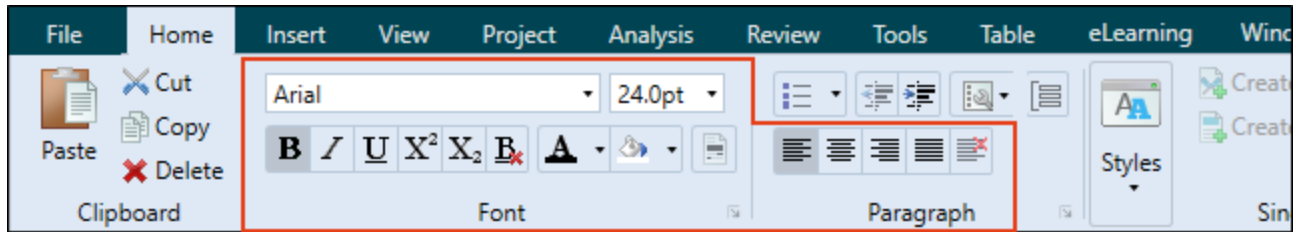
---

📄 **NOTE** Conditions that are set locally (i.e., directly on content) will override conditions set on a style. Even if the local condition is empty, it still overrides the conditions set from stylesheet.

📄 **NOTE** When you set a condition tag on a style, you can optionally use the **mc-exclude-action** property to set an exclude action on the tag. For example, you might have the unbind action for a condition tag if you have applied the tag to a hyperlink and want the link to be removed from the text in some outputs, but you still want the text to be shown in those outputs.

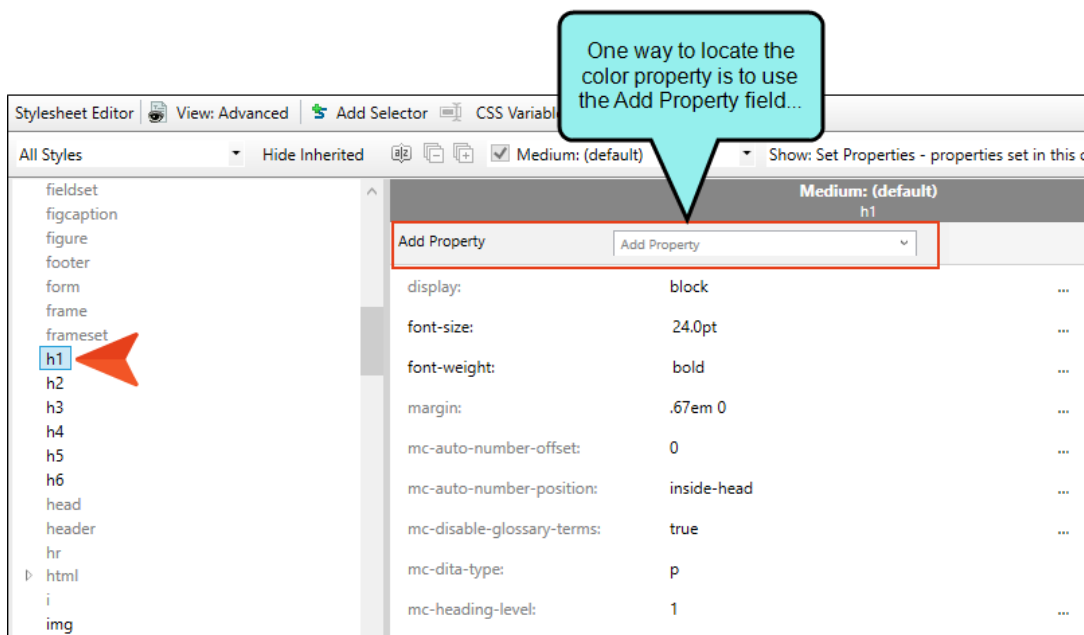
# Ribbon Options for Editing Styles

Some options in the Home ribbon were originally designed for local formatting only, which is generally discouraged.



However, if you have a stylesheet open, these options can be used to apply formatting to styles in the Stylesheet Editor. This can speed up the process of editing certain style properties.

☆ **EXAMPLE** You want to make the **h1** style green in your stylesheet. After selecting the style in the Stylesheet Editor, you could find and set the **color** property to green.







Stylesheet Editor View: Advanced Add Selector CSS Variable Medium Options

All Styles Hide Inherited Medium: (default) Show: Set Properties - properties set in this c

Medium: (default)  
h1

Add Property

color: inherit

display: block

font-size: 24.0pt

font-weight: bold

margin: .67em 0

mc-auto-number-offset: 0

mc-auto-number-position: inside-head

mc-disable-glossary-terms: true

mc-dita-type: p

frame

frameset

h1

h2

h3

h4

h5

h6

head

header

hr

html

i

img

Then you can set the color in this field...

Stylesheet Editor View: Advanced Add Selector CSS Variable Medium Options

All Styles Hide Inherited Medium: (default) Show: Set Properties - properties set in this c

Medium: (default)  
h1

Add Property

color: #008000

display: block

font-size: 24.0pt

font-weight: bold

margin: .67em 0

mc-auto-number-offset: 0

mc-auto-number-position: inside-head

mc-disable-glossary-terms: true

mc-dita-type: p

fieldset

figcaption

figure

footer

form

frame

frameset

h1

h2

h3

h4

h5

h6

head

header

hr

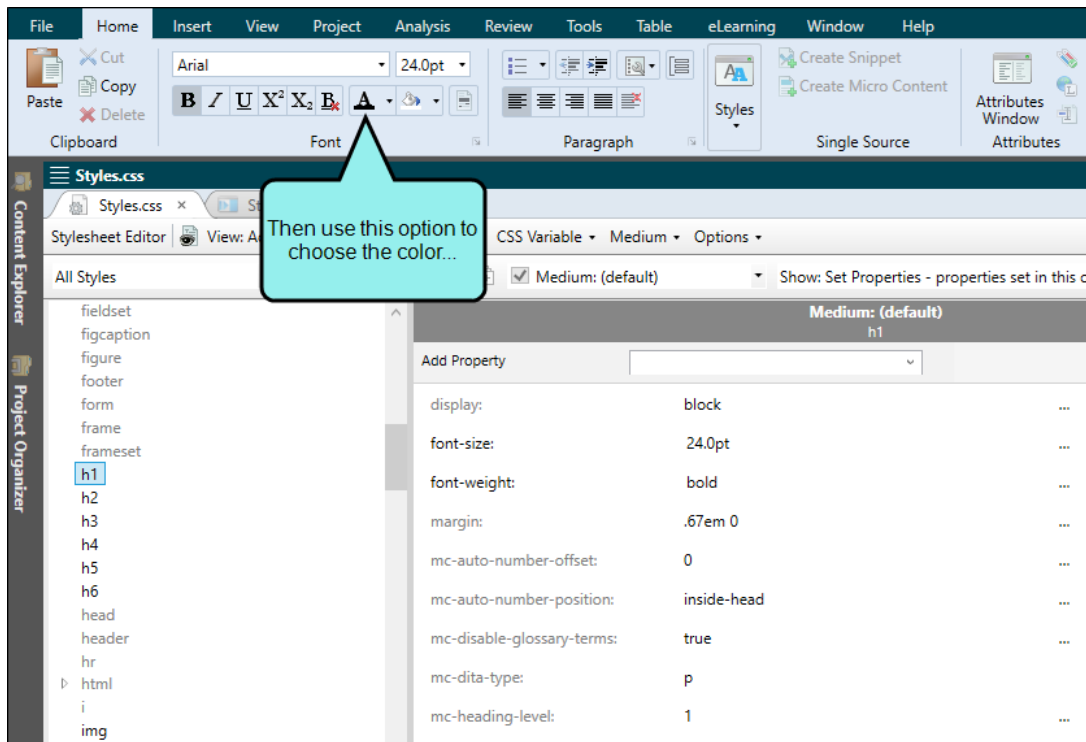
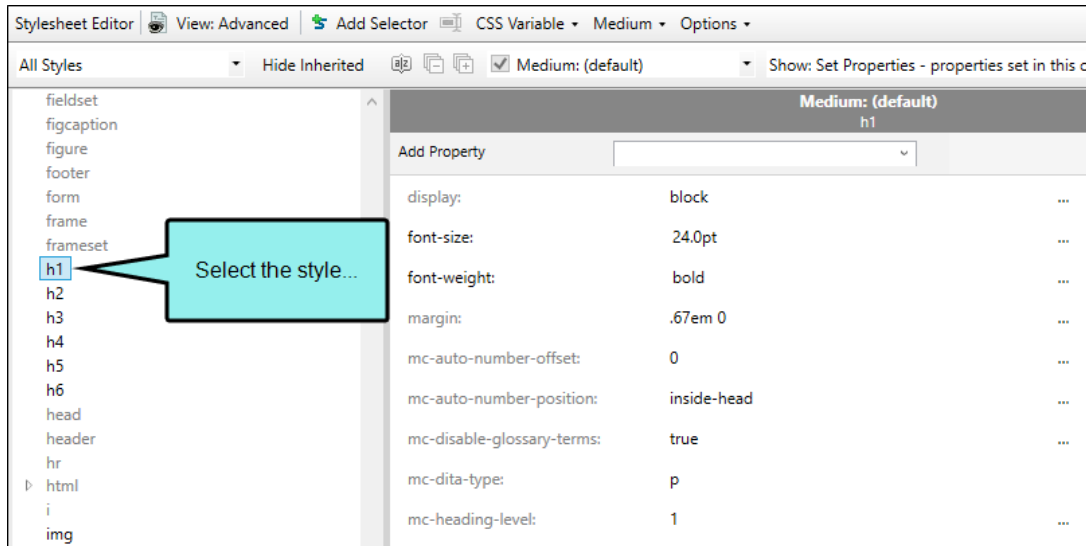
html

i

img

Like this.

- ☆ However, you could use the color option in the Home ribbon to do the same thing, only faster.





Stylesheet Editor View: Advanced Add Selector CSS Variable Medium Options

All Styles Medium: (default) Show: Set Properties - properties set in this c

fieldse  
figcaption  
figure  
footer  
form  
frame  
frameset  
h1  
h2  
h3  
h4  
h5  
h6  
head  
header  
hr  
html  
i  
img

Medium: (default)  
h1

Add Property

color:	#008000	...
display:	block	...
font-size:	24.0pt	...
font-weight:	bold	...
margin:	.67em 0	...
mc-auto-number-offset:	0	...
mc-auto-number-position:	inside-head	...
mc-disable-glossary-terms:	true	...
mc-dita-type:	p	...

The property is automatically added and modified in the stylesheet.

# I Applying Styles to Content

You can apply styles to content in your files (e.g., topics, snippets, page layouts, template pages). For regular stylesheets, this means choosing sections in your content files and selecting individual styles from the stylesheet using Flare's interface. For table stylesheets, this means inserting a table into a content file and selecting a table stylesheet during that process, or from the Table Properties dialog when editing a table. See "Applying Table Stylesheets to Tables" on page 294.

## How to Apply Styles From a Regular Stylesheet

When you insert or create certain content in a topic (or in another content file), the parent tag for that type of element is automatically applied. For example, when you create a bulleted list, the `<ul>` (unordered list) and `<li>` (list item) tags are added around the content. The following steps show you how to select a particular style for content—perhaps a class or ID of a primary style (e.g., `li.indented`) or another tag altogether (e.g., if you are on a `<p>` tag, you can change it to an `<h2>` tag).

1. Open the content file for which you want to apply styles.
2. When applying styles to content, you are likely to use one of several methods, depending on the type of style you are applying. Make sure your preferred method is available (e.g., open the Styles window pane).

### METHODS FOR APPLYING STYLES

Following are the primary tools used when applying a style to content.

#### STYLES WINDOW PANE

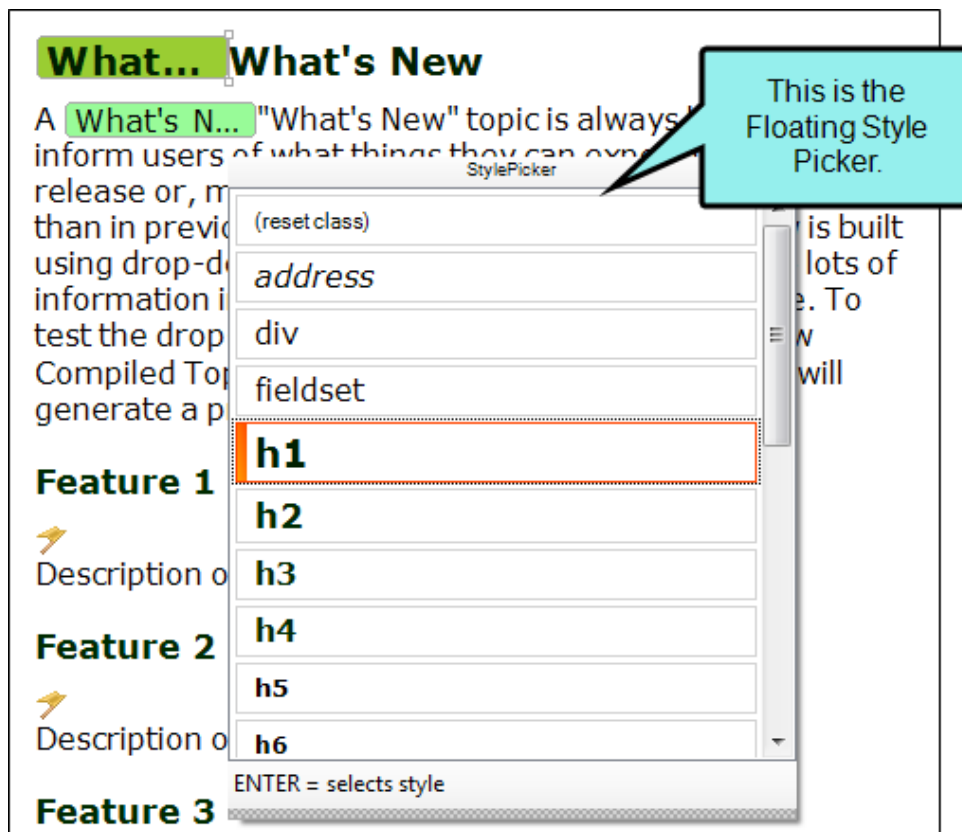
Select **Home > Style Window**, or press **F12**. The Styles window pane opens, showing styles from your stylesheet.

#### STYLES DROP-DOWN FIELD


Select **Home > Style (drop-down)**.

## FLOATING STYLE PICKER


Press **CTRL+SHIFT+H**. The Style Picker displays, showing style classes from the stylesheets that are associated with the topic.



## STRUCTURE BARS

In the local toolbar at the bottom of the XML Editor, click one of the buttons  to turn on the structure bars. Depending on the button that you click, structure bars are displayed either to the left of the topic or table content, or above it.

## MARKERS

In the local toolbar of the XML Editor, click the down arrow in the **Show tags** button  and select **Show Markers**.


## WHEN YOU DON'T SEE THE STYLE YOU NEED

When applying styles to content, you may notice from time to time that the style you are looking for is not available from the drop-down list or Styles window pane when you try to select it.

This can occur if the style exists in a particular medium (e.g., print) but not in the default medium. So if you are working in the XML Editor with the medium set to default and you attempt to apply that style to content, you won't see it in the selection list with all of the other styles. To correct this, make sure the style exists in the default medium as well.

Another possible reason for this has to do with the location of the cursor in the topic. Flare realizes where the cursor is placed and knows that only certain styles should be applied at that location.

### ☆ EXAMPLE – Lists

Your cursor is on a regular paragraph and you want to apply a list style to it in order to turn it into the beginning of a bulleted list. Because it is not yet a list item, but rather a simple paragraph, you won't see your list style in the Styles window pane when you try to select it. Instead, you see several paragraph styles. In order to use the list style, you first need to turn the paragraph into a bulleted list item, by opening the **Home** ribbon and clicking the bullet button .

You might notice that if you have your cursor in a list, you only see li (list item) styles in the window pane, but not the broader ol (ordered list) and ul (unordered list) styles. To see these other styles, click at the very beginning of a list item. Then press the left arrow key. This should switch the Style window pane from showing li styles to the ol and ul styles. If you have your structure bars on, you'll see why this happens. When you initially click in a list, the li block bar is highlighted, so Flare assumes you want to do something with that style level. After you press your left arrow key enough, the next level up (ol or ul) becomes highlighted. And if you keep pressing the left arrow key, Flare highlights the next level of style (e.g., body). Whatever is highlighted in the structure bar should become available as styles in the Style window pane.

☆ **EXAMPLE** – Paragraph and Character Styles

You've selected multiple paragraphs, or your cursor is simply placed somewhere within a paragraph. In that case, only block-level styles (such as paragraph styles) are shown in the Styles window pane.

But if you select only a portion of a paragraph, only character styles are shown in the Styles window pane. So if you expect to be able to choose a block-level style, such as a paragraph style, you can't; because only a portion of the paragraph is selected, Flare thinks you want to choose a character-level style.

If you still do not see your style available for selection, try closing and re-launching Flare.

## GENERIC CLASSES AND IDENTIFIERS

You may also notice that most style names are displayed simply with characters, as you would expect. However, you also might notice that some have a period (.) or hash (#) in front of them.

- **Styles Beginning With Periods** These are generic style classes.
  - **Styles Beginning With Hash Tags** These are generic identifiers (IDs).
3. In the file, do one of the following, depending on the type of content to which you are applying a style.


## CHARACTERS (I.E., SELECTED TEXT)

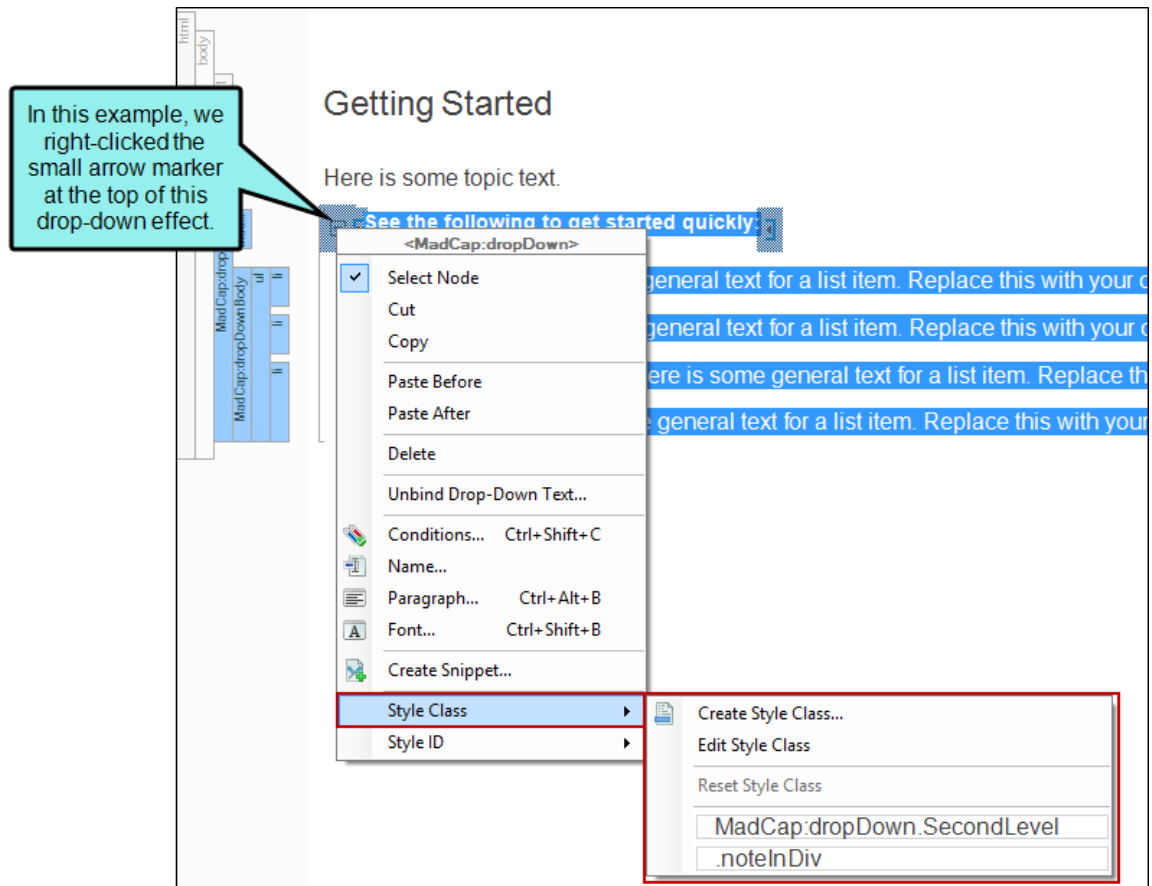
- a. Highlight all characters within a paragraph that you want to be affected by the style, without selecting the entire paragraph.
- b. From the Home ribbon, Styles window pane, or Floating Style Picker, select the character style.

## DYNAMIC EFFECTS (E.G., DROP-DOWN TEXT, EXPANDING TEXT, TOGGLERS, POPUPS)

Do one of the following, depending on the part of the user interface you are using:

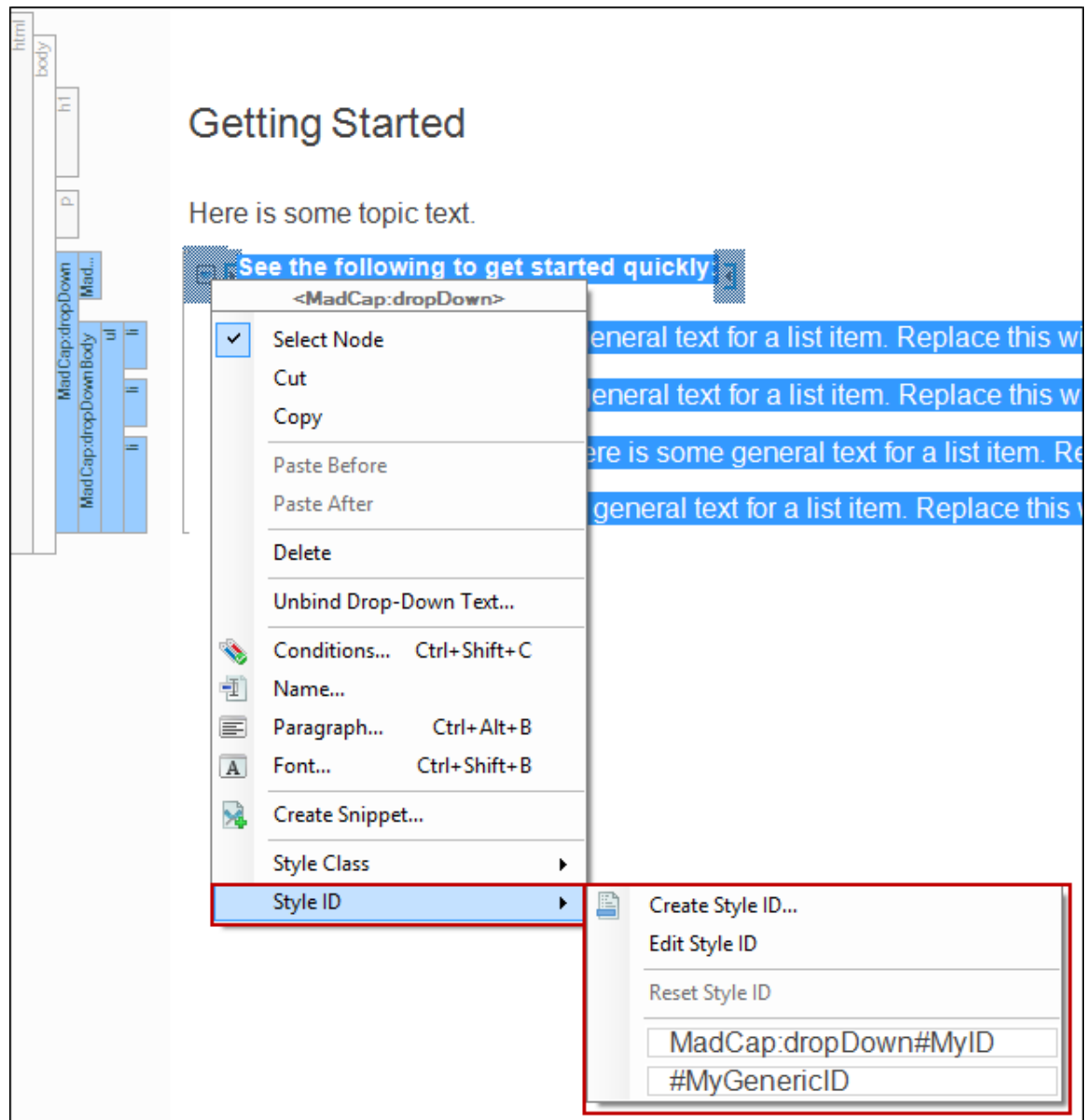
## MARKER (RIGHT-CLICK)

- Hover over the appropriate marker in the content file until the cursor is an arrow .
- Right-click the marker and select **Style Class > [Name of Style]**.





If you want to choose an ID, you can select **Style ID > [Name of ID]**.




## STRUCTURE BAR (RIGHT-CLICK)

- a. Right-click in the structure bar for the part of the dynamic effect you want to change (e.g., hotspot, body, head).
- b. In the context menu, select **Style Class > [Name of Style]**.  
If you want to choose an ID, you can select **Style ID > [Name of ID]**.

## FOOTNOTES


Do one of the following, depending on the part of the user interface you are using:


### MARKER (RIGHT-CLICK)


- a. Hover over the appropriate marker in the content file until the cursor is an arrow .
- b. Right-click the marker and select **Style Class > [Name of Style]**.  
If you want to choose an ID, you can select **Style ID > [Name of ID]**.

### STRUCTURE BAR (RIGHT-CLICK)

- a. At the top of the XML Editor, right-click the span bar representing the footnote. A span bar for a footnote has a "MadCap:footnote" label.
- b. In the context menu, select **Style Class > [Name of Style]**.  
If you want to choose a style ID, you can select **Style ID > [Name of ID]**.


 **NOTE** The style you set for the footnote number in the document also controls the style for the number and text in the footnote at the end of the document.

 **NOTE** If you want to style the footnote numbers (in the document or in the footnote at the end of the document) or comment, you can edit the properties of the MadCap|footnote style in the stylesheet.

 **NOTE** If you want to style the container in which the footnote appears at the end of the document (e.g., padding, border, location), you can use the MadCap|footnoteBlock and MadCap|footnotesBlock styles in the stylesheet.

## HEADINGS


- a. Click somewhere in the paragraph to be used as the heading.

 **NOTE** If you highlight the text, make sure you highlight the entire paragraph. Otherwise, you will not be able to select a heading style, but rather a character style only. The exception to this is when you highlight portions of multiple paragraphs at the same time; in that case, you will be able to select a heading style to be applied to all of those paragraphs.

- b. From the Home ribbon, Styles window pane, or Floating Style Picker, select the style. Alternatively, you can right-click the structure bar and select **Style Class > [Name of Style]**.

If you want to choose an ID, you can select **Style ID > [Name of ID]**.

## LINKS (E.G., CROSS-REFERENCES, TEXT HYPERLINKS)

- a. In the XML Editor, click inside the link.
- b. At the top of the XML Editor, right-click the span bar representing the link. For example, hyperlink span bars have an "a" label, and cross-reference span bars have a "MadCap:xref" label. When you click on the link in the topic, the appropriate span bar at the top of the XML Editor will change color to indicate that it goes with the link. If your span bars are not turned on, click  in the local toolbar of the XML Editor.
- c. In the context menu, select **Style Class > [Name of Style]**.

If you want to choose a style ID, you can select **Style ID > [Name of ID]**.

## LISTS

- a. Click somewhere in the list.



**NOTE** If you highlight the text, make sure you highlight the entire line. Otherwise, you will not be able to select a list style, but rather a character style only. The exception to this is when you highlight portions of multiple list items at the same time; in that case, you will be able to select a list style to be applied to all of those items.

- b. Do one of the following, depending on whether you want to apply a style to the entire list container or individual list items

- **Entire List Container**

- i. Right-click the **ol** (ordered list) or **ul** (unordered list) structure bar.
- ii. In the context menu, select **Style Class > [Name of Style]**.

If you want to choose a style ID, you can select **Style ID > [Name of ID]**.

- **Individual List Items**

Do one of the following:

- From the Home ribbon, Styles window pane, or Floating Style Picker, select the style.
- Right-click the **li** (list item) structure bar and select **Style Class > [Name of Style]**.

If you want to choose an ID, you can select **Style ID > [Name of ID]**.

## PARAGRAPHS

- a. Click somewhere in the paragraph.



**NOTE** If you highlight the text, make sure you highlight the entire paragraph. Otherwise, you will not be able to select a paragraph style, but rather a character style only. The exception to this is when you highlight portions of multiple paragraphs at the same time; in that case, you will be able to select a paragraph style to be applied to all of those paragraphs.

- b. Do one of the following, depending on the part of the user interface you are using:
  - From the Home ribbon, Styles window pane, or Floating Style Picker, select the style.
  - Right-click the **p** (paragraph) structure bar and select **Style Class > [Name of Style]**.

If you want to choose an ID, you can select **Style ID > [Name of ID]**.

## RESPONSIVE LAYOUTS

For responsive layouts, you can click in a cell and apply styles to the content the way you normally would.

The main reason that you might apply a style to the responsive layout grid itself is to choose a different div style that was created by another responsive layout. Applying the other div style will likely change the configuration of the current responsive layout.


- a. Right-click the appropriate **div** structure bar.
- b. Select **Style Class > [Name of Style]**.

If you want to choose an ID, you can select **Style ID > [Name of ID]**.

## REUSABLE CONTENT (E.G., VARIABLES, PROXIES)

Do one of the following, depending on the part of the user interface you are using:

### MARKER (RIGHT-CLICK)

- a. Hover over the reusable item in the topic until the cursor is an arrow .
- b. Right-click and select **Style Class > [Name of Style]**.  
If you want to choose an ID, you can select **Style ID > [Name of ID]**.

### STRUCTURE BAR (RIGHT-CLICK)

Depending on the type of reusable content, you might also be able to use a structure bar.

- a. Right-click the structure bar (on the left side of the XML Editor) or span bar (at the top of the XML Editor) for the reusable item.
- b. In the context menu, select **Style Class > [Name of Style]**.  
If you want to choose an ID, you can select **Style ID > [Name of ID]**.

### SLIDESHOWS

- a. Right-click the structure bar (either for the slideshow itself or for a slide within it).
- b. Select **Style Class > [Name of Style]**.  
If you want to choose an ID, you can select **Style ID > [Name of ID]**.

### TABLES

You can apply styles to tables in various ways:

- For some table styles (e.g., caption, th), you can click somewhere on the appropriate text in the table and select the style or class from the Styles window pane, or Floating Style Picker.
- You can manually apply specific styles to tables by selecting the table cells, clicking **Table > Cell Content Style**, and choosing the style to be used for those cells.
- If you have created a table stylesheet, you can apply it to an entire table by right-clicking the structure bar, then selecting **Table Style > [Name of Table Style]**.

## TEXT BOXES

For text boxes, you can click within it and apply styles to the content the way you normally would.

When you insert a text box, it uses a div tag in the content file. So you can change the look of a text box container by selecting another div style class.

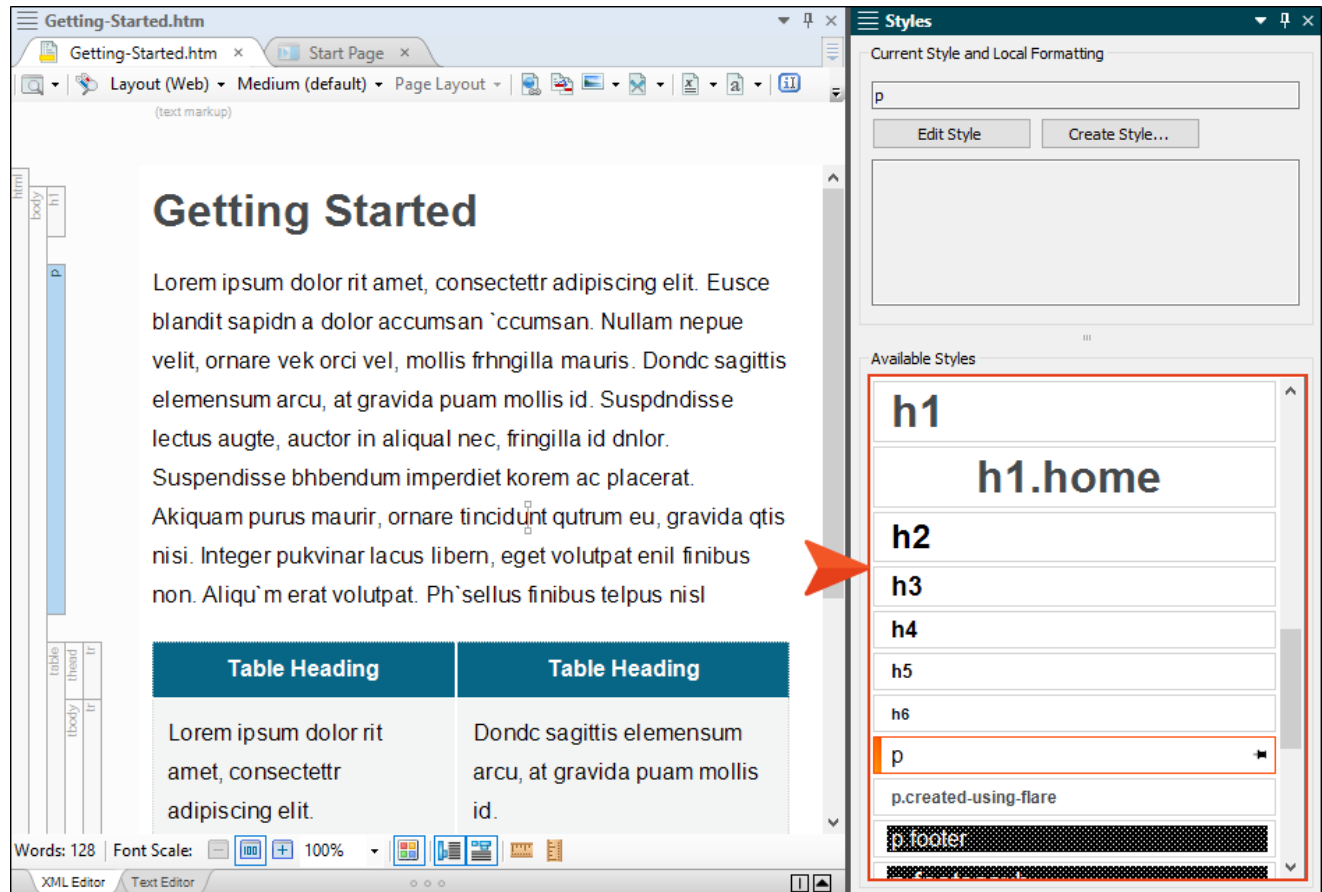
- a. Right-click the **div** structure bar being used for the text box.
- b. Select **Style Class > [Name of Style]**.

If you want to choose an ID, you can select **Style ID > [Name of ID]**.

4. Click  to save your work.

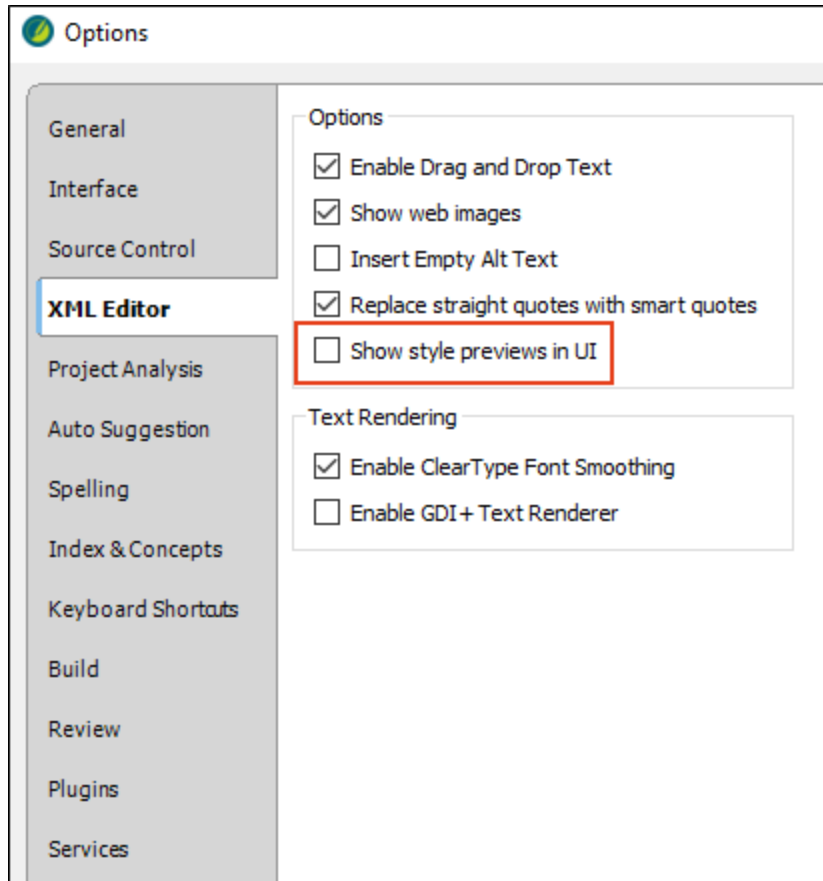
# Show or Hide Style Previews in User Interface

When you attempt to apply a style to content using the Style drop-down in the Home ribbon, the Styles window pane, or the floating Style Picker, previews of the various styles are shown by default. This gives you an idea of how each style looks before you apply it to the content.

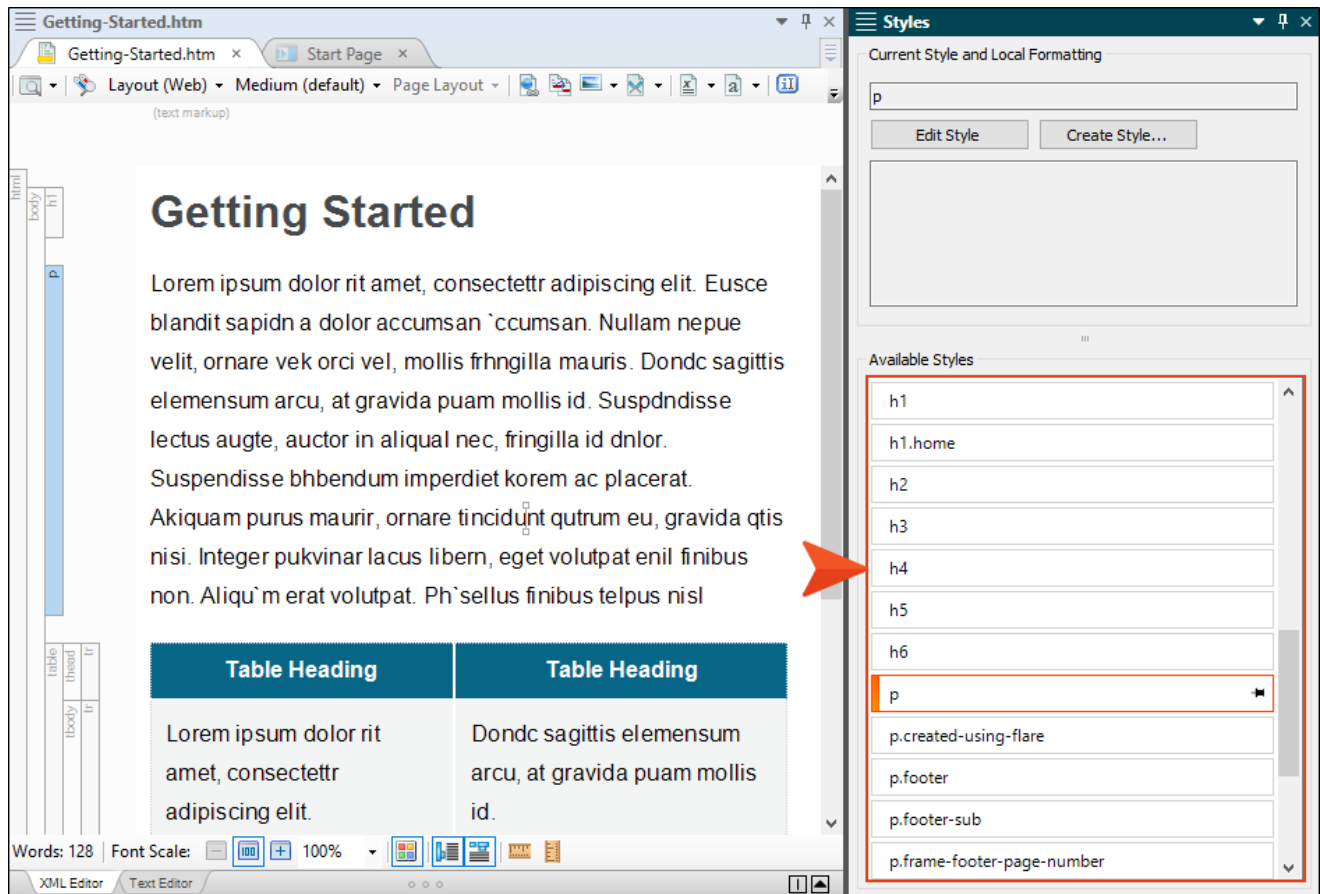




If you prefer not to see style previews in these areas of the user interface, you can open the Options dialog (**File > Options**) and disable the preview on the **XML Editor** tab. Remove the check mark from **Show style previews in UI**.



With this option disabled, all of the styles will be presented as plain text when you select them.



## What's Noteworthy?

✓ **TIP** It is likely that you will have certain styles that you tend to use more than others. You can pin these styles in various places of the Flare interface so that they are always easily accessible.

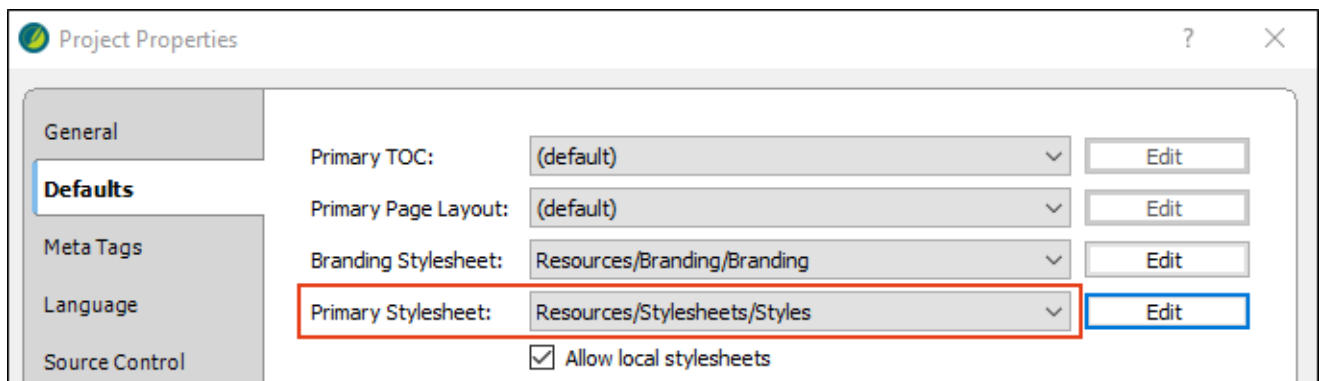
📄 **NOTE** Normally you would apply a condition to a piece of content or a file. But in Flare you can also set conditions on styles and then apply those styles to content. This is simply another alternative and might be more efficient for some authors. You might even find that you use both methods in your projects.

# I Associating Primary Stylesheets With All Files


When you want to use styles in your content, the stylesheet needs to be made available for the content in question. In Flare, you can associate regular stylesheets with individual files (see "Associating Stylesheets Locally With Specific Files" on page 153). However, you also have the option of using a regular stylesheet as the primary one, applying it at either the project or target level, or both.

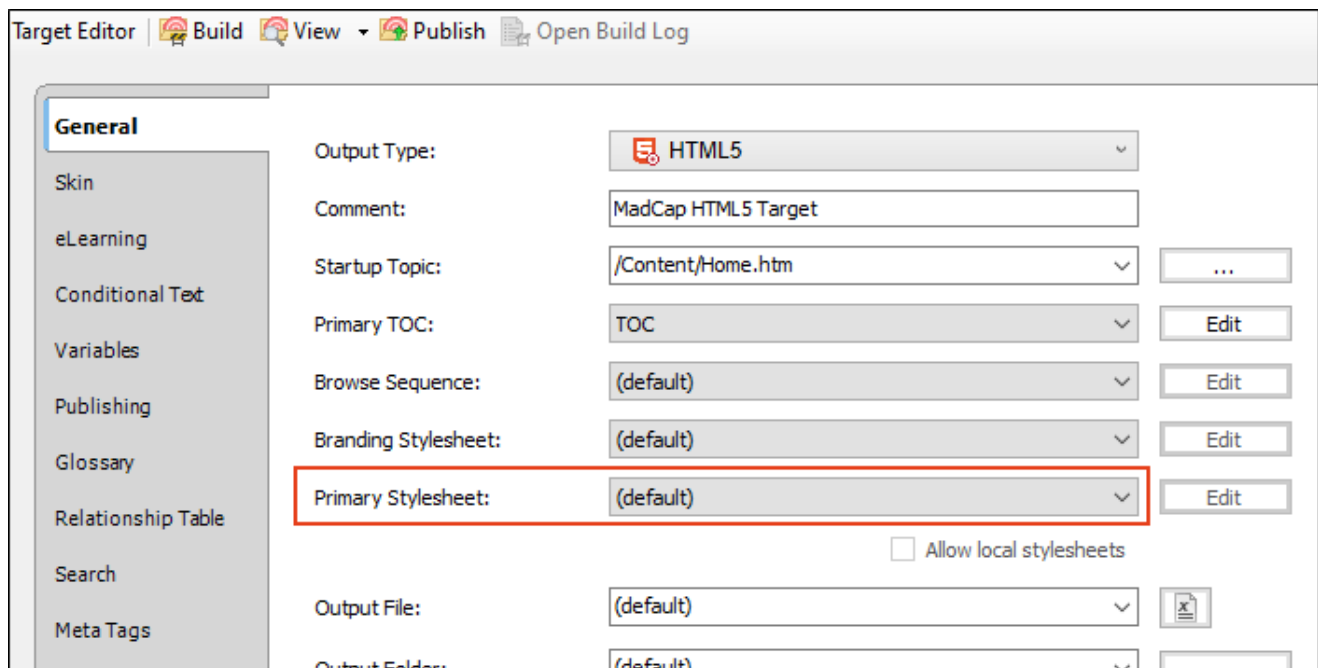
## How to Set a Primary Stylesheet at the Project level

1. In the **Project** ribbon, select **Project Properties**.
2. Select the **Defaults** tab.
3. In the **Primary Stylesheet** field, select the stylesheet.
4. (Optional) If you have other stylesheets and you want to associate them locally with topics, select **Allow local stylesheets**.
5. Click **OK**.



# How to Set a Primary Stylesheet at the Target level

1. Open the target and select the **General** tab.
2. In the **Primary Stylesheet** field, select the stylesheet.
3. (Optional) If you have other stylesheets and you want to associate them locally with topics, select **Allow local stylesheets**.
4. Click  to save your work.




# I Associating Stylesheets Locally With Specific Files


After creating a regular stylesheet, you need to associate (link) the stylesheet with files (e.g., topics, template pages, snippets, micro content) where you want to use those styles. You can associate a primary stylesheet at the project or target level, which means that the stylesheet will automatically be applied to all topics and micro content in that project or target (see "Associating Primary Stylesheets With All Files" on page 151). Alternatively, you can associate stylesheets with specific content files. This is the method to use when you have multiple stylesheets for a particular output. You can even associate multiple stylesheets with a single file.


The following steps show how to associate stylesheets with topics, template pages, or snippets. For steps on associating a stylesheet with a micro content file, see the online Help or the *Micro Content Guide*.


## How to Set a Stylesheet Locally on a Single File

1. Open the content file (e.g., topic, template page, snippet) to which you want to associate the stylesheet.
2. Select **Home > Stylesheet Links**. The Stylesheet Links dialog opens, showing all the regular stylesheets in your project.
3. Double-click the stylesheet(s) that you want to associate with the topic. The stylesheet is added to the Current Links section on the right.


 **NOTE** For topics, you can alternatively right-click the topic file in the Content Explorer, select **Properties**, and on the **Topic Properties** tab use the **Stylesheet** field to choose the stylesheet. You can also select the option **Disable project and target stylesheets** if you do not want to use any primary stylesheets along with the locally set stylesheet.


 **NOTE** If you associate multiple stylesheets with the content file, the last one you selected is the most recent one (the one on the bottom of the list) and therefore has precedence over the others. However, you can use the up and down arrows to change the order of the stylesheets.


 **NOTE** If the stylesheet does not yet exist, you can click **Add** to create a new stylesheet.

4. Click **OK**. The stylesheet is now associated with the file, and the look of the content file changes in the XML Editor accordingly.
5. Click  to save your work.

# How to Set a Stylesheet Locally on Multiple Files

1. Select **View > File List**, or press **CTRL+SHIFT+J**. The File List window pane opens.
2. (Optional) From the **Filter** drop-down list in the local toolbar, you can select topic, template page, or snippet files to limit the results in the grid.
3. Select the files to which you want to apply a stylesheet. You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items. However, if you select multiple files, they must be of the same type (e.g., only topic files, only template pages).
4. In the local toolbar, click .
5. In the Properties dialog, do one of the following, depending on whether you are associating the stylesheet with topics, template pages, or snippets:
  - **Topics** Click the **Topic Properties** tab. Then in the **Stylesheet** field, click **Select**. The Stylesheet Links dialog opens.
  - **Template Pages or Snippets** Click the **Stylesheet Links** tab.
6. Double-click the stylesheet(s) that you want to associate with the topic. The stylesheet is added to the Current Links section on the right.


 **NOTE** If you associate multiple stylesheets with the content file, the last one you selected is the most recent one (the one on the bottom of the list) and therefore has precedence over the others. However, you can use the up and down arrows to change the order of the stylesheets.

 **NOTE** If the stylesheet does not yet exist, you can click **Add** to create a new stylesheet.

7. Click **OK**. If you are working with template pages or snippets, this is the last step.
8. (Optional for Topics) If you have a primary stylesheet at the project or target level and no longer want to use it, you can select **Disable project and target stylesheets**. For more about how primary and local stylesheets work in the same project, see "Primary and Local Stylesheets (and Precedence)" on page 86.
9. Click **OK**.

# What's Noteworthy?

 **NOTE** You cannot use multiple stylesheets per content file for native Adobe PDF output.

 **NOTE** Although it is possible to associate a stylesheet locally with a snippet, the only reason to do this is if you do not have any primary stylesheets in your project. Without a primary stylesheet, a snippet's content will look very plain when you open it. That's because Flare doesn't know which styles to use for it. In order to work in that snippet and apply styles to the content, you will need to associate the snippet with a stylesheet.

# What's Next?

After you associate a stylesheet with a file, you can apply styles from the stylesheet to content in the file. You can also create new styles, adding them to the stylesheet. See "Applying Styles to Content" on page 136 and "Creating Selectors" on page 106.



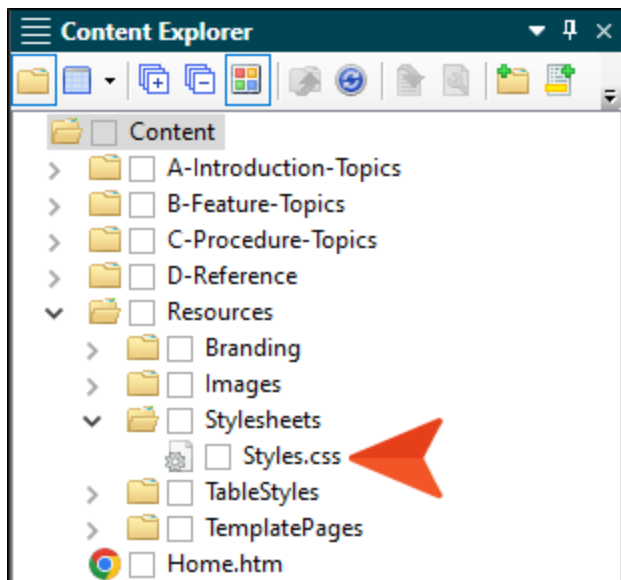
## CHAPTER 5

---

# Regular Stylesheets

A branding stylesheet specifically identifies values for branding elements. A regular stylesheet lets you store styles for general content in your project, including tables, to control how that content looks. A table stylesheet, is used *only* for tables.

You can have as many styles as you want within one regular stylesheet, and you can create as many stylesheets as you need (although one stylesheet is often sufficient for many authors and projects). The recommended location to store a regular CSS stylesheet in the Content Explorer is in the Resources > Stylesheets folder. However, you can store it anywhere in the Content Explorer that you like.



The exception to this is when you import source files that already include a stylesheet. In that case, Flare retains the structure of the imported files, storing the stylesheet in the same location where it resided in the source files.

This chapter discusses the following:

Stylesheets .....	159
Styles .....	168
Property Values .....	215
CSS Variables .....	218
Style Inspector .....	235
Style Reports .....	269

# Stylesheets

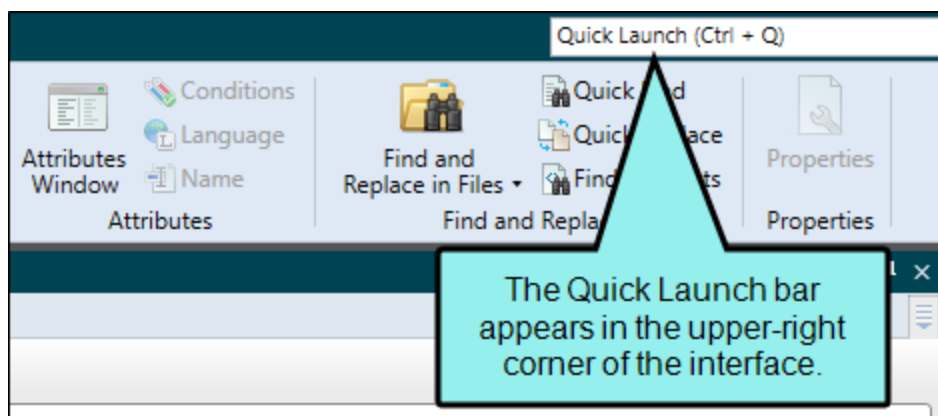
You can perform the following additional tasks with stylesheets:

## Opening Stylesheets

When you create a new stylesheet for your project, the stylesheet opens automatically in the Stylesheet Editor or Table Style Editor. If you want to make changes to a stylesheet that is closed, use the following steps to open it.

## How to Open a Stylesheet From the Quick Launch Bar


The Quick Launch bar lets you search for any Flare file or command. It is located in the upper-right corner of the interface. You can press **CTRL+Q** on your keyboard to move focus to the Quick Launch bar so you can begin typing.




1. In the Quick Launch bar, type a few letters of the name of the file you want to open. Any available results appear in a drop-down list.
2. From the list, click the name of the file.

# How to Open a Stylesheet From the Content Explorer

1. Open the Content Explorer.
2. Double-click the **Resources** folder to open it.
3. Do one of the following:
  - If you want to open a regular stylesheet, double-click the **Stylesheets** subfolder.
  - If you want to open a table stylesheet, double-click the **TableStyles** subfolder.
  - If you want to open a branding stylesheet, double-click the **Branding** subfolder.

 **NOTE** These are the traditional locations for storing stylesheet files. However, you can store them in custom folders elsewhere in the Content Explorer. Also, if you imported a project that has a stylesheet, it will not be located in one of these folders unless you move it there. Instead, the stylesheet will be located in the same place where it was stored in the source project (e.g., at the root level of the Content Explorer).




4. Do one of the following:
  - Locate and double-click the stylesheet (CSS file) that you want to open.
  - Locate and click the stylesheet (CSS file) that you want to open. In the local toolbar, click .

The stylesheet opens in the Stylesheet Editor, Table Style Editor, or Branding Editor, depending on which one you are opening.

# Importing Stylesheets

If you already have a stylesheet (CSS file) somewhere else and want to reuse it in your current project, you can import that stylesheet.

## How to Import a Stylesheet

1. Select **Project > New > Stylesheet**. The Add File dialog opens.
2. Select **New from existing** and click .
3. Find and select the stylesheet file that you want to import.
4. Click **Open**. The Source File field now contains the path to the file that you are importing. Also, the name of the file is displayed in the File Name field.
5. In the **Folder** field, you can leave it as **(root folder)**, or you can choose a folder for the stylesheet. Depending on the type of stylesheet, you might select Resources > Stylesheets or Resources > Branding.
6. If you want to give the stylesheet a different name than that for the imported file, click in the **File name** field and replace the text.
7. (Optional) If you want to apply condition tags to the file, expand the **Attributes** section at the bottom of the dialog. Next to the **Condition Tags** field, click  and select the conditions you want to apply. Click **OK**.
8. (Optional) If you want to apply file tags, expand the **Attributes** section at the bottom of the dialog. Next to the **File Tags** field, click  and select the file tags you want to apply. Click **OK**.
9. Click **Add**. The stylesheet is added to the Content Explorer and opens in an editor (i.e., the Stylesheet Editor or the Branding Editor).



**NOTE** You can also import individual styles from another stylesheet. See "Importing Styles" on page 168.



# Linking Stylesheets

If you have more than one stylesheet in your project, you can link them together. By doing this, one stylesheet can adopt the styles of the other stylesheet so that they can be used in content files where that stylesheet is applied.

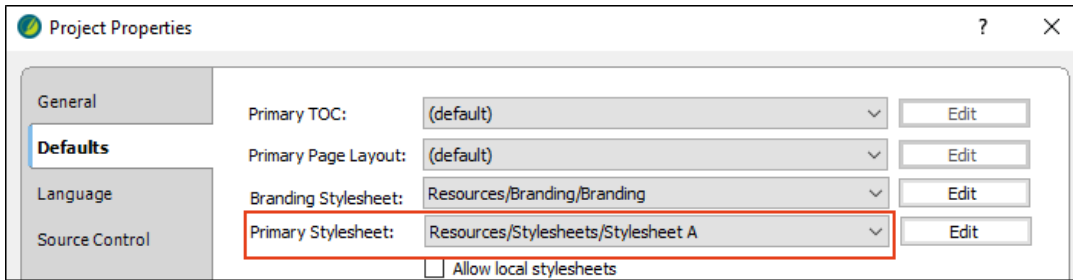
If the linked stylesheets share the same style and have conflicting settings, the stylesheet that is most closely associated with the content file has precedence (e.g., the stylesheet is the most recent one associated with the topic, or the stylesheet is set as the primary at the project or target level).

However, if there are shared styles between the linked stylesheets but the primary stylesheet (which would normally have precedence) does not have a value explicitly set for a certain property while the other stylesheet does, that explicitly set value will be seen. Therefore, you should use caution with this feature.

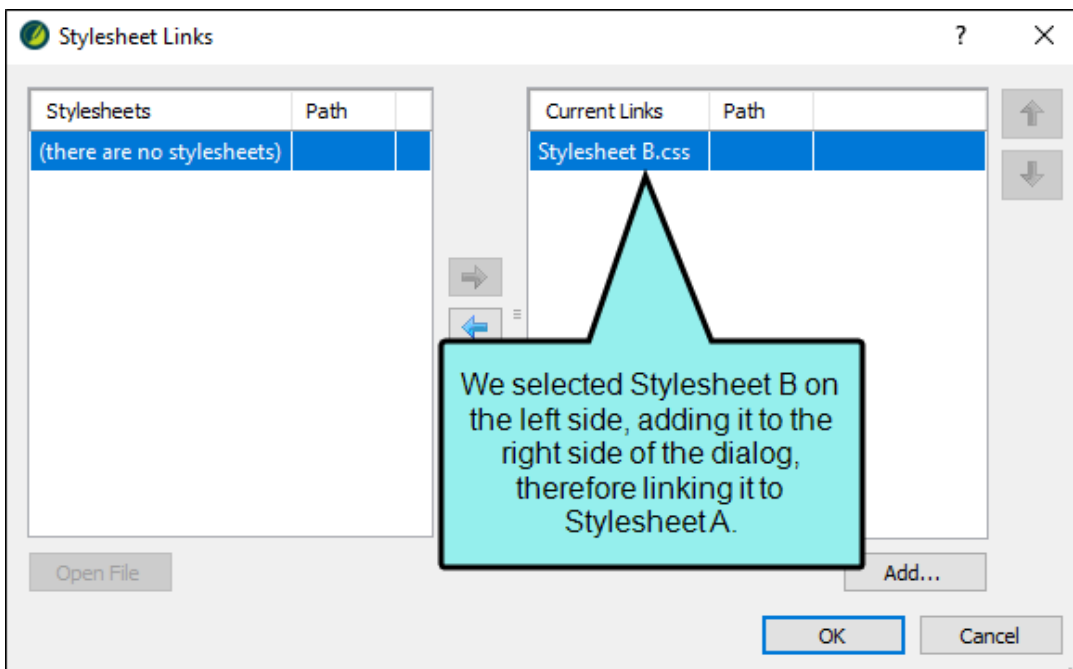
## How to Link Stylesheets

1. Open a stylesheet that you want to link to another stylesheet. The stylesheet being opened will adopt the styles from the stylesheet(s) that you link it to.
2. In the local toolbar of the Stylesheet Editor, click the **Options** button and select **Stylesheet Links**. The Stylesheet Links dialog opens.
3. Do one of the following:
  - On the left side of the dialog, select the stylesheet(s) that you want to link to the current stylesheet. Then click  to add the stylesheet(s) to the Current Links section on the right.
  - Double-click the stylesheet(s) that you want to link to the current stylesheet. The stylesheet is added to the Current Links section on the right.
4. Click **OK**.
5. Click  to save your work.

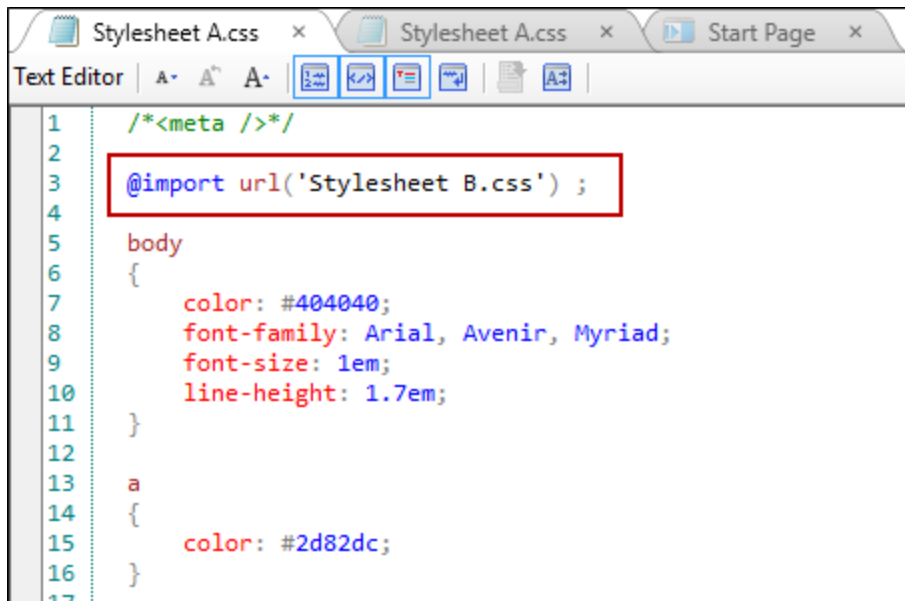
☆ **EXAMPLE** You have two stylesheets: Stylesheet A and Stylesheet B. You select Stylesheet A in the Project Properties dialog, making it the primary stylesheet for the entire project.



After opening Stylesheet A in the Stylesheet Editor, you link it to Stylesheet B.



- ☆ If you open Stylesheet A in the Internal Text Editor, you would see this at the top of it, indicating the link:



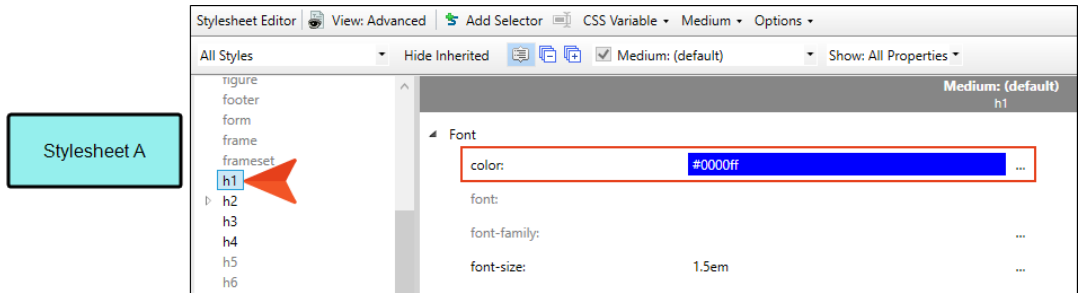
The screenshot shows a text editor window with three tabs: 'Stylesheet A.css', 'Stylesheet A.css', and 'Start Page'. The editor displays the following CSS code:

```
1  /*<meta />*/
2
3  @import url('Stylesheet B.css') ;
4
5  body
6  {
7      color: #404040;
8      font-family: Arial, Avenir, Myriad;
9      font-size: 1em;
10     line-height: 1.7em;
11 }
12
13 a
14 {
15     color: #2d82dc;
16 }
17
```

The line `@import url('Stylesheet B.css') ;` is highlighted with a red rectangular box.



- ☆ Each stylesheet has the h1 style. In Stylesheet A you've made the font blue, but in Stylesheet B you've made it red. Because Stylesheet A is the primary stylesheet for the project, the XML Editor and the output would display h1 content in blue.

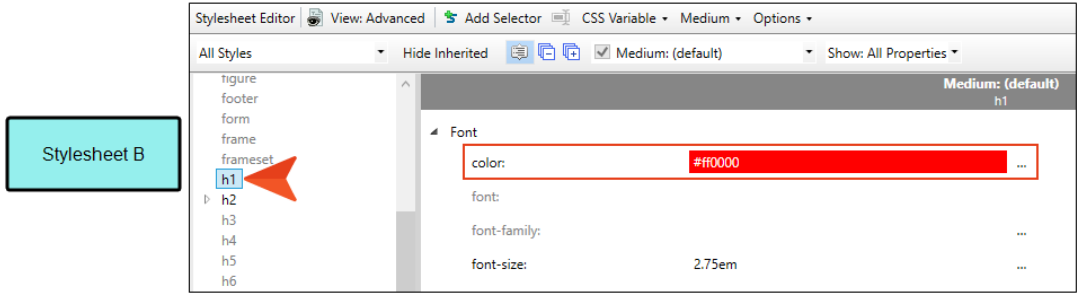


Stylesheet Editor View: Advanced Add Selector CSS Variable Medium Options

All Styles Hide Inherited Medium: (default) Show: All Properties

figure footer form frame frameset h1 h2 h3 h4 h5 h6

Font color: #0000ff font: font-family: font-size: 1.5em



Stylesheet Editor View: Advanced Add Selector CSS Variable Medium Options

All Styles Hide Inherited Medium: (default) Show: All Properties

figure footer form frame frameset h1 h2 h3 h4 h5 h6

Font color: #ff0000 font: font-family: font-size: 2.75em

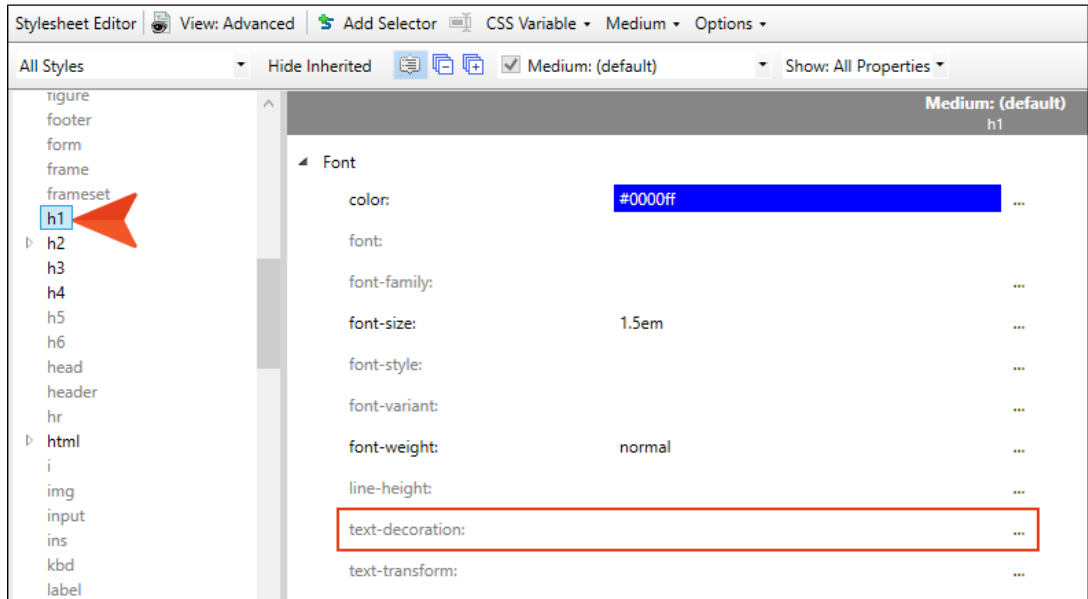
Output

Getting Started

See the following to get started quickly:

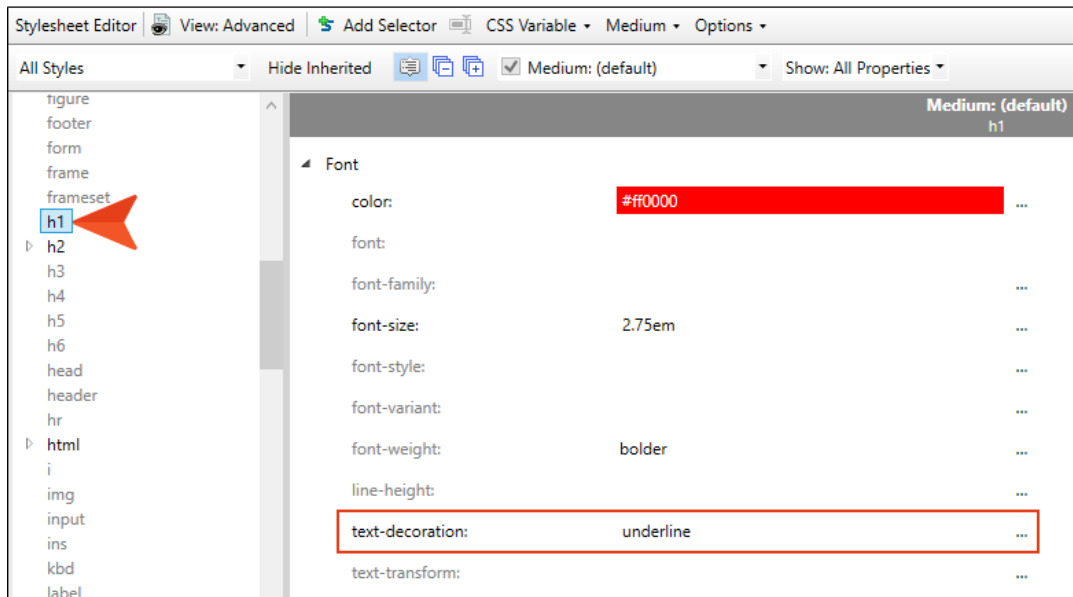
- ☆ If you look at the h1 properties in the Stylesheet Editor for Stylesheet A, you will notice that the text-decoration property is not set; it is using the default value.

Stylesheet A



☆ Let's say you open Stylesheet B and set the text-decoration for h1 to underline.


Stylesheet B



Even though Stylesheet A is the primary stylesheet, it does not have that value explicitly set, so Flare will use the value from Stylesheet B. As a result, in the output you will see h1 text that is blue (from Stylesheet A) and underlined (from Stylesheet B).

## Getting Started

See the following to get started quickly:

 **NOTE** If you have created a link from one stylesheet to another, you can double-click on an inherited property to open that other stylesheet. See "Editing Styles in a Regular Stylesheet" on page 111.

# I Styles

You can perform the following additional tasks with styles:

## Importing Styles

You can manually create new style classes in a stylesheet. Another option is to import existing styles from another stylesheet.

## How to Import a Style

1. From the Content Explorer, open the stylesheet that you want to modify.
2. In the local toolbar, click the **Options** button and select **Import Styles**.
3. Do one of the following, depending on the stylesheet containing the style you want to import.

### TO SELECT A STYLE SHEET FROM ONE OF THE AVAILABLE FOLDERS

- a. In the **Library Folders** section, select one of the folders.
  - **Factory Stylesheets** Holds stylesheets that are provided by Flare. This folder includes a stylesheet called "SearchHighlight," which provides styles that let you control the look of terms that are highlighted in searches performed by users.
  - **My Templates** Holds your own stylesheets that you store in your Documents\My Templates\Stylesheets folder.
  - **Project Stylesheets** Holds stylesheets added to a project.
- b. In the **Styles** section to the right, select a stylesheet contained in the folder.

### TO SELECT A STYLE SHEET NOT FOUND IN ONE OF THE AVAILABLE FOLDERS


- a. Click the **Browse** button.
  - b. In the dialog, find and double-click the stylesheet.
4. (Optional) In the **[Show Styles]** drop-down list, you can make a selection to filter which types of styles to show in the area below.

5. (Optional) In the **[Medium]** drop-down list, you can select a specific medium. This determines the medium to which the styles are imported in your current stylesheet. If you select "default," the imported style properties will be applied to all of the mediums in the other stylesheet. If you select a custom medium, the imported style properties will be imported to that medium in the other stylesheet. For more information see "Mediums and Media Queries" on page 298.
6. Click the **Import** check box next to each style that you want to import.
7. Click **OK**. The styles are added to the current stylesheet.

# Renaming Selectors

You can rename a selector (e.g., class, ID) after you have created it. However, you cannot rename existing parent styles, such as p, h1, or span. When you rename a style class or ID, you have the option to automatically rename all instances of that class or ID in the stylesheet accordingly.

## How to Rename a Selector

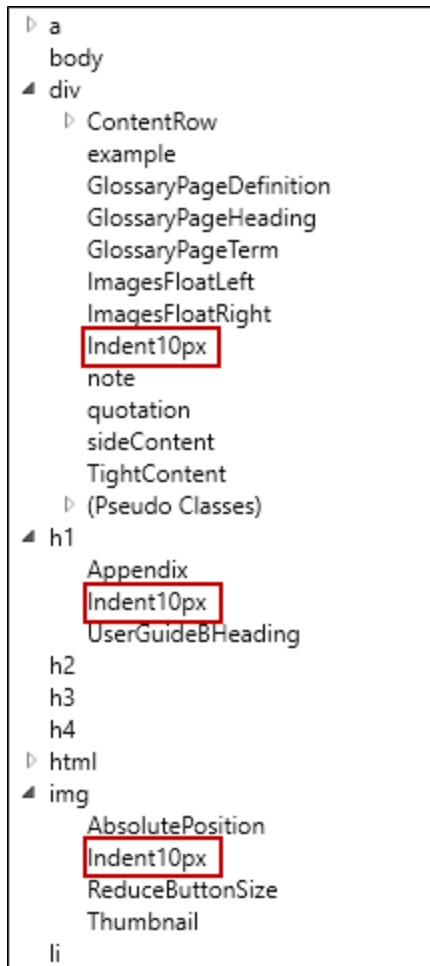
1. From the Content Explorer, open the stylesheet that you want to modify.
2. In the Stylesheet Editor, choose the selector that you have created (not a parent style).
3. Do one of the following, depending on the part of the user interface you are using:
  - **Local Toolbar** In the local toolbar, click  **Rename**.
  - **Right-Click** After right-clicking the style class or ID, choose **Rename**.
  - **Keyboard Shortcut** Press **F2**.

The Rename Class dialog opens.

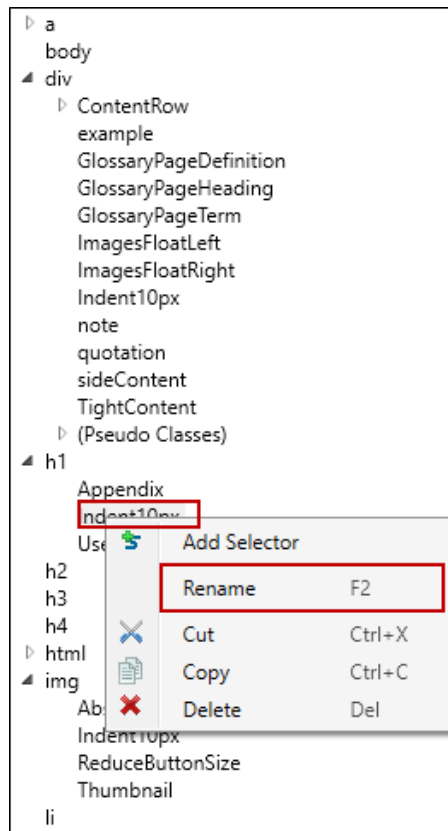
4. Enter a new name for the selector.

- (Optional) If you want to rename all instances of that class or ID in the stylesheet, select **Rename all instances**.

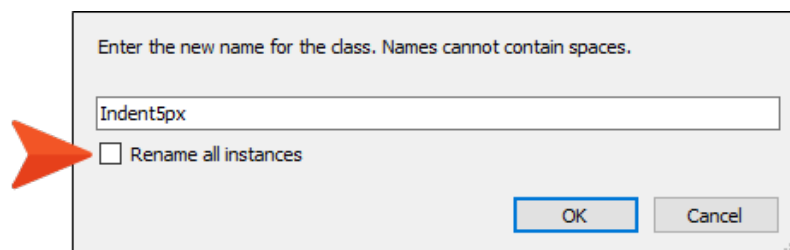
☆ **EXAMPLE** You have created a class named "Indent10px" under three styles—div, h1, and img.



☆ You decide to rename the class under the h1 style. So you right-click the class and select **Rename**.

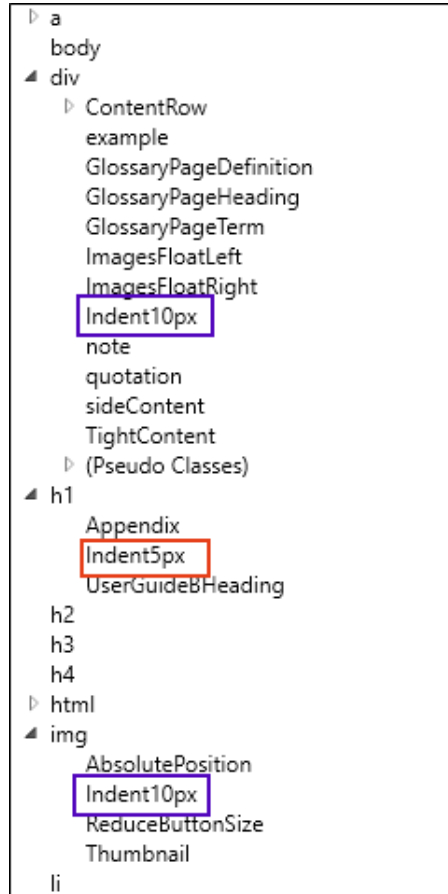


In the dialog that opens, you rename it `Indent5px`. However, you leave the option **Rename all instances** deselected.

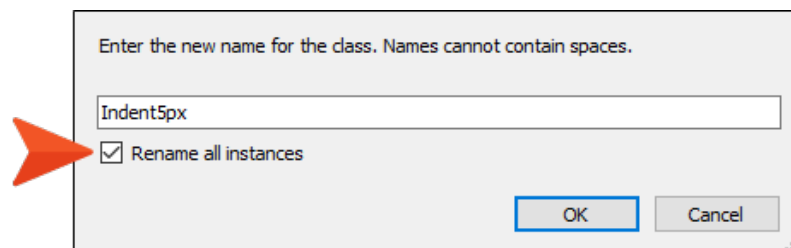




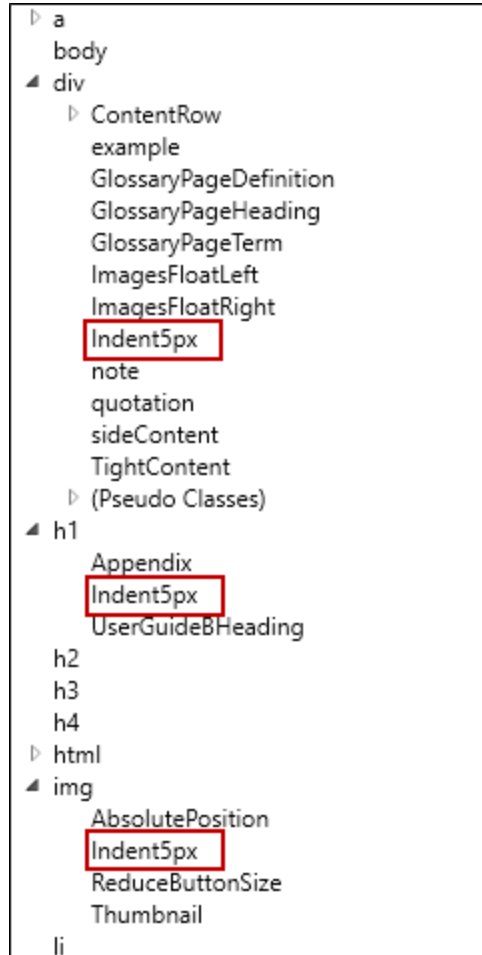
☆ As a result, only the class under h1 is renamed.



But let's say instead you rename the class and enable the **Rename all instances** option.




☆ As a result, all three of the classes are renamed.



6. Click OK.

7. Click  to save your work.

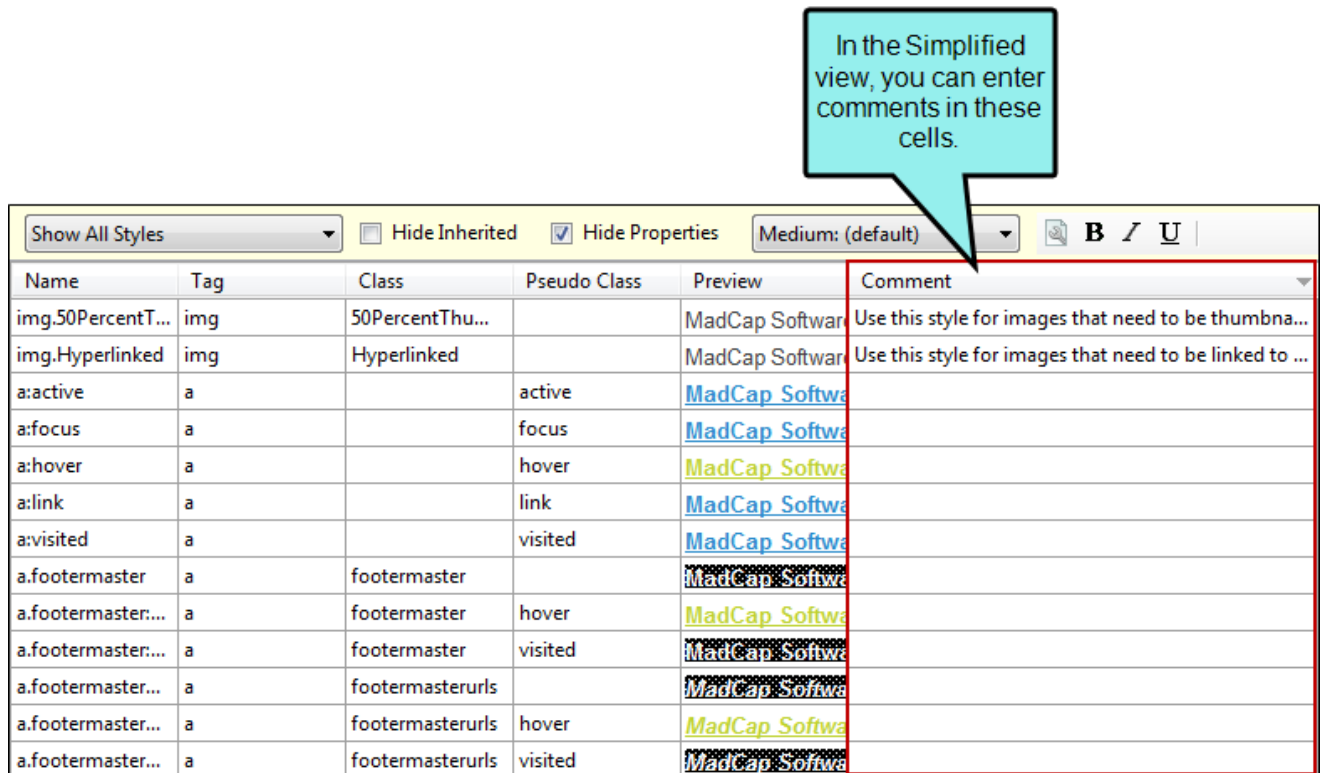
 **IMPORTANT** Renaming a selector in the stylesheet will *not* automatically update any instances throughout your project where you have previously applied that class or ID to content. If you want to update areas of content where the selector has been applied, you can use the Find and Replace window pane; use the **Find in source code** option to quickly replace all instances of the old name with the new name. However, be careful when using this method for a global find and replace so that you do not accidentally introduce invalid code or replace the wrong text. It is recommended that you have a backup of your project before you perform a global find and replace such as this.

# Adding Comments to Styles

If you are familiar with using cascading stylesheets (CSS) in a text editor, you probably already know that you can add comments to styles. This is simply a way to remind you or others about information related to a style (e.g., which situations are appropriate to use a certain style). In Flare you can add, edit, and view these style comments through the user interface. This can be done in both views of the Stylesheet Editor, the Internal Text Editor, the Create Style dialog, the New Selector dialog, and the Style Inspector.

## Simplified View of the Stylesheet Editor

In the Simplified view of the Stylesheet Editor, you can click twice (or click once and press F2) in the **Comment** cell and type text related to the style.

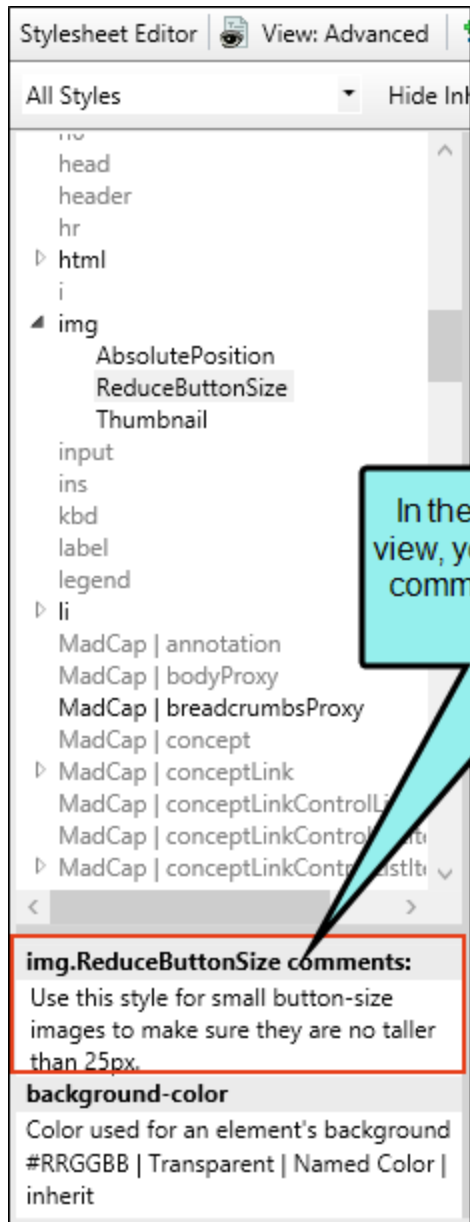


In the Simplified view, you can enter comments in these cells.

Name	Tag	Class	Pseudo Class	Preview	Comment
img.50PercentT...	img	50PercentThu...		MadCap Softwar	Use this style for images that need to be thumbna...
img.Hyperlinked	img	Hyperlinked		MadCap Softwar	Use this style for images that need to be linked to ...
a:active	a		active	MadCap Softwa	
a:focus	a		focus	MadCap Softwa	
a:hover	a		hover	MadCap Softwa	
a:link	a		link	MadCap Softwa	
a:visited	a		visited	MadCap Softwa	
a.footermaster	a	footermaster		MadCap Softwa	
a.footermaster...	a	footermaster	hover	MadCap Softwa	
a.footermaster...	a	footermaster	visited	MadCap Softwa	
a.footermaster...	a	footermasterurls		MadCap Softwa	
a.footermaster...	a	footermasterurls	hover	MadCap Softwa	
a.footermaster...	a	footermasterurls	visited	MadCap Softwa	

# Advanced View of the Stylesheet Editor

In the Advanced view of the Stylesheet Editor, you can select the style and then type a comment in the field at the bottom of the **Styles** section.



In the Advanced view, you can enter comments in this field.

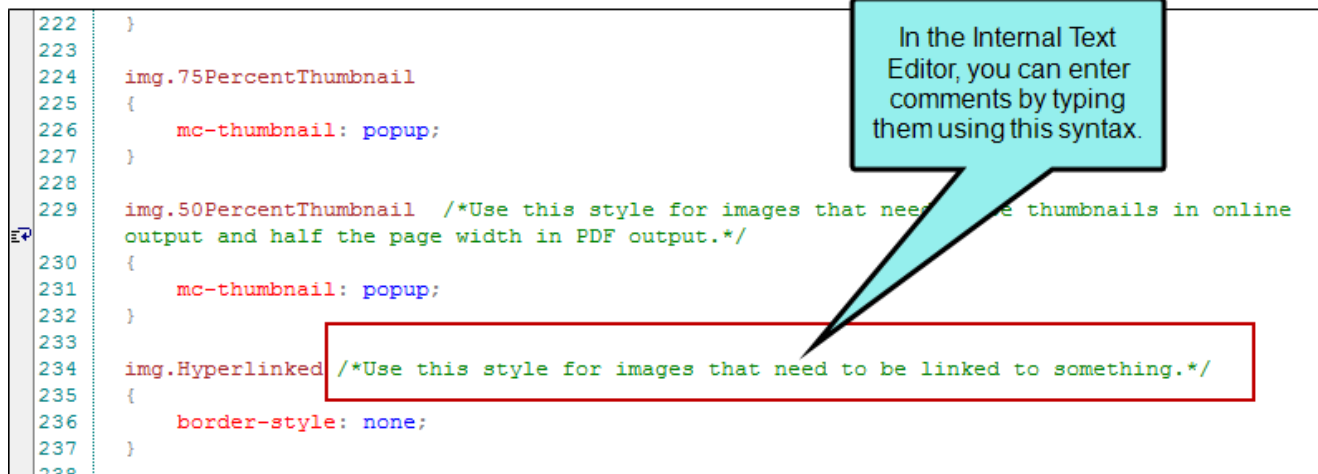
# Internal Text Editor

In the Internal Text Editor, you can type your comment after a style name, using the following syntax.

```
/*[comment text]*/
```

After typing your comment using this format, it should display in a green font to signify that it is a comment.

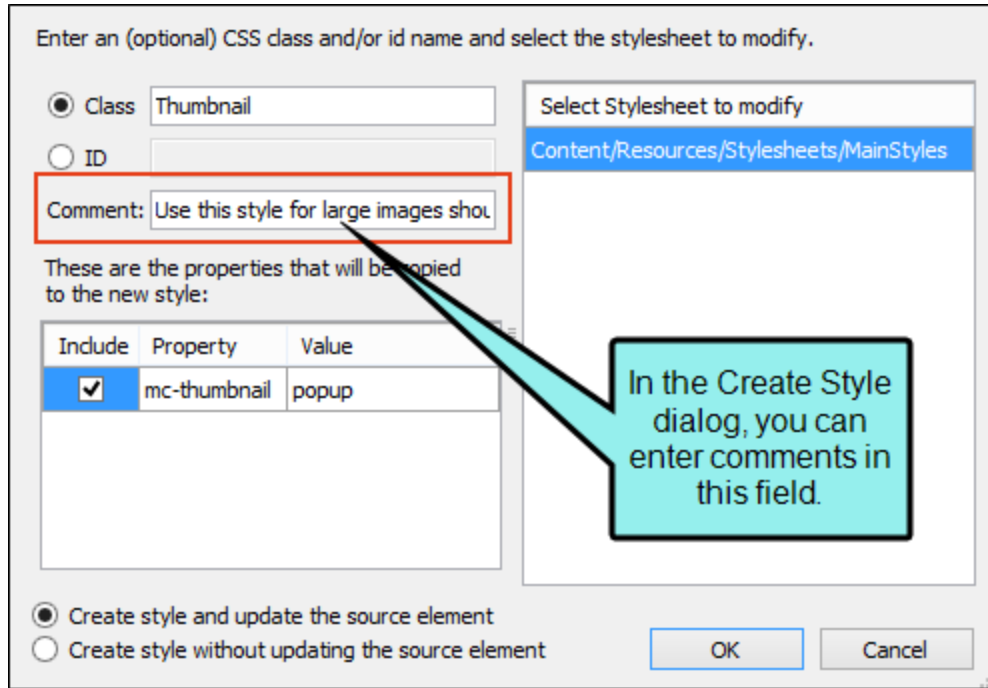
```
222     }
223
224     img.75PercentThumbnail
225     {
226         mc-thumbnail: popup;
227     }
228
229     img.50PercentThumbnail /*Use this style for images that need to be thumbnails in online
output and half the page width in PDF output.*/
230     {
231         mc-thumbnail: popup;
232     }
233
234     img.Hyperlinked /*Use this style for images that need to be linked to something.*/
235     {
236         border-style: none;
237     }
238
```



In the Internal Text Editor, you can enter comments by typing them using this syntax.

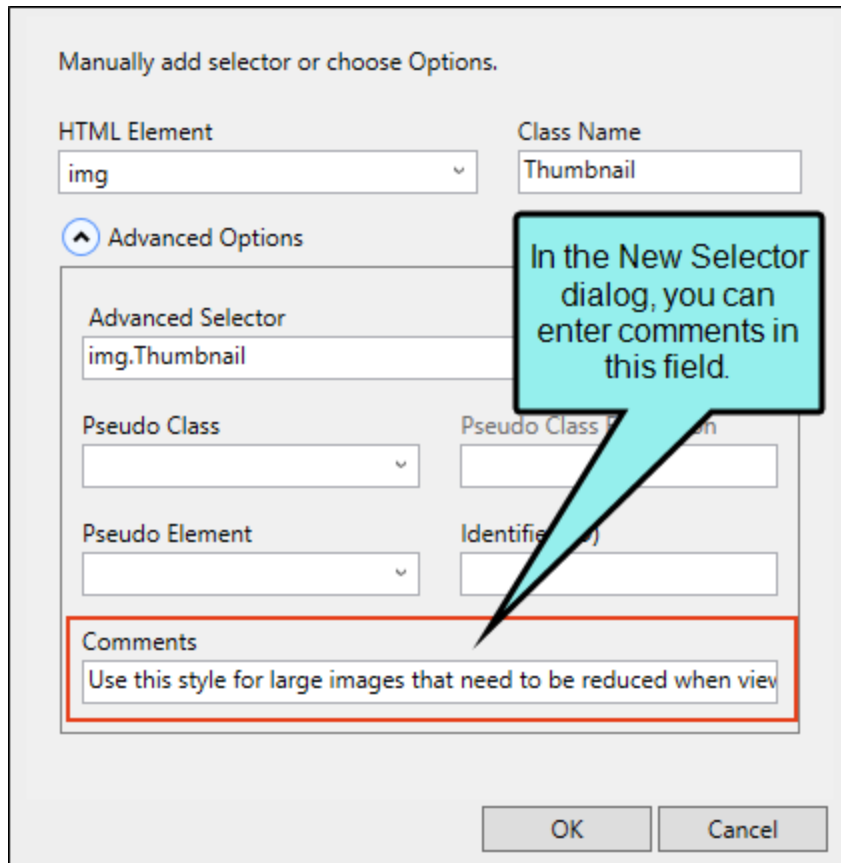
# Create Style Dialog

When you open the Create Style dialog, you can enter text in the **Comment** field.



# New Selector Dialog

When you open the New Selector dialog, you can enter text in the **Comments** field.



Manually add selector or choose Options.

HTML Element:  Class Name:

Advanced Options

Advanced Selector:

Pseudo Class:  Pseudo Class Extension:

Pseudo Element:  Identifier:

Comments:

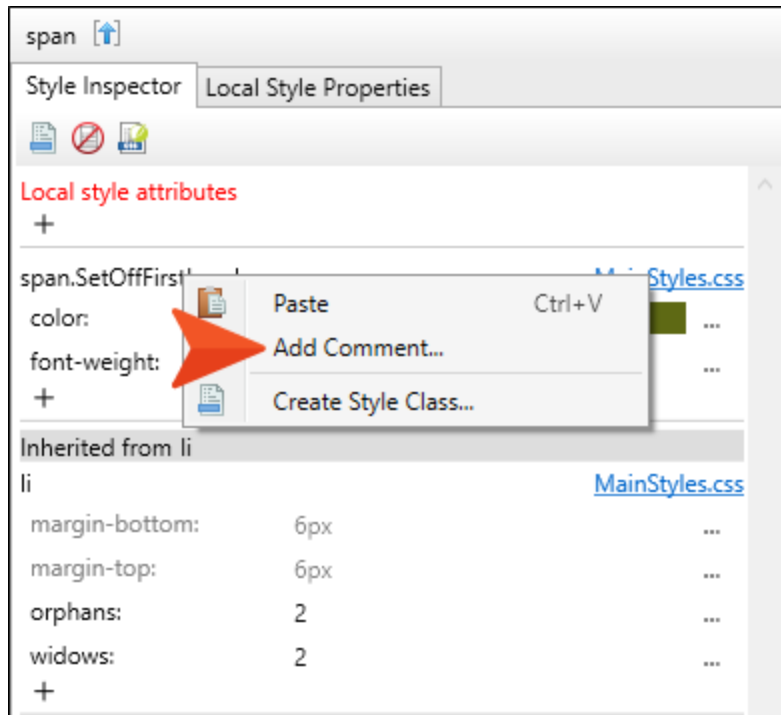
OK Cancel

In the New Selector dialog, you can enter comments in this field.

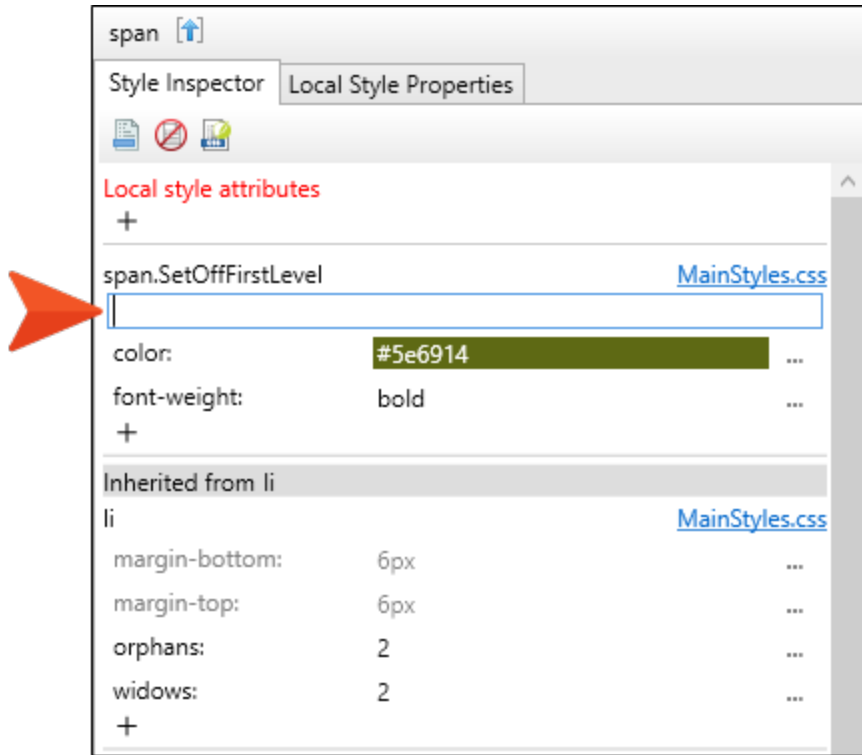


# Style Inspector

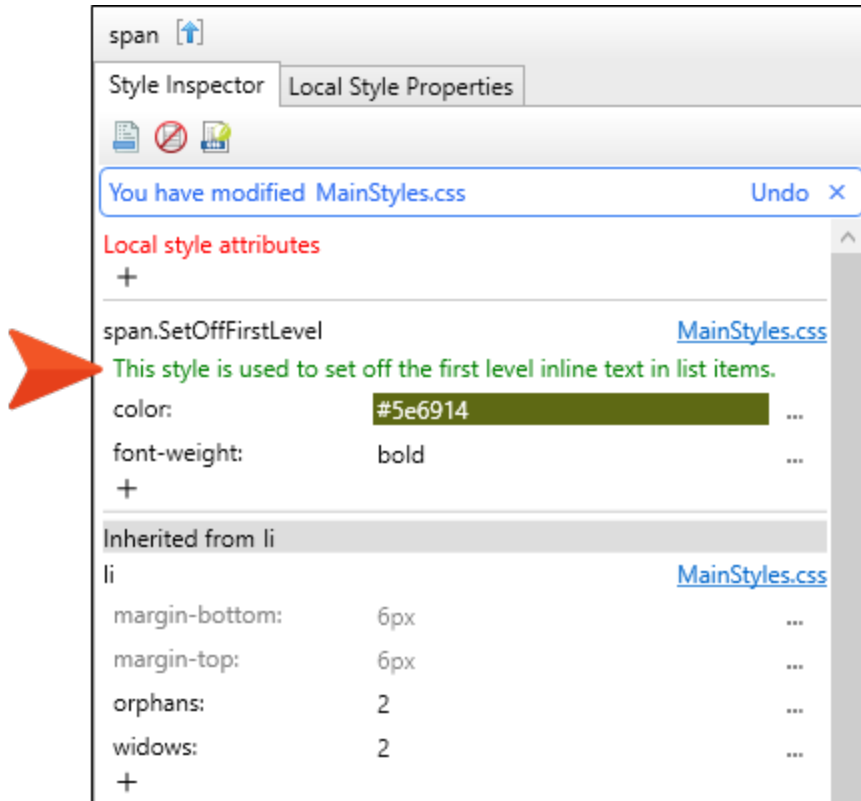
In the Style Inspector, start by right-clicking a style, then select **Add Comment**.



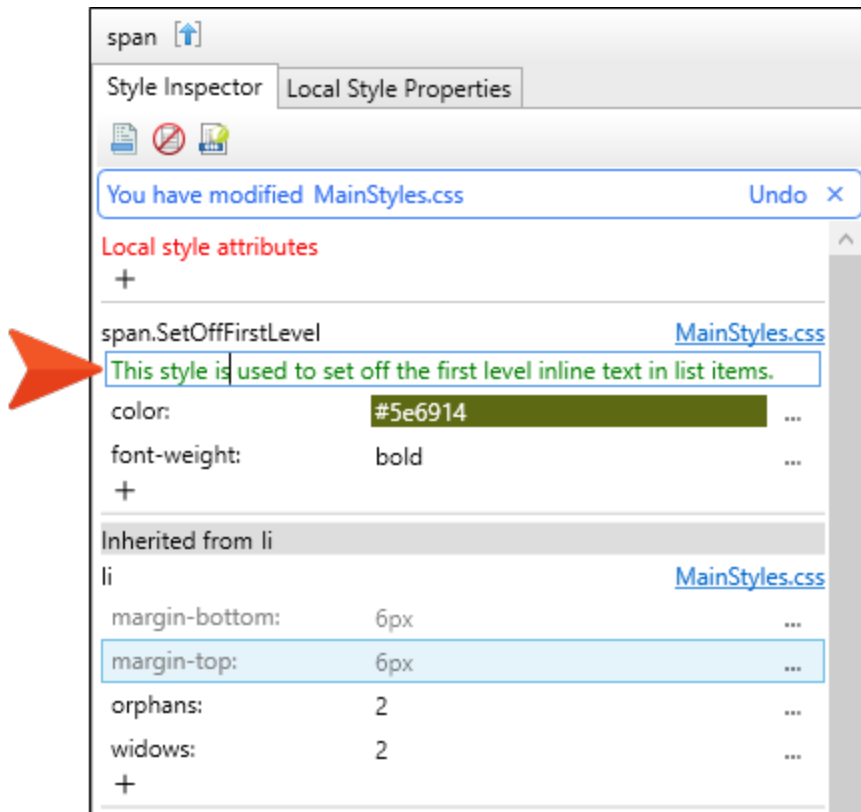
A text box displays below the style, where you can type the comment.




After you press ENTER, the comment appears in green text.




If you want to edit the comment, simply click on the comment and the text box will appear, allowing you to type in it.



## What's Noteworthy?

 **NOTE** When you add or edit a comment using any of these methods, the comment is generated and displayed in the other views. For example, if you add a comment to a style in the Simplified view of the Stylesheet Editor and then open the CSS file in the Internal Text Editor, you will see that same comment.

 **NOTE** If you add or edit a style comment in the Stylesheet Editor when the (default) medium is selected, the same comment is shown in the editor when you switch to the other mediums. If you add or edit a comment when one of the other mediums is selected, that comment displays only when that specific medium is selected in the editor.

# Setting a "Next" Style

You can specify that a particular style should be used when you press ENTER at the end of the current style. For example, after you type text for a heading and press ENTER, you might want the next style to be something like p.TopicText, rather than the main p style.

In this version of Flare, you cannot specify this setting in the user interface. Instead, you need to open the stylesheet in the Internal Text Editor, or another editor such as Notepad, and enter the settings manually.

## How to Set a "Next" Style

1. In the Content Explorer, right-click on the stylesheet, and from the context menu select **Open with > Internal Text Editor** or **Open with > Notepad**.
2. Find the "current" style (i.e., the style that will immediately precede the "next" style).
3. Within the curly brackets in the CSS file, enter the following text if the next style is a primary style (e.g., p, li).

```
mc-next-tag: [tag];
```

OR

Within the curly brackets, enter the following text if the next style is a class.

```
mc-next-tag: [tag];
```


```
mc-next-class: [class];
```

4. Save your changes.

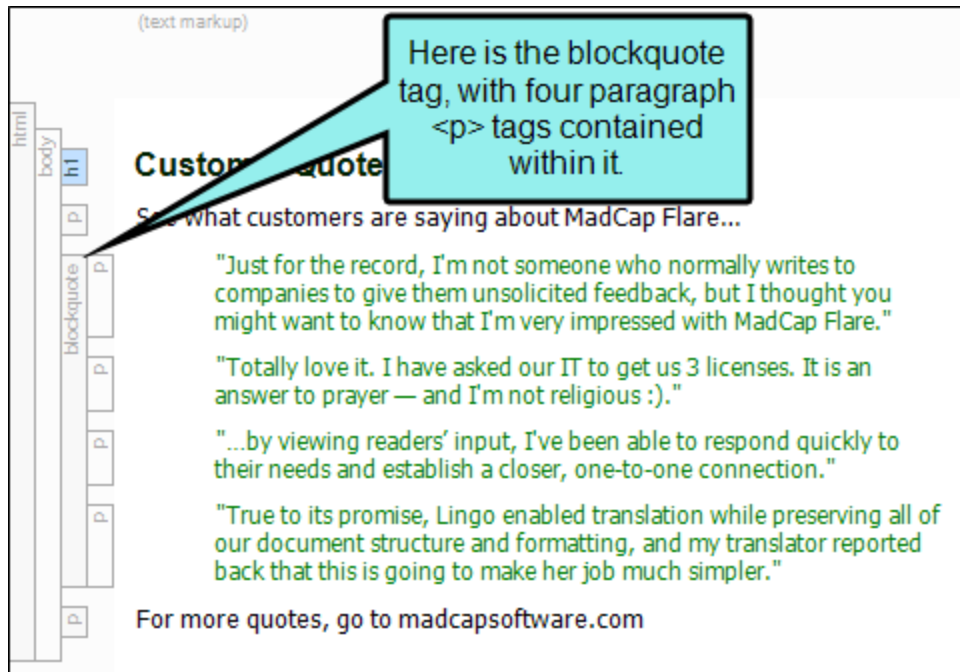
# Creating Divs and Other Tag Groups

In your content in the XML Editor, you can group selected items within one of the following types of block styles: blockquote, div, fieldset, or form. These styles let you create block-level content in a unique "container" for different purposes.

# How to Create a Tag Group

1. In the XML Editor, select one or more blocks of content.
2. In the **Home** ribbon click .
3. Select a tag from the list, then click **OK**.
  - **blockquote** The <blockquote> tag is typically used to format text used as a quotation. Usually the <blockquote> tag has margin indentations to set it apart from the rest of the content in the topic.

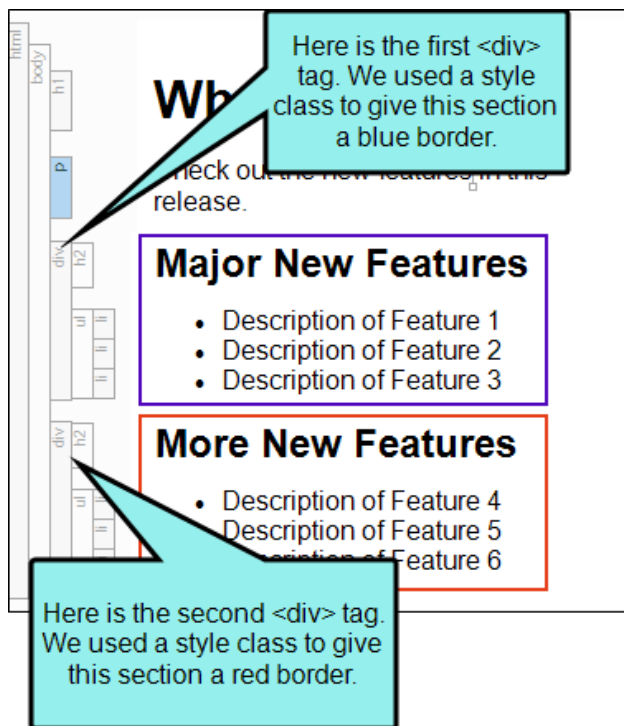
☆ **EXAMPLE** The following image shows four p tags grouped into a <blockquote> tag in the XML Editor. The structure bars on the left side of the content provide a visual representation of the grouping.





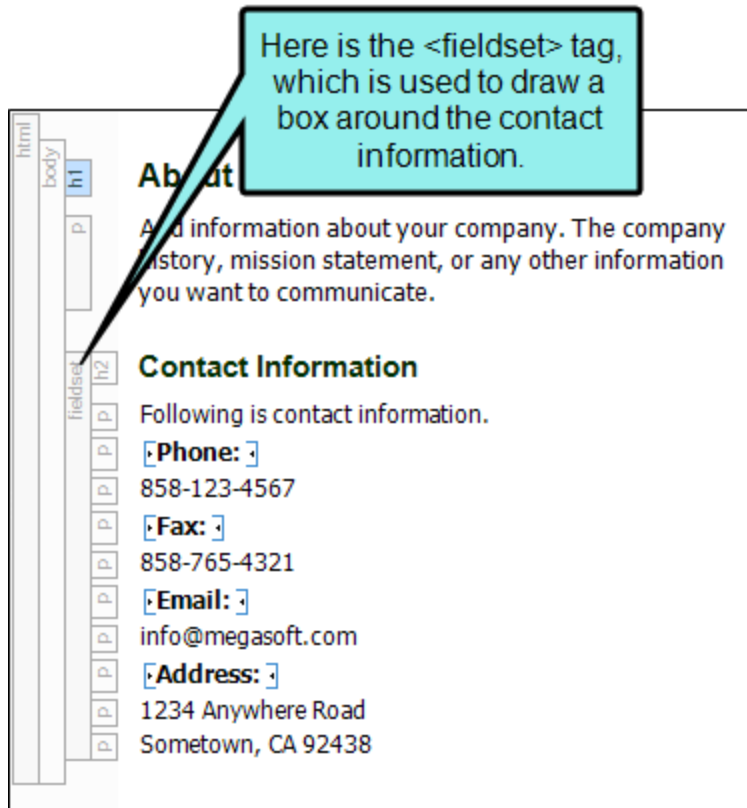
- **div** The `<div>` tag is used to define logical divisions in your topics or hold objects that can be “floated” (such as text boxes). You can put content using other tags into the same `<div>` tag, then use style properties to change the look of that entire “container.” One common use of a `<div>` tag is to indent lots of content. For example, you might have a section of content containing four paragraphs, a numbered list, and an image. Rather than creating special style classes for all of those different elements with an indentation setting (e.g., `margin-left` or `padding-left`) on each, you can place your indentation setting on the `<div>` tag. That way, any content contained within that `<div>` tag will be indented accordingly.

☆ **EXAMPLE** The following image shows two `<div>` tags. Each of these tags represents a logical grouping, containing a heading tag and an unordered list with multiple `li` tags. The structure bars on the left side of the content provide a visual representation of the grouping.



- **fieldset** The <fieldset> tag is used to combine multiple tags into a group, drawing a box around all of the content.

☆ **EXAMPLE** The following image shows a <fieldset> tag that contains a heading tag and several p tags. The structure bars on the left side of the content provide a visual representation of the grouping. To see the actual box around the content, you need to preview or generate the topic.





This is how it looks when you generate the output.

## About Fiction Soft

Add information about your company, company history, mission statement, or any other information you want to communicate.

## Contact Information

Following is contact information.

**Phone:**

858-123-4567

**Fax:**

858-765-4321

**Email:**

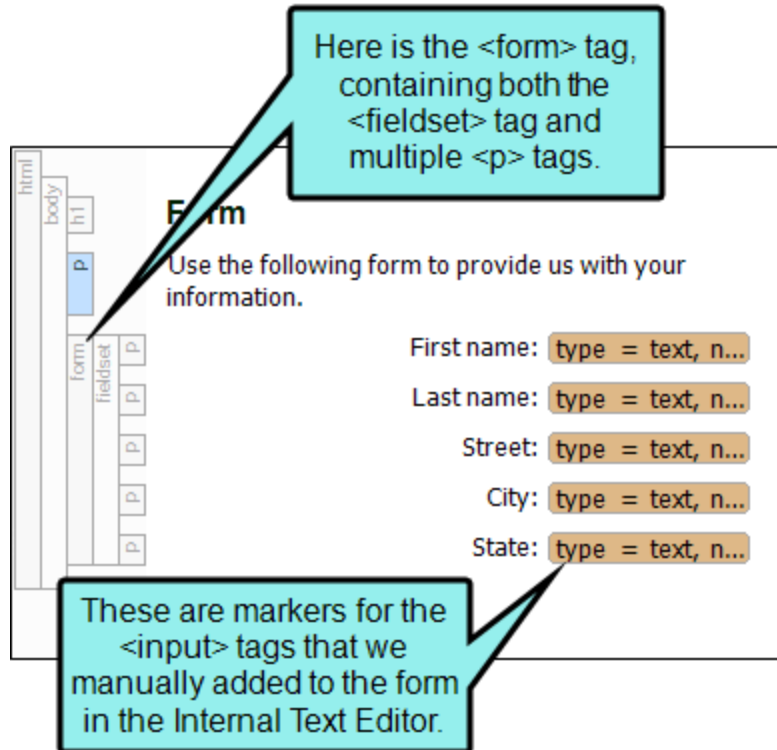
info@megasoft.com

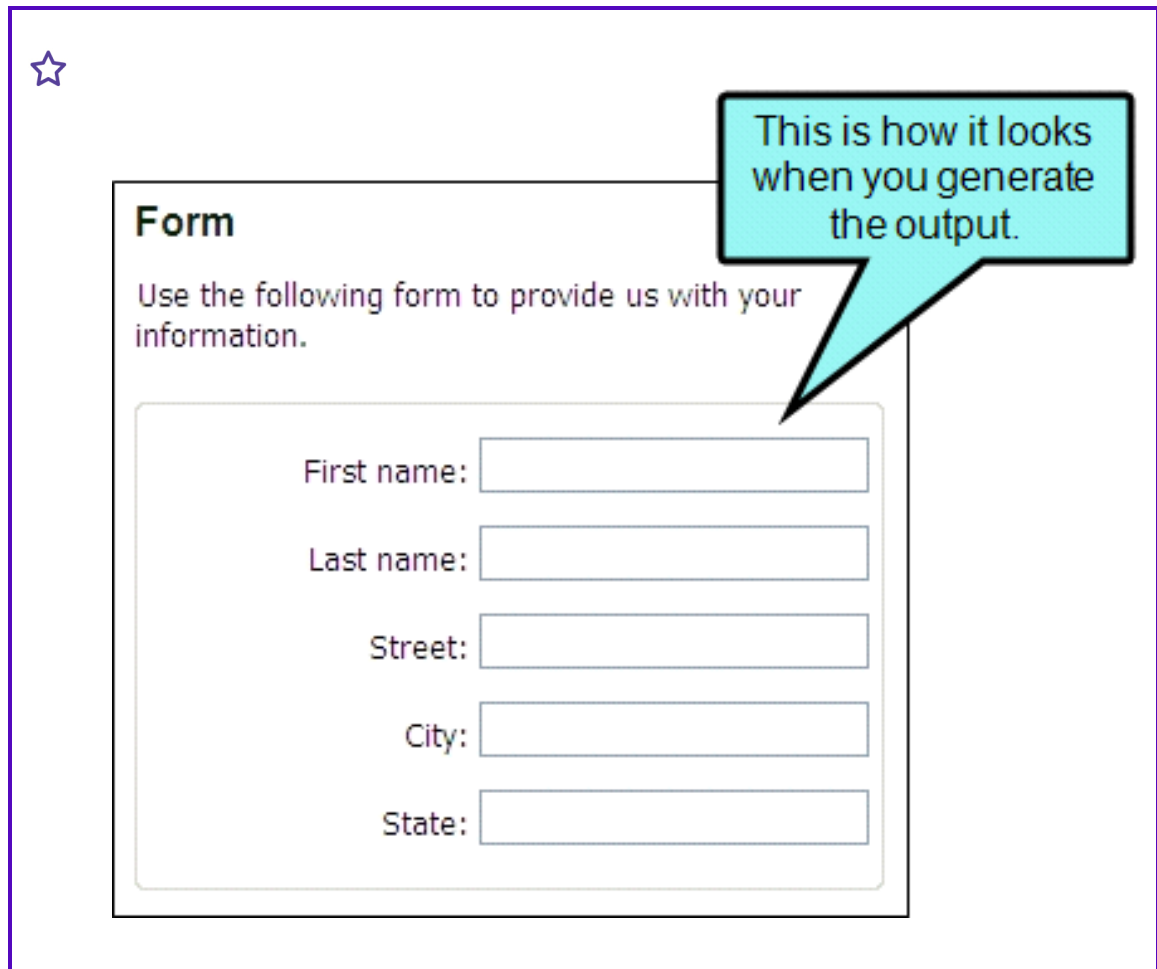
**Address:**

1234 Anywhere Road  
Sometown, CA 92438


- **form** The <form> tag is used to create a form for user input. After you create the group with the <form> tag, you can open the topic in the Internal Text Editor to supply the necessary code for the form fields.

☆ **EXAMPLE** The following image shows multiple <p> tags grouped into a form tag in the XML Editor. We also grouped the content into a <fieldset> tag, in order to place the form fields in a box. The structure bars on the left side of the content provide a visual representation of the grouping.






4. (Optional) You can create a div style class and apply it to the div you created to change its look.

 **NOTE** If you are attempting to simply indent content, see the online Help for information about indenting paragraphs or lists.


# Setting the HTML Style for Topics

Using the Properties dialog, you can select an HTML style class for a topic. This is the outer tag in a topic, so the properties for this class will be applied to the entire topic. The following style classes are provided in the drop-down list: task, concept, reference, topic. These options are typically used if you are generating DITA output from the project. If you do not select any of these, the "topic" class is applied to the topic by default. If you want to use a custom style class instead, you can open the stylesheet, select the **html** style, and create your own topic style class.


## How to Set the HTML Style for a Single Topic

1. Open the Content Explorer.
2. Find and select the topic.
3. In the local toolbar of the Content Explorer, click . The Properties dialog opens.
4. Select the **Topic Properties** tab.
5. In the **Topic Style Class** field, click the down arrow and select a class from the list.
6. Click **OK**.

## How to Set the HTML Style for Multiple Topics

1. Select **View > File List** or press **CTRL+SHIFT+J** on your keyboard.
2. From the **Filter** list in the local toolbar, select **Topic Files (\*.htm;\*.html)**.
3. Select the files for which you want to set a style class. You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
4. In the local toolbar, click . The Properties dialog opens.
5. Select the **Topic Properties** tab.
6. In the **Topic Style Class** field, click the down arrow and select a class from the list.
7. Click **OK**.

## What's Noteworthy?

 **NOTE** If you have imported DITA file content, the topic types associated with each of those files previously will remain that way, unless you override the settings in the Properties dialog.

# Styles for Generated Pages

Supported In:



Sometimes your output may display content that is entirely auto-generated, rather than pulling content from one of your topics. This occurs when search results are displayed. It also occurs if your output is integrated with MadCap Pulse and a user clicks the Edit User Profile button, which then displays information on the Pulse home page, with no topic content shown.

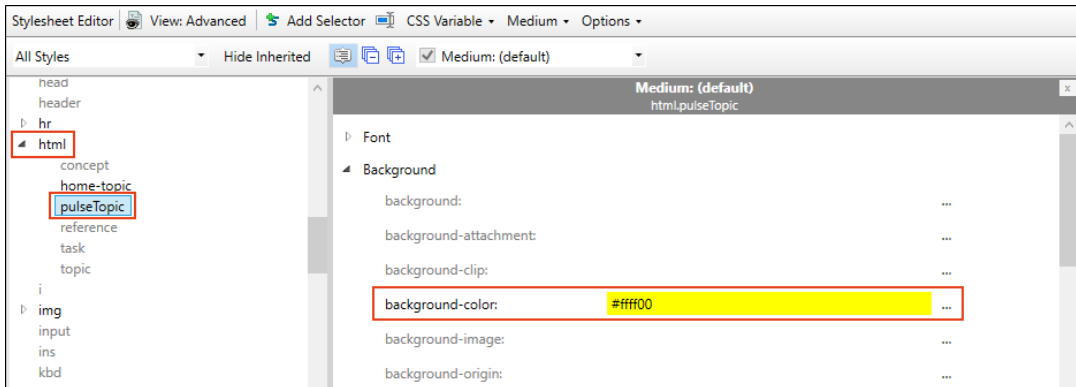
## Style Classes

For HTML5 Side and Top Navigation (and skinless outputs), you can add the following classes of the html style to control the look of these generated pages:

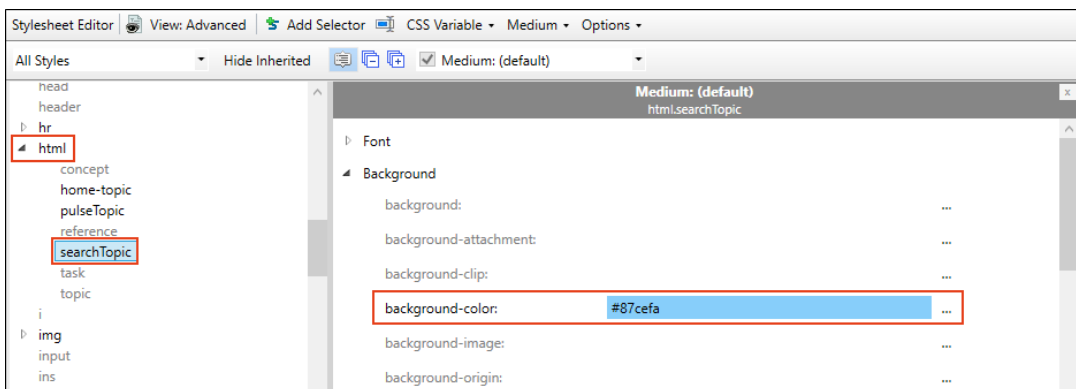
- **pulseTopic** This style class controls the look of a generated Pulse page (i.e., page opened via the Edit User Profile button).
- **searchTopic** This style class controls the look of a generated search results page.
- **templateTopic** This style class controls the look of all generated pages and has precedence over the other the pulseTopic and searchTopic classes.



☆ **EXAMPLE** You want Pulse-generated pages to show a yellow background. So you add a class to `html`, name it `pulseTopic`, and set the background color to yellow.

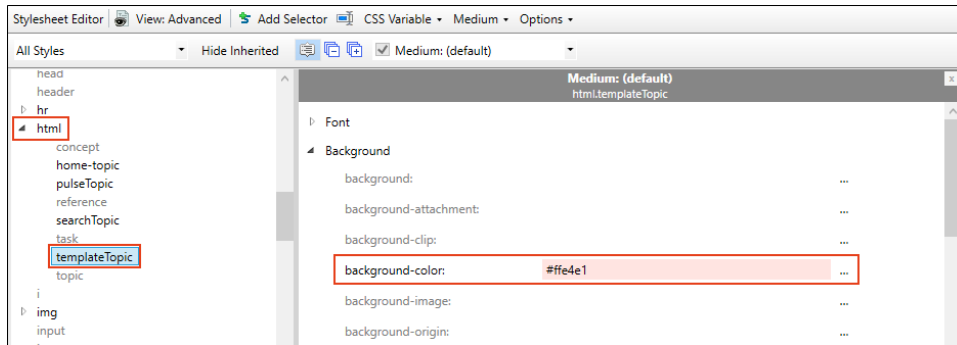


On the page showing search results, you want the background color to be light blue. So you add a class to `html`, name it `searchTopic`, and set the background color to light blue.



When you view those pages in the output, the background colors are just as you specified.

- ☆ But then let's say you add a class to `html`, name it `templateTopic`, and set the background color to light red.



As a result, the yellow and blue background colors will be overridden. Both kinds of generated pages will display with a light red background.

## Suggested Style Setting

If you include a side menu—via a Menu proxy—that is not context-sensitive, this menu may display on generated pages, not just in regular topics. This is probably something you want to avoid.

Therefore, to prevent this issue, you may want to copy the following to your stylesheet via the Internal Text Editor.

```
html.templateTopic div.sideContent
{
  display: none;
}
```

## What's Noteworthy?

- 📄 **NOTE** These styles are supported only in HTML5 Side Navigation, Top Navigation, and skinless outputs. They are not supported in HTML5 Tripane output.

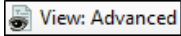
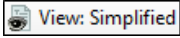

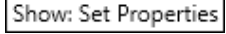




# Editing the Display for Help Control Links

Supported In:



When you insert Help control links (concept links, index keyword links, related topics links), you can specify whether the default setting for Help control links should be "list" or "popup". This can be done at the time you insert the link, but you can also use the following steps to specify this setting on a style. Therefore, that Help control link will always be used as the default setting, unless you override it at the spot where you have inserted the link. By default this is already set to "popup" for all of the Help control link types, but you might want to change it to "list."

# How to Edit the Display for Help Control Links


1. From the Content Explorer, open the stylesheet that you want to modify.
2. In the local toolbar, make sure the first button displays . If the button displays  instead, then click it.
3. In the upper-left corner of the editor, click in the drop-down field and select .
4. On the left side of the editor, find and select one of the following, depending on the type of Help control link: **MadCap|conceptLink**, **MadCap|keywordLink**, or **MadCap|relatedTopics**.
5. From the **Show** drop-down list on the upper-right side of the editor, select . This displays only the properties that have been set for that particular selector.
6. (Optional) You can use the toggle button in the local toolbar to show properties below in a group view  or an alphabetical view .
7. If you are using the group view, expand the **Unclassified** property group.
8. To the right of **mc-help-control-display**, click  and select one of the options:
  - **list** Displays the related links in a simple list.
  - **popup** Displays the related links in a popup window.
9. Click  to save your work.


# Setting Styles for Print Output in a Regular Stylesheet


A regular stylesheet lets you single-source formatting by setting the properties in one place and reusing them throughout your project. But what if you want your online output to look one way and your printed output to look another way? Rather than creating a style for online output and another style for printed output, you can use a single style and provide it with two sets of properties—one set to use for online output and another set to use with printed output. You can accomplish this through the use of a medium in your stylesheet.

# How to Set Styles for Print Output in a Regular Stylesheet

1. From the Content Explorer, open the stylesheet that you want to modify.
2. Set the style properties to be used for your online output as follows:
  - a. In the local toolbar of the Stylesheet Editor, click in the **Medium** field and make sure the medium for your online output is selected.
  - b. Select a style and set the properties for it. Do this for each style that you want to use in your online output.
3. Set the style properties to be used for your printed output as follows:
  - a. In the local toolbar of the Stylesheet Editor, click the down arrow in the **Medium** field and select **Medium: print** or **Medium: [name of print medium]**.

 **NOTE** You can also create a new medium if necessary. You actually might find it preferable to do this. For example, if you want page breaks before a particular heading for print output, but not when users send online topics to the printer, it is a good idea to create a custom print medium. The reason for this is that browsers respect the settings in the "print" medium provided by Flare. Therefore, even though your online output style medium does not have page breaks set before that heading, the application will see that you *do* have a page break specified in the print medium. And when a user tries to print a topic from your online Help, the printer will start a new page at that heading. The solution is to create a custom print medium (perhaps calling it "PDF"), specifying page breaks in that medium, and using it for your print output (instead of using the "print" medium provided by Flare).

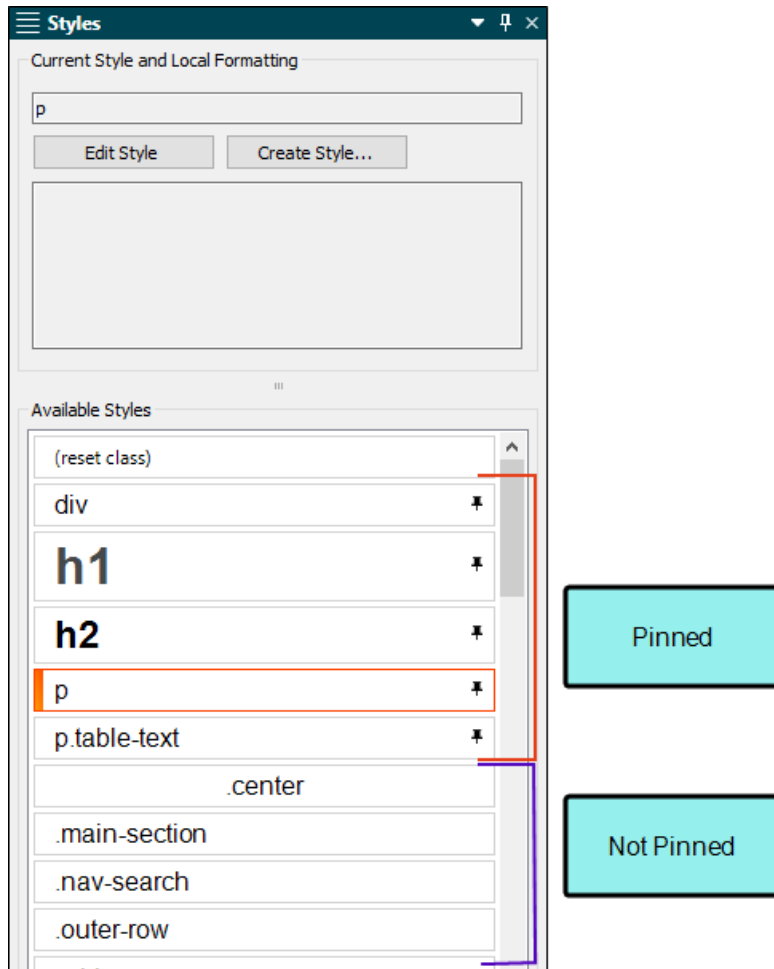
- b. Select the same styles that you edited for the online output, and set different properties for them (for the purpose of printed output).
    - c. Click  to save your work.
4. Apply the styles to the content throughout your project.  
See "Applying Styles to Content" on page 136.

5. Associate the online medium with your online target as follows:
  - a. Open the target to be used for online output (HTML5, Clean XHTML, Eclipse Help, Microsoft HTML Help, WebHelp, WebHelp Plus).
  - b. In the Target Editor, select the **Advanced** tab.
  - c. In the **Stylesheet Medium** section, select the medium that you used for online output.
6. Associate the print medium with your print target as follows:
  - a. Open the target to be used for print output (Adobe PDF, Microsoft Word).
  - b. In the Target Editor, select the **Advanced** tab.
  - c. In the **Stylesheet Medium** section, select the medium that you used for printed output (e.g., print).
7. Click  to save all files.

# Pinning Styles

You can pin your favorite styles to keep them displayed at the top of the list. This can be done in the following places in Flare:

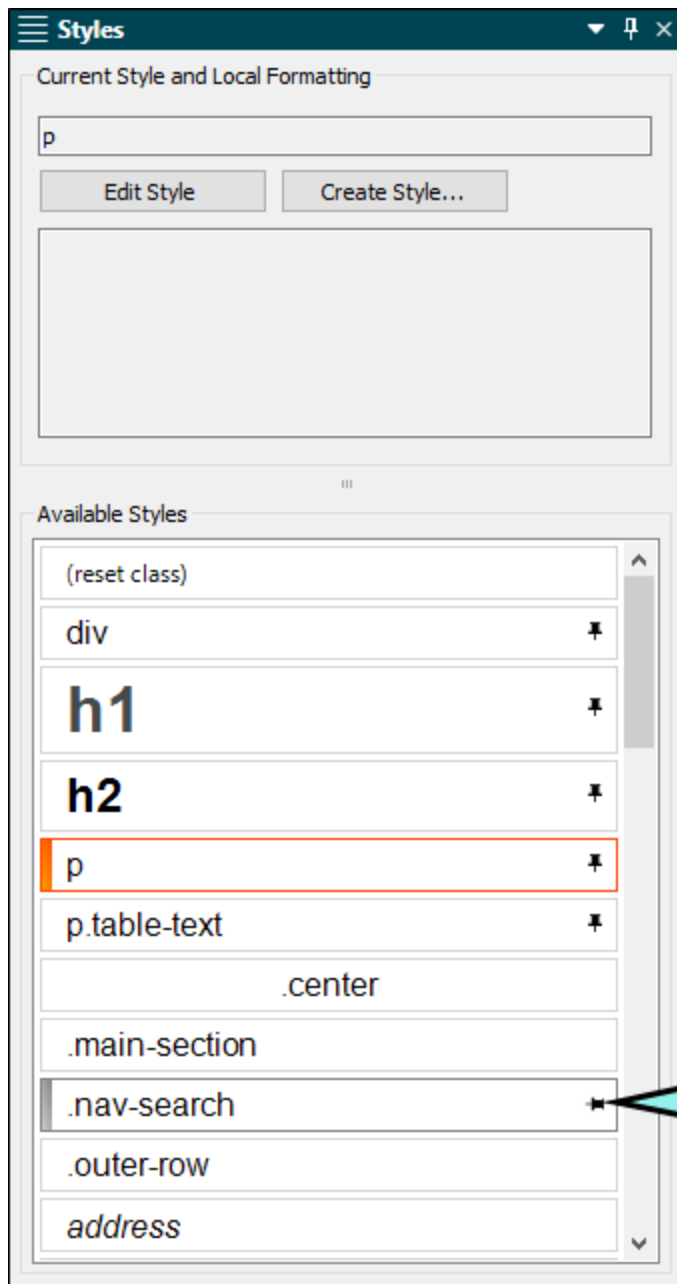
- Styles Window Pane (F12)
- Styles Drop-Down Field (Home Ribbon)
- Floating Style Picker (CTRL+SHIFT+H)





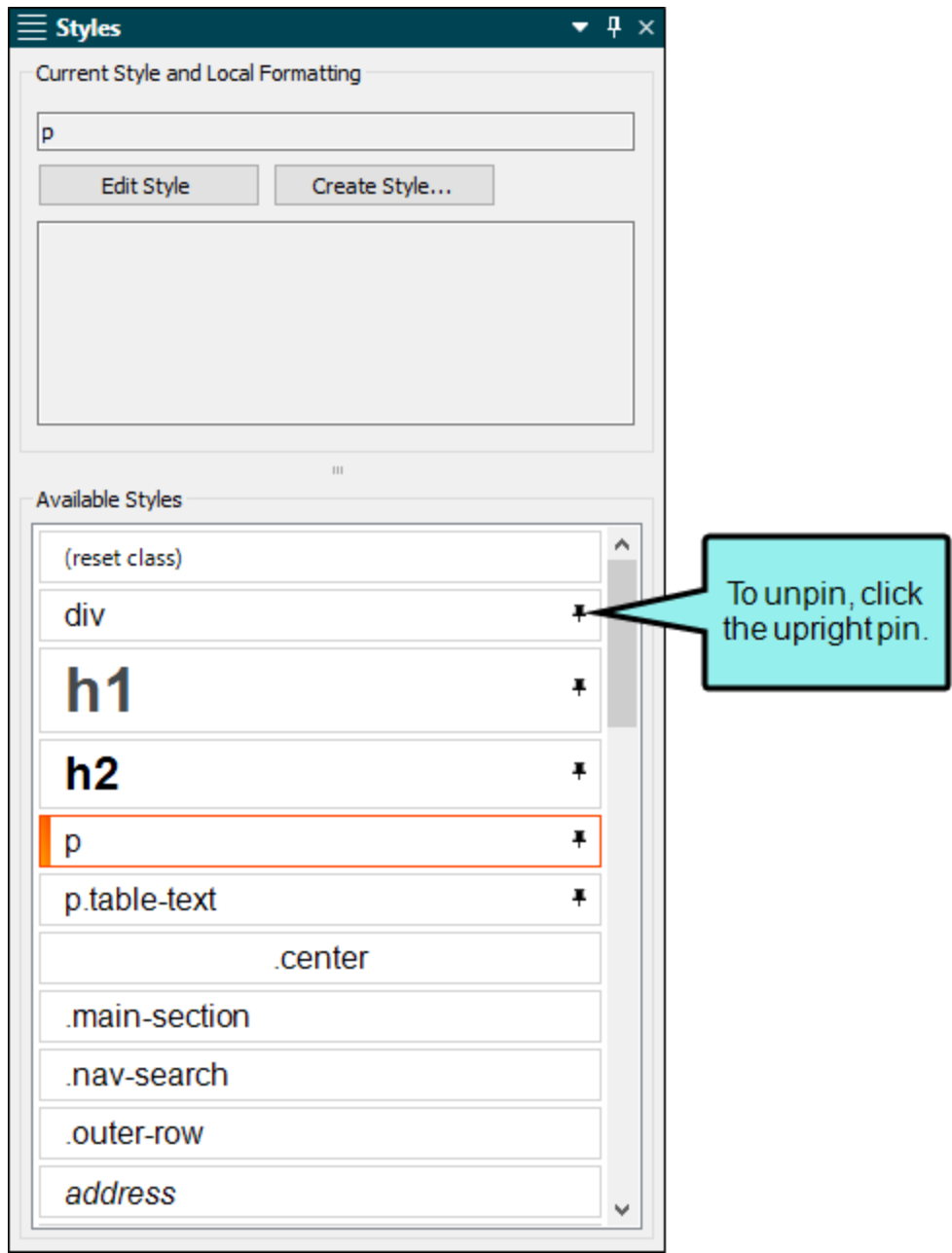
To pin a style, hover over the style row and click the little pin. It will then be moved to the top of the list.

**NOTE** If you do not see the style pinned right away, try closing the style interface you used (i.e., Styles window pane, drop-down, floating style picker), selecting different content in the topic or snippet, and then reopening the style interface.



To pin a style, hover over it and click the sideways pin.


To unpin a style, just click the pin again and it will be moved back to the bottom section of the list.

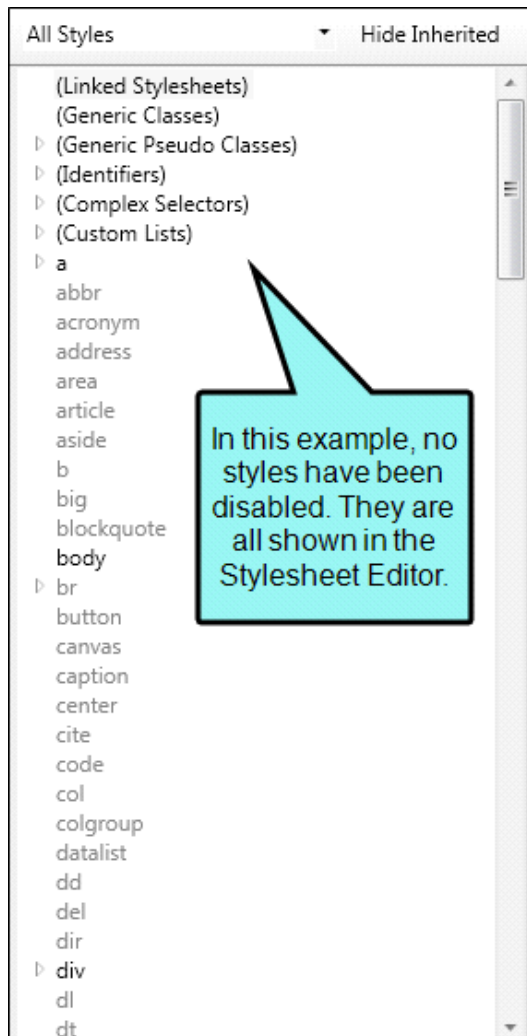


# Disabling and Hiding Styles

You cannot remove root styles provided by Flare. However, you can prevent certain styles from being shown in the Stylesheet Editor and in the Flare interface. Rather than being overwhelmed with the sight of all styles in the stylesheet, you can ensure that you see only the styles that you tend to use. Those styles will not be removed from the stylesheet; they will simply be hidden until you enable them.

## How to Disable Styles

1. From the Content Explorer, open the stylesheet that you want to modify.
2. In the local toolbar, click the **Options** button and select **Disable Styles**. The Disable Styles dialog opens.
3. On the right side of the dialog, select the styles that you want to disable. You can hold down the **SHIFT** or **CTRL** key and click, selecting a range of styles or many individual styles not next to each other.
4. Click . The selected styles are moved to the left side of the dialog.
5. Click **OK**. The style classes are removed from view in the stylesheet.



In this example, no styles have been disabled. They are all shown in the Stylesheet Editor.



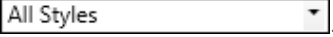
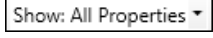




All Styles  Hide Inherited

- (Linked Stylesheets)
- (Generic Classes)
- (Generic Pseudo Classes)
- ▷ (Identifiers)
- ▷ (Complex Selectors)
- ▷ (Custom Lists)
- ▷ a
- body
- div
- ▷ h1
- h2
- h3
- h4
- h5
- h6
- ▷ img
- ▷ li
- MadCa
- MadCa
- MadCa
- ▷ ol
- ▷ p
- ▷ ul


This is the same style sheet. However, in this example many of the styles have been disabled, and are therefore hidden from view in the Stylesheet Editor.

# How to Hide Styles

If you don't want a style to be visible in the interface, but you also don't want to completely disable it, you can choose to hide it instead. This is slightly different from disabling a style because it is still accessible from the stylesheet, but it is still hidden in the interface.

1. From the Content Explorer, open the stylesheet that you want to modify.
2. In the local toolbar, make sure the first button displays  **View: Advanced**. If the button displays  **View: Simplified** instead, then click it.
3. In the upper-left of the editor, make sure the drop-down field is set to .
4. On the left side of the dialog, select the style that you want to hide.
5. From the **Show** drop-down list on the upper-right side of the editor, select .
6. (Optional) You can use the toggle button in the local toolbar to show properties below in a group view  or an alphabetical view .
7. If you are using the group view, expand the **Unclassified** group.
8. To the right of **mc-hidden**, click , and select **hidden**.  
To unhide a hidden property, select **unhidden**.
9. Click  to save your work.


## What's Noteworthy?

 **WARNING** Even if you have some styles that you never modify, you may want to avoid disabling them in your stylesheet. The reason for this is that disabling a style in a stylesheet also disables its use in the rest of the interface. For example, let's say that you have inserted several images in your project. When you do this, an `<img>` tag is used. Therefore, although you may never edit the properties for the `img` style, you still require it in order to insert images in the future. Otherwise, features such as the option to insert images become disabled in the interface. The bottom line is that you should use caution when disabling styles, making sure that you truly will not need to use those styles.

# Deleting Styles

You can delete style classes that you have added to a regular stylesheet. However, you cannot delete parent styles or classes added by Flare.

## How to Delete Styles

1. From the Content Explorer, open the stylesheet that you want to modify.
2. Select the style class you want to delete.
3. On your keyboard press **DELETE**.
4. Click  to save your work.

# Units of Measurement

When changing the look of content, you may often need to select a unit of measurement (UOM). Flare offers the following relative and absolute UOMs (definitions are taken from w3c.org).

## Relative UOMs

Following are UOMs that are relative. In other words, they might be displayed differently, depending on the circumstance.

**Pixel (px)** Pixel units are relative to the resolution of the viewing device (i.e., most often a computer display). If the pixel density of the output device is very different from that of a typical computer display, the user agent should rescale pixel values. It is recommended that the reference pixel be the visual angle of one pixel on a device with a pixel density of 96 dpi and a distance from the reader of an arm's length. For a nominal arm's length of 28 inches, the visual angle is therefore about 0.0213 degrees.

---

**Ems (em)** The em unit is equal to the computed value of the font size property of the element on which it is used. The exception is when "em" occurs in the value of the "font-size" property itself, in which case it refers to the font size of the parent element. It may be used for vertical or horizontal measurement. (This unit is also sometimes called the quad-width in typographic texts.)

---

**Percentage (%)** This is the percentage value of the element. Please be aware that if you are using percentage for the size of an object such as an image, the percentage refers to the block containing that image, not to the image itself. For example, if you have an image in a topic and set the width to 60%, this does not mean that the image will be reduced to 60% of its size. Instead, it means that the image will be resized so that its width is 60% of the "container" where it is inserted.

---



**X-Height (ex)** The ex unit is defined by the element's first available font. The x-height is so called because it is often equal to the height of the lowercase "x." However, an "ex" is defined even for fonts that don't contain an "x."


The x-height of a font can be found in different ways. Some fonts contain reliable metrics for the x-height. If reliable font metrics are not available, the x-height may be determined from the height of a lowercase glyph. One possible heuristic is to look at how far the glyph for the lowercase "o" extends below the baseline, and subtract that value from the top of its bounding box. In the cases where it is impossible or impractical to determine the x-height, a value of 0.5 em should be used.

---

<b>0-Width (ch)</b>	Relative to the width of the 0 (ZERO, U+0030) glyph in the element's font.
<b>Root-element (rem)</b>	Relative to the font size of the root element.
<b>Viewport-width (vw)</b>	Relative to 1% of the width of the viewport. <sup>1</sup>
<b>Viewport-height (vh)</b>	Relative to 1% of the height of the viewport.
<b>Viewport-min (vmin)</b>	Relative to 1% of the viewport's smaller dimension.
<b>Viewport-max (vmax)</b>	Relative to 1% of the viewport's larger dimension.

---

<sup>1</sup>The viewport is the size of the browser window.

 **NOTE** For more information, see <http://www.w3.org/TR/css3-values/> and [http://www.w3schools.com/cssref/css\\_units.asp](http://www.w3schools.com/cssref/css_units.asp).

# Absolute UOMs

Following the UOMs that are absolute. In other words, they are always displayed the same way.

**Point (pt)**            The points used by CSS 2.1 are equal to 1/72nd of an inch.

---

**Centimeter (cm)**

---

**Millimeter (mm)**

---

**Inch (in)**            One inch is equal to 2.54 centimeters.

---

**Pica (pc)**            One pica is equal to 12 points.


# Property Values

You can perform the following additional tasks with property values:


## Cutting, Copying, and Pasting Style Property Values

In both the Stylesheet Editor and Formatting window pane, you can display properties in alphabetical view, as opposed to grouped view. When you are in alphabetical view, you can select multiple properties for a particular selector. Then you can cut/copy and paste these properties into another selector; they can also be pasted in the Stylesheet Editor or in a text-based editor, such as Notepad or the Internal Text Editor. Declarations can also be copied from a separate CSS document and into your stylesheet in Flare.


### How to Cut Style Property Values

1. Open the Stylesheet Editor or Formatting window pane.
2. Make sure the properties are displayed in alphabetical view. If they aren't, click  in the local toolbar to switch to that view.
3. Right-click the property (or properties) you want to cut. You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
4. From the context menu select **Cut**.

### How to Copy Style Property Values

1. Open the Stylesheet Editor or Formatting window pane.
2. Make sure the properties are displayed in alphabetical view. If they aren't, click  in the local toolbar to switch to that view.
3. Right-click the property (or properties) you want to copy. You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
4. From the context menu select **Copy**.

# How to Paste Style Property Values

1. Open the Stylesheet Editor or Formatting window pane.
2. Make sure the properties are displayed in alphabetical view. If they aren't, click  in the local toolbar to switch to that view.
3. Choose a selector where you want to paste the properties.
4. Right-click in the properties grid.
5. From the context menu select **Paste**.

## What's Noteworthy?





**NOTE** You can also delete property values from a selector. See "Deleting Style Property Values" on the next page.

# Deleting Style Property Values

In both the Stylesheet Editor and Formatting window pane, you can display properties in alphabetical view, as opposed to grouped view. When you are in alphabetical view, you can select multiple properties for a particular selector. Once selected, you can delete the values for those properties from the current selector.

## How to Delete Style Property Values

1. Open the Stylesheet Editor or Formatting window pane.
2. Make sure the properties are displayed in alphabetical view. If they aren't, click  in the local toolbar to switch to that view.
3. Right-click the property (or properties) you want to delete. You can hold the **SHIFT** key to select a range, or you can hold the **CTRL** key to select individual items.
4. From the context menu select **Delete**.

 **NOTE** You can also cut, copy, and paste style property values. See "Cutting, Copying, and Pasting Style Property Values" on page 215.

# CSS Variables

Supported In:



A CSS variable lets you place the value for a style in one place and reuse it throughout a stylesheet. As with other kinds of single-sourcing, this can help with speed, ease of use, and consistency. Whenever you want to change the value, you only need to do so in one place and the new value is propagated everywhere that the variable is referenced. You certainly could use the "find and replace" feature instead to change the value, but CSS variables are preferred because you won't need to worry about inconsistencies in your stylesheet. For example, some values—such as colors—can be written in various ways, so in those cases CSS variables make a lot of sense.


Of course, color isn't the only place where you can use CSS variables; they can be used for all kinds of properties. But color might be where CSS variables are most commonly seen, so in this section we are mainly using color to illustrate how CSS variables work.

## General Information

"Basics of CSS Variables" on the next page

## Process

1. "Creating CSS Variables" on page 221
2. "Inserting CSS Variables" on page 225
3. "Editing CSS Variables" on page 228
4. "Resolving CSS Variables for Internet Explorer" on page 232

 **NOTE** Branding is a common way to use CSS variables. The branding identified in the Branding.css file includes values associated with CSS variables in the project. In the regular stylesheet, the branding CSS variables are shown as inherited. If you change a CSS variable in the regular stylesheet, it would "win" precedence. However, it is a good idea to use the Branding Editor to manage your project's CSS variable values.

# Basics of CSS Variables

You can easily spot a CSS variable. It is any property that begins with two dashes.

```
--my-css-variable: green;
```

A CSS variable is sometimes called a “custom variable” because you can use whatever name you want (but don’t use spaces), rather than always adhering to standard CSS property names such as “color,” “width,” and “font-size.” For example, you might have a CSS variable written like this in order to identify a product color at your company.

```
--my-product-color: #bed420;
```

When a CSS variable is referenced by another property, it is done so with the `var()` function.

```
color: var(--my-product-color);
```


You can define a CSS variable either locally or globally. A locally defined CSS variable is set on a particular selector.

```
.banner-text  
{  
  --my-product-color: #bed420;  
}
```

On the other hand, a globally defined CSS variable is set on the `:root` selector.


```
:root  
{  
  --my-product-color: #bed420;  
}
```

When using the regular stylesheet, you can use the `:root` selector for your CSS variables.

 **NOTE** Using the regular stylesheet is one place to manage CSS variables. If you are using CSS variables for branding, it is a good idea to use the branding stylesheet (i.e., the `Branding.css` file) to group all of your CSS variables in one place. When using the regular stylesheet (or a skin), you will have access to those variables to insert them as property values for different selectors.

Following are some other important considerations of CSS variables:

- Normal rules of inheritance and cascading apply to CSS variables.
- They are case-sensitive, so make sure you are consistent when creating and using them.
- You can change the value under `@media` sections in order to override the value on the default medium.


 **EXAMPLE** You might have the following set in the default medium.

```
:root
{
  --my-product-color: #bed420;
}
```

But then you have this under the `@tablet` media query.

```
@media tablet
{
  :root
  {
    --my-product-color: #1c5a97;
  }
}
```

As expected, the color will change from green to blue once the screen size is reduced to a tablet.

 **NOTE** For more detailed information about CSS variables, you can refer to numerous third-party websites on the internet.

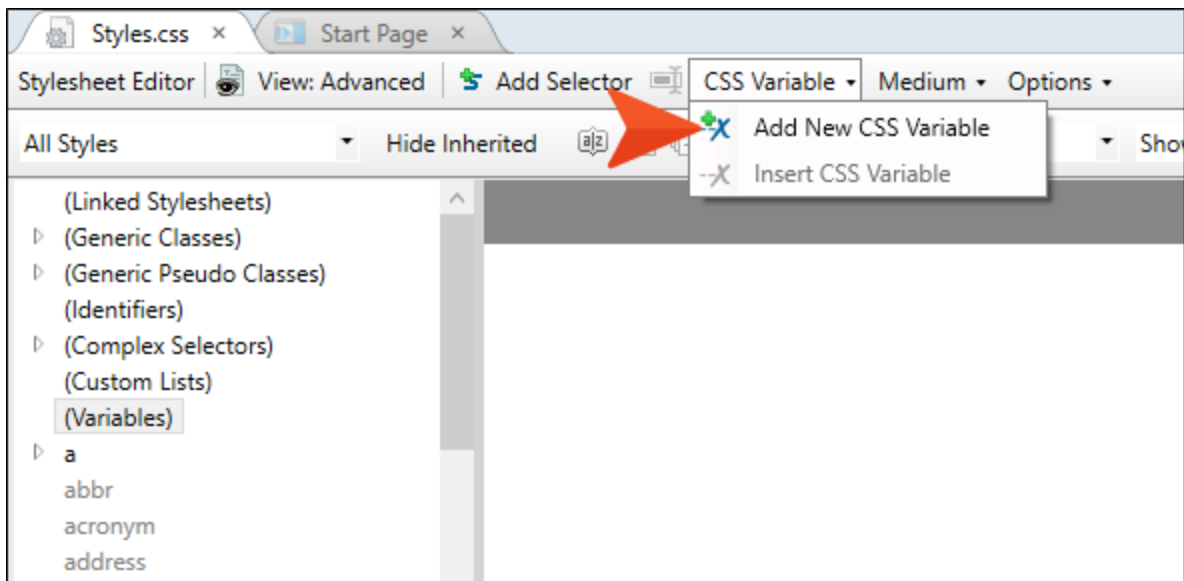


# Creating CSS Variables

The first step in using CSS variables is to create them in your stylesheet.

## How to Create a CSS Variable

1. From the Content Explorer, open the stylesheet that you want to modify.
2. In the local toolbar, click the **CSS Variable** down arrow and select **Add New CSS Variable**.



The Add New CSS Variable dialog opens.

3. Complete the fields:
  - **HTML Element** By default this field is populated with `:root` for new CSS variables. This is probably what you will use most of the time so that the variable is global. But you can replace this with a specific selector (e.g., `p`, `ol`, `img`) if you want.
  - **Name** Enter a name for the variable.
  - **Property Type** Select a property to associate with the variable. The most common property types are listed, but you can also choose "(Custom)" if you need something other than the options shown.

- **Value** Enter a value that is appropriate for the variable, based on the property type you selected.
- **Resulting CSS** As you complete the fields above, this field shows how the variable will appear in the stylesheet. If you forgot to add two dashes in front of the variable name, Flare adds them automatically in this field (e.g., `--MyVariable: #6be42e`).

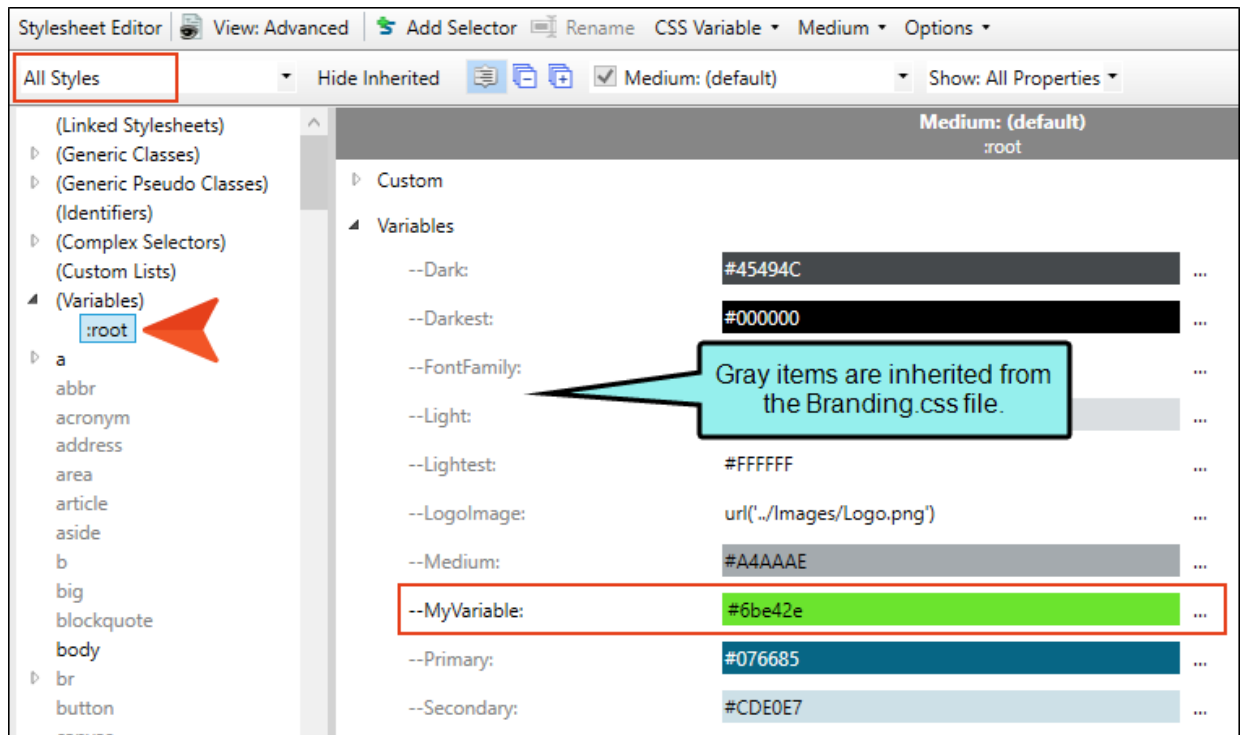
The screenshot shows a dialog box titled "Add New CSS Variable". It has a title bar with a question mark and a close button. The dialog is divided into several sections:

- HTML Element:** A dropdown menu with the value ":root".
- Name:** A text input field containing "MyVariable".
- Property Type:** A dropdown menu with the value "Color".
- Value:** A color picker showing the hex code "#6be42e".
- Resulting CSS:** A text area containing the CSS rule "--MyVariable: #6be42e;".

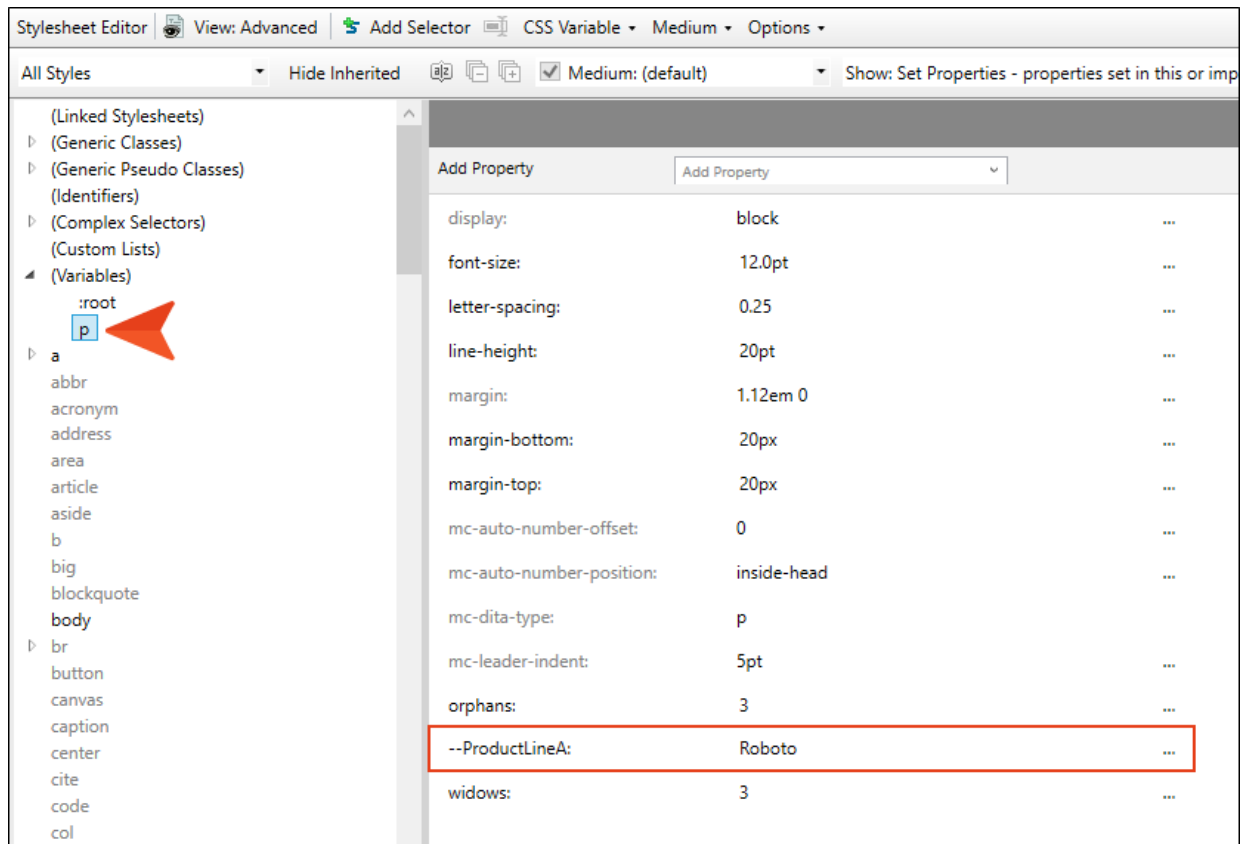
At the bottom of the dialog are two buttons: "OK" and "Cancel".

4. Click **OK**.

The new CSS variable is added to the stylesheet. It can be found in the Advanced View of the Stylesheet Editor in the (Variables) section. Make sure **All Styles** is selected in the filter field in order to see it.



If you set a CSS variable on a regular selector (rather than `:root`), it will also be seen under that section.



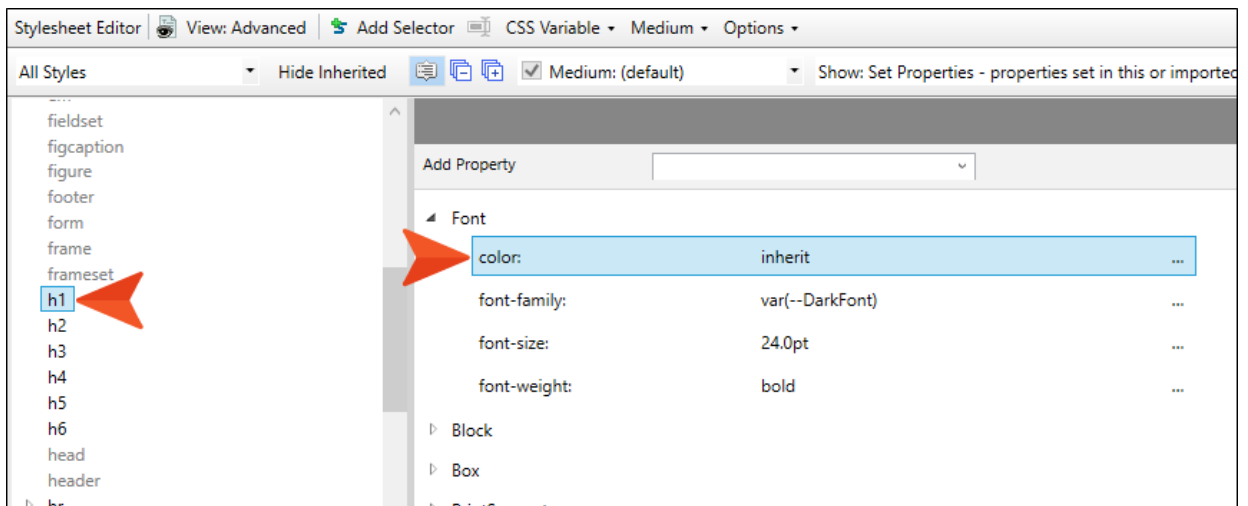
5. Click  to save your work.

# Inserting CSS Variables

After you've created CSS variables, you can insert them into style properties, in place of regular values.

## How to Insert (Use) a CSS Variable

1. From the Content Explorer, open the stylesheet that you want to modify.
2. In the local toolbar, make sure the first button displays **View: Advanced**. If the button displays **View: Simplified** instead, then click it.
3. Locate and select a style and property where you want to use the CSS variable. In the grid, all you need to do is select the appropriate row for that property.



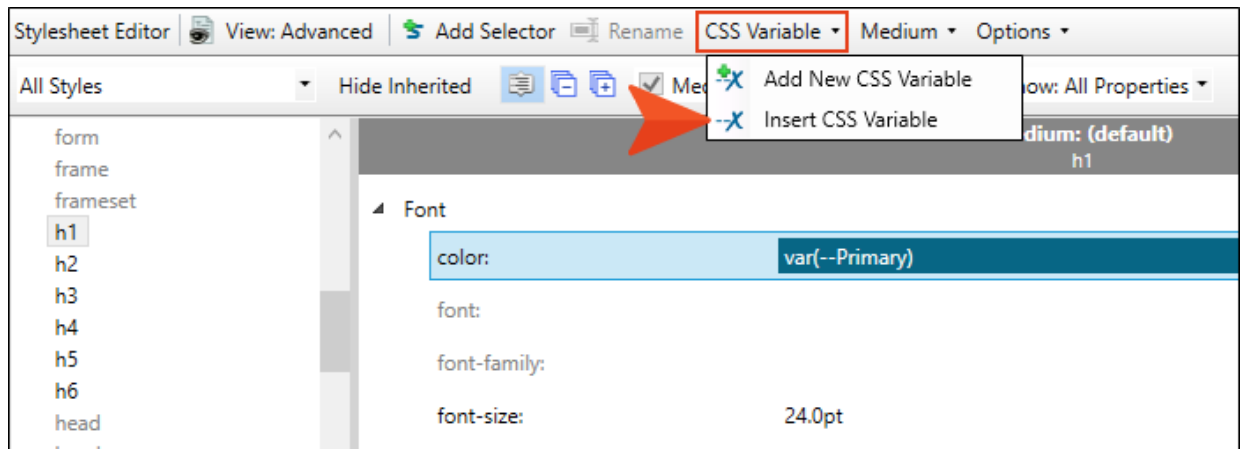
**NOTE** A project with a branding stylesheet already has CSS variables defined with default or custom values. You might see fields in the regular stylesheet that point to those CSS variables in the Branding.css file.

**NOTE** If you have a property that already contains a value and you want to add a CSS variable next to it, selecting the property row and using the following steps will overwrite the other value. However, you can easily work around this limitation by clicking in the value cell and simply typing the CSS variable syntax next to the existing other value(s).

For example, the following border-left property already has 2px solid as the value. So if you want to insert a CSS variable called "--Light-Gray2" next to it, you would type it like this:

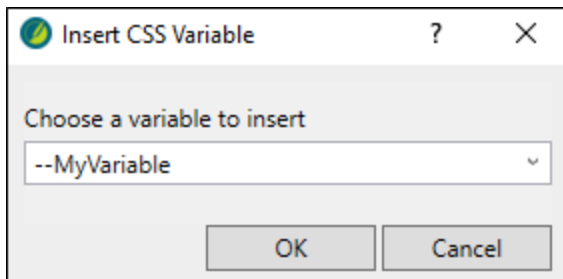
```
Border
border-left: 2px solid var(--Light-Gray2)
```

4. In the local toolbar, click the **CSS Variable** down arrow and select **Insert CSS Variable**.



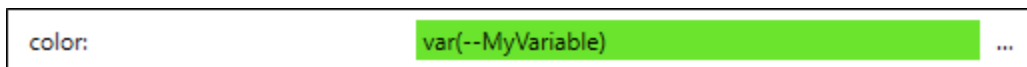
The Insert CSS Variable dialog opens.

- From the drop-down, select the CSS variable you want to insert.



- Click **OK**.

The property field changes, showing "var" followed by the name of the CSS variable in parentheses.






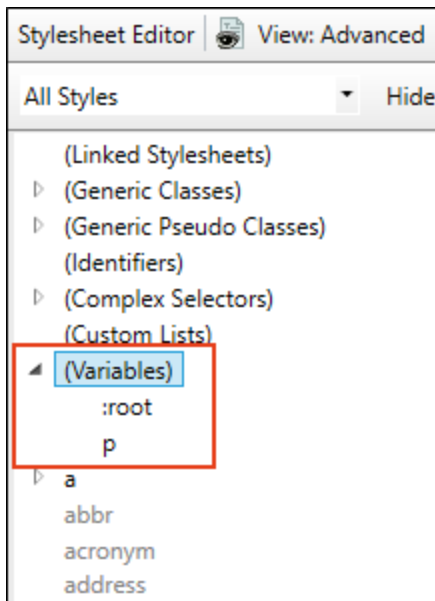
- Click  to save your work.

# Editing CSS Variables

Periodically, you might need to edit CSS variables that you've created.

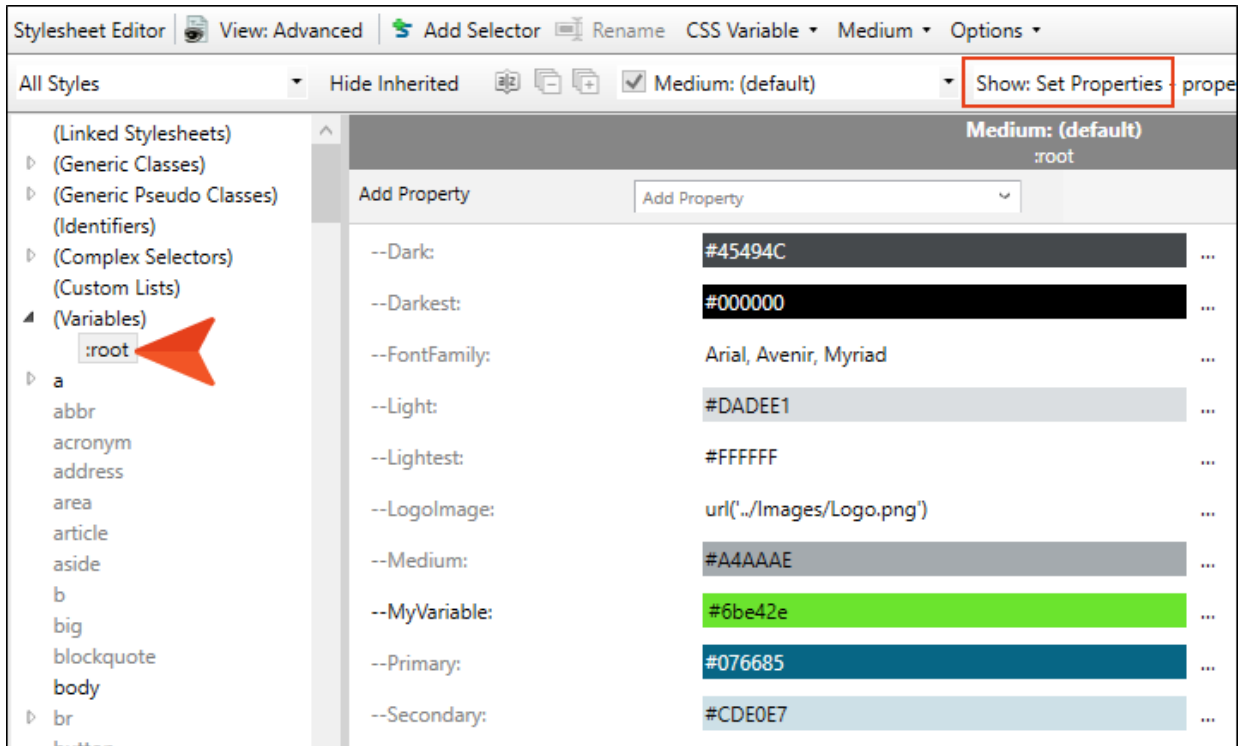
## How to Edit a CSS Variable



1. From the Content Explorer, open the stylesheet that you want to modify.
2. In the local toolbar, make sure the first button displays  **View: Advanced**. If the button displays  **View: Simplified** instead, then click it.
3. In the upper-left of the editor, make sure the drop-down field is set to  **All Styles**.
4. On the left side of the editor, expand **(Variables)**.
5. Click the HTML element with the CSS variable you want to edit. Most of the time this will probably be `:root`, but it might be a specific selector.



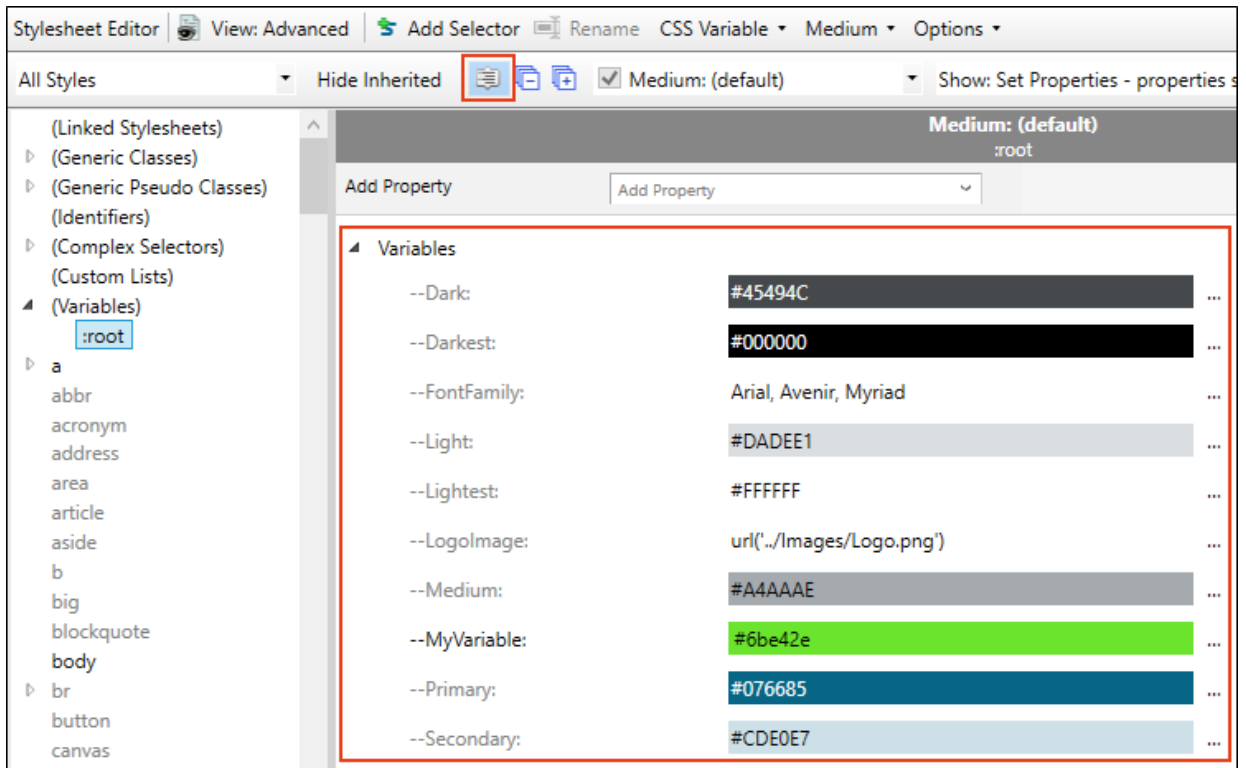



6. From the **Show** drop-down list on the upper-right side of the editor, select **Show: Set Properties**. This displays only the properties that have been set for that particular selector.

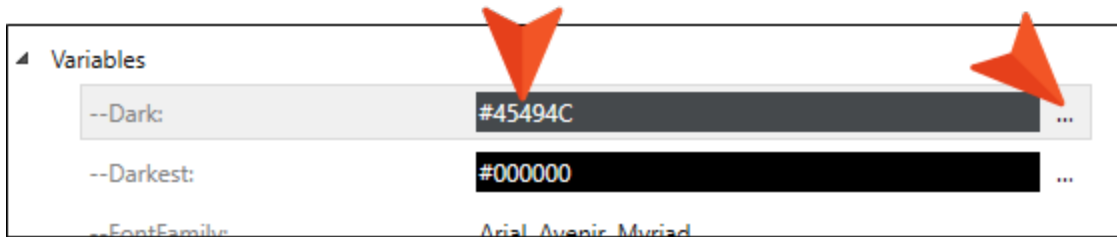



7. (Optional) You can use the toggle button in the local toolbar to show properties below in a group view  or an alphabetical view .

If it is in group view, expand **Variables** to see the variables under it.



- If you only want to change the property value, you can click in the row and type it. Otherwise, on the right side of the row (variable) you want to edit, click . The Edit CSS Variable dialog opens.



- You can change any of the fields in the Edit CSS Variable dialog:
  - **HTML Element** By default this field is populated with `:root` for new CSS variables. This is probably what you will use most of the time so that the variable is global. But you can replace this with a specific selector (e.g., `p`, `ol`, `img`) if you want.
  - **Name** Enter a name for the variable.
  - **Property Type** Select a property to associate with the variable. The most common property types are listed, but you can also choose "(Custom)" if you need something other than the options shown.
  - **Value** Enter a value that is appropriate for the variable, based on the property type you selected.
  - **Resulting CSS** As you complete the fields above, this field shows how the variable will appear in the stylesheet. If you forgot to add two dashes in front of the variable name, Flare adds them automatically in this field (e.g., `--MyVariable: #6be42e`).
- Click **OK**.
- Click  to save your work.

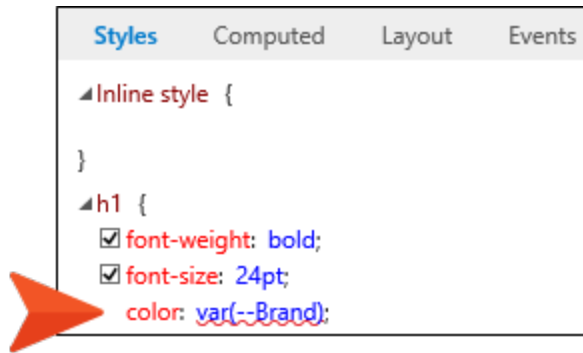
# Resolving CSS Variables for Internet Explorer

Unfortunately, CSS variables are not supported in Internet Explorer. However, there is an option on the Advanced tab of the Target Editor that is enabled by default to correct this.

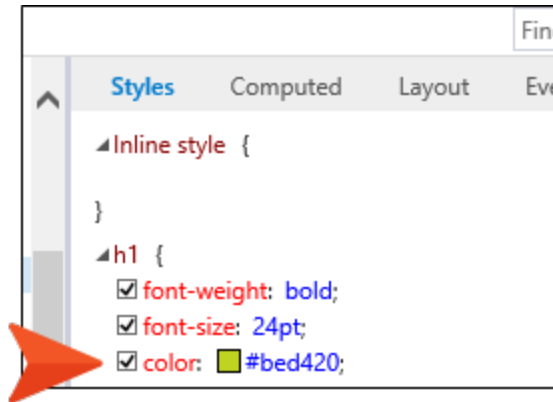
With this feature enabled, the CSS variable will be replaced with the actual value in the output. In other words, you will not see the `var(--[VariableName])` instance. However, this will be done only if the CSS variable is declared in the `:root` or `html` selector.

☆ **EXAMPLE** You create a CSS variable on the `:root` selector and you name it `--Brand.` On this variable, you set the color `#bed420`. Then, you insert that variable into the color property for your `h1` style.

If you *do not enable the option* to resolve CSS variables, the color will not be seen in the topic in Internet Explorer. And if you inspect that topic in the output (e.g., right click on the topic and select **Inspect element**), you will see this:

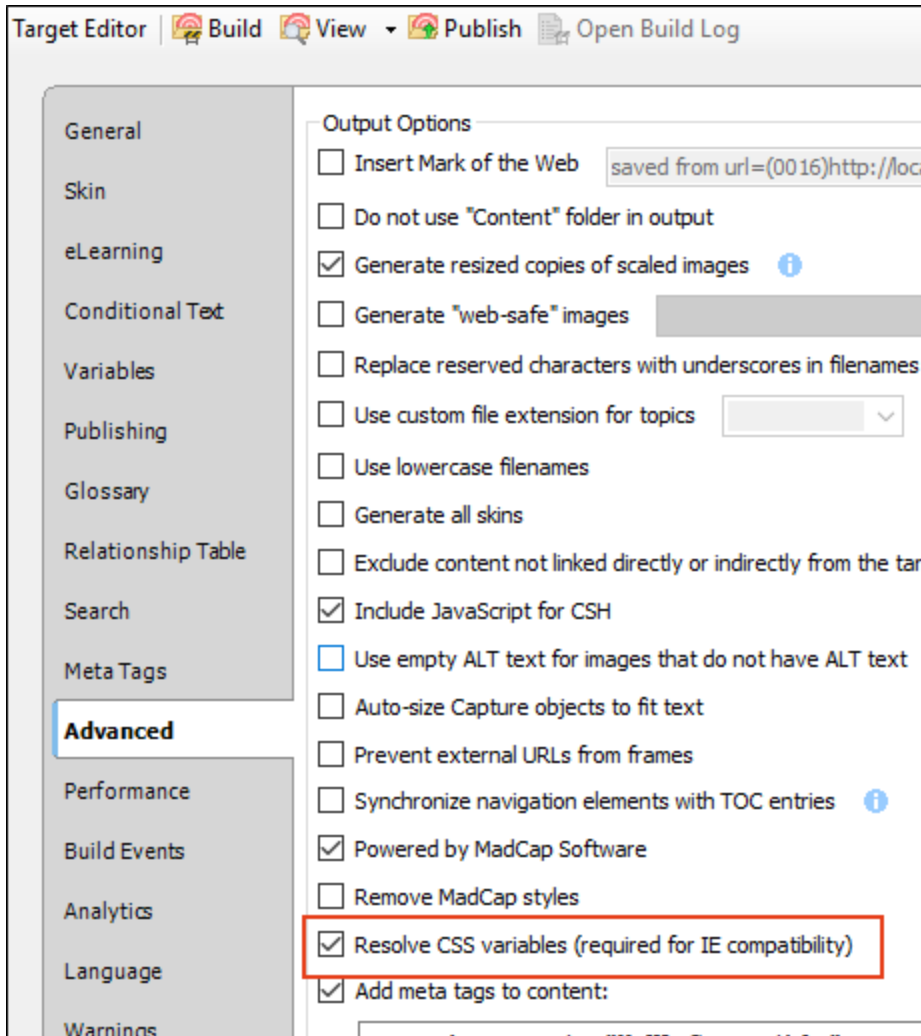


☆ But if you *do enable the option*, the color will be seen in the topic in Internet Explorer. And it will look like this if you inspect the topic:




# How to Resolve CSS Variables for Internet Explorer

1. Open an HTML5 target.
2. Select the **Advanced** tab.
3. Select **Resolve CSS variables (required for IE compatibility)**, if it is not already selected.

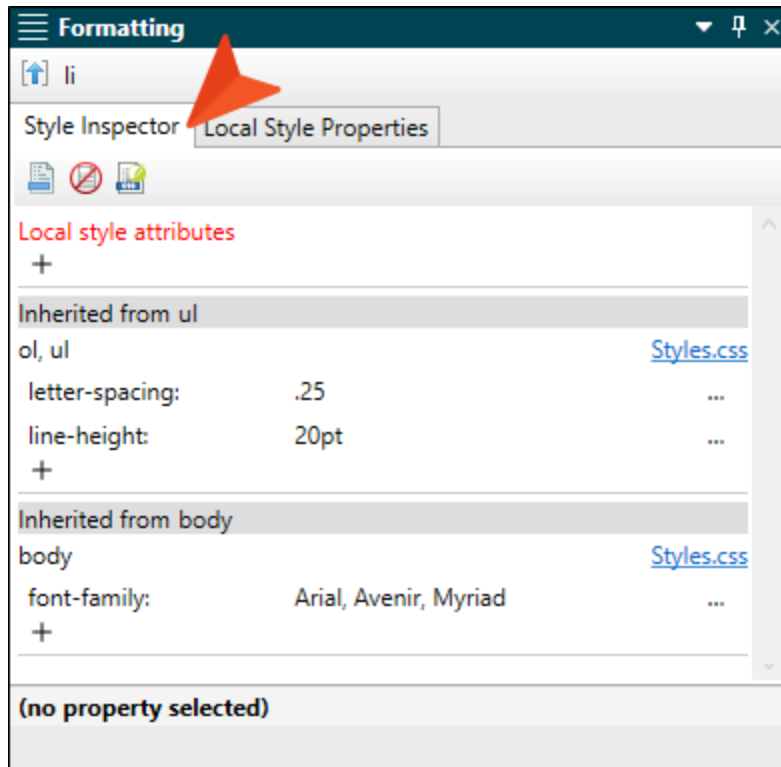


4. Click  to save your work.

 **NOTE** This option is not included in Eclipse and HTML Help targets, because the feature is always enabled for those outputs.

# Style Inspector

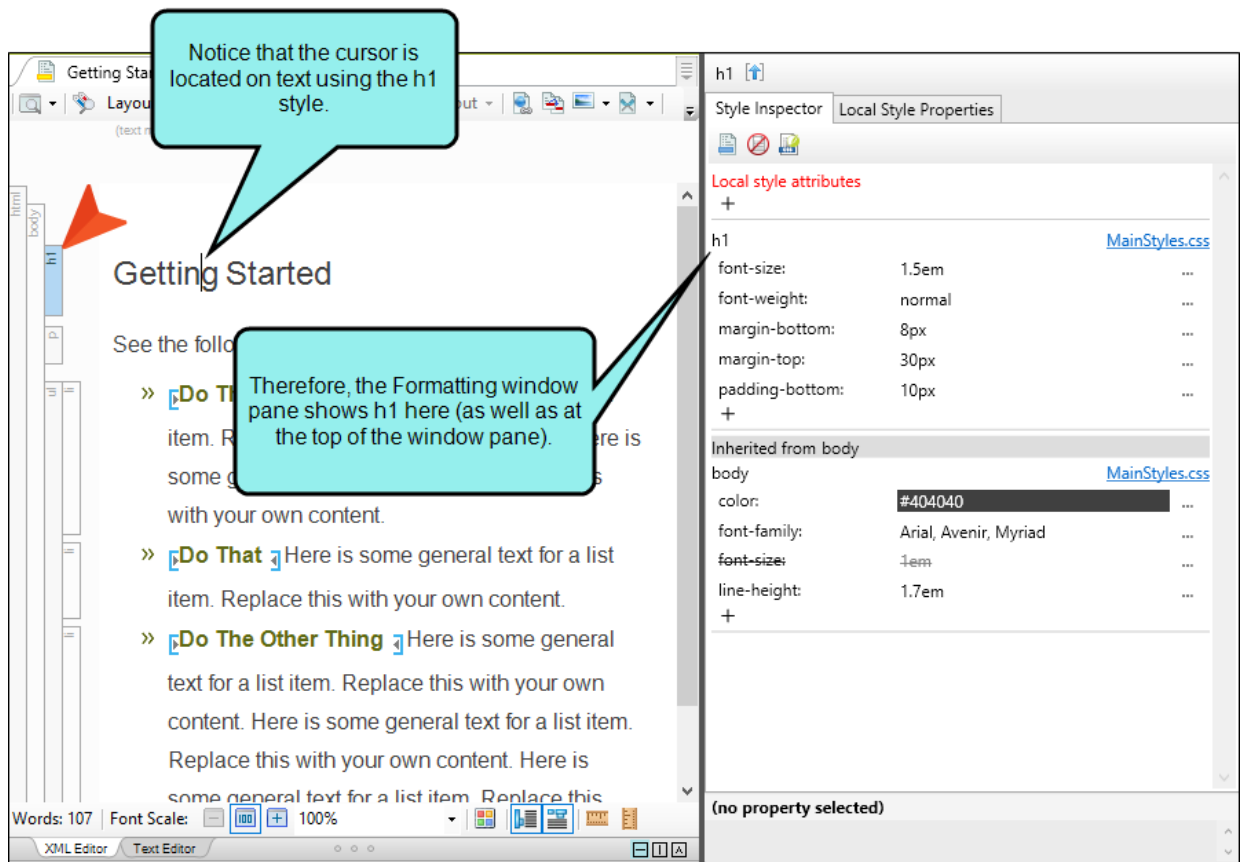
The Style Inspector is part of the Formatting window pane (**Home > Formatting Window**). This feature lets you see the style details for selected content in the open file (e.g., topic, snippet), and even edit those styles if necessary, without having to open the full stylesheet.




# Viewing Style Settings and Formatting for Specific Content

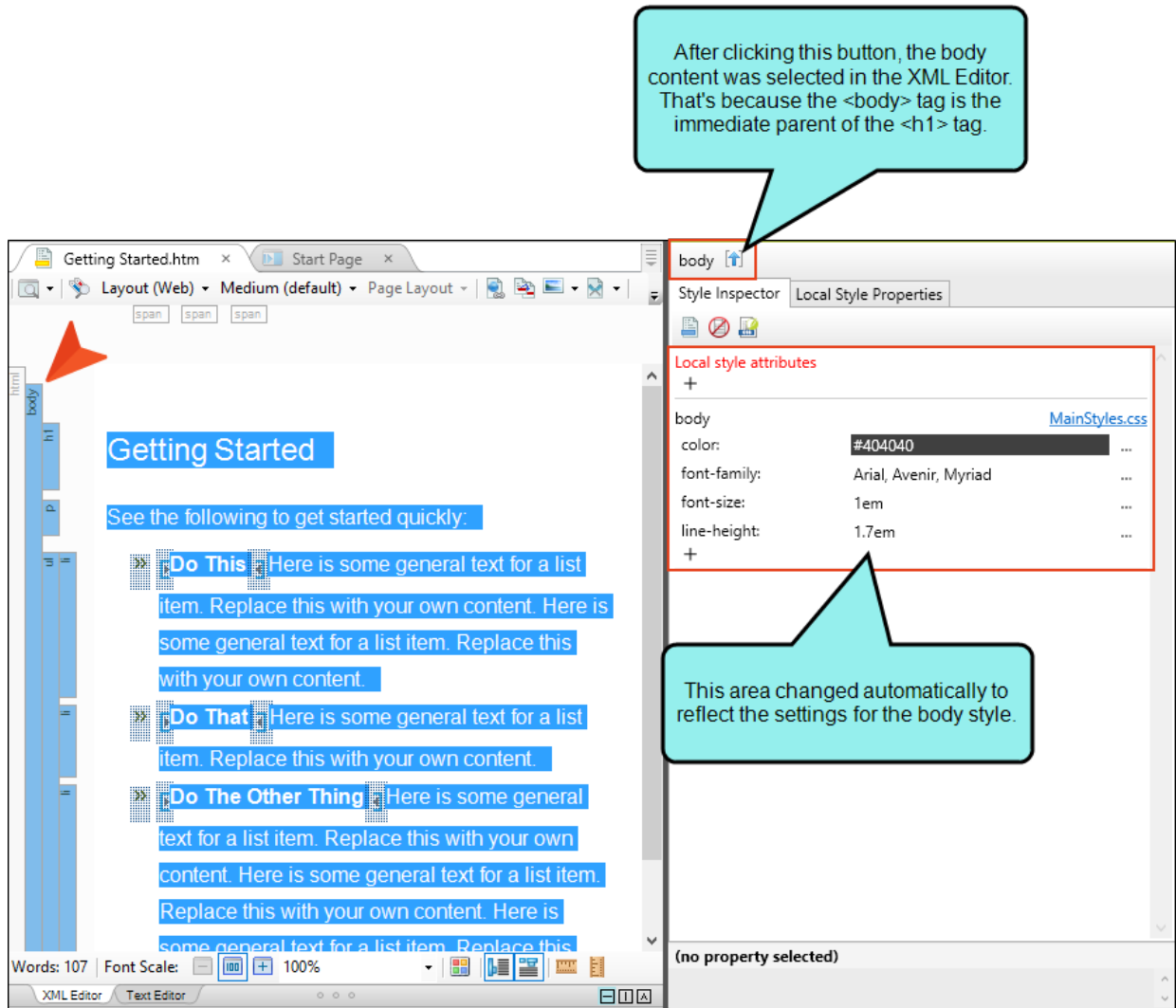
Whenever you click on certain content or select it in the XML Editor, the Style Inspector adjusts, showing the following information:

- Name of Style





If you click  next to the style name, the content for the parent tag will be selected and its style and properties will be shown in the window pane instead.



After clicking this button, the body content was selected in the XML Editor. That's because the <body> tag is the immediate parent of the <h1> tag.

Getting Started.htm x Start Page x

Layout (Web) Medium (default) Page Layout

span span span

body

h1

# Getting Started

See the following to get started quickly:

- Do This Here is some general text for a list item. Replace this with your own content. Here is some general text for a list item. Replace this with your own content.
- Do That Here is some general text for a list item. Replace this with your own content.
- Do The Other Thing Here is some general text for a list item. Replace this with your own content. Here is some general text for a list item. Replace this with your own content. Here is some general text for a list item. Replace this with your own content.

Words: 107 | Font Scale: 100%

XML Editor Text Editor

Style Inspector Local Style Properties

Local style attributes

body [MainStyles.css](#)

color: #404040

font-family: Arial, Avenir, Myriad

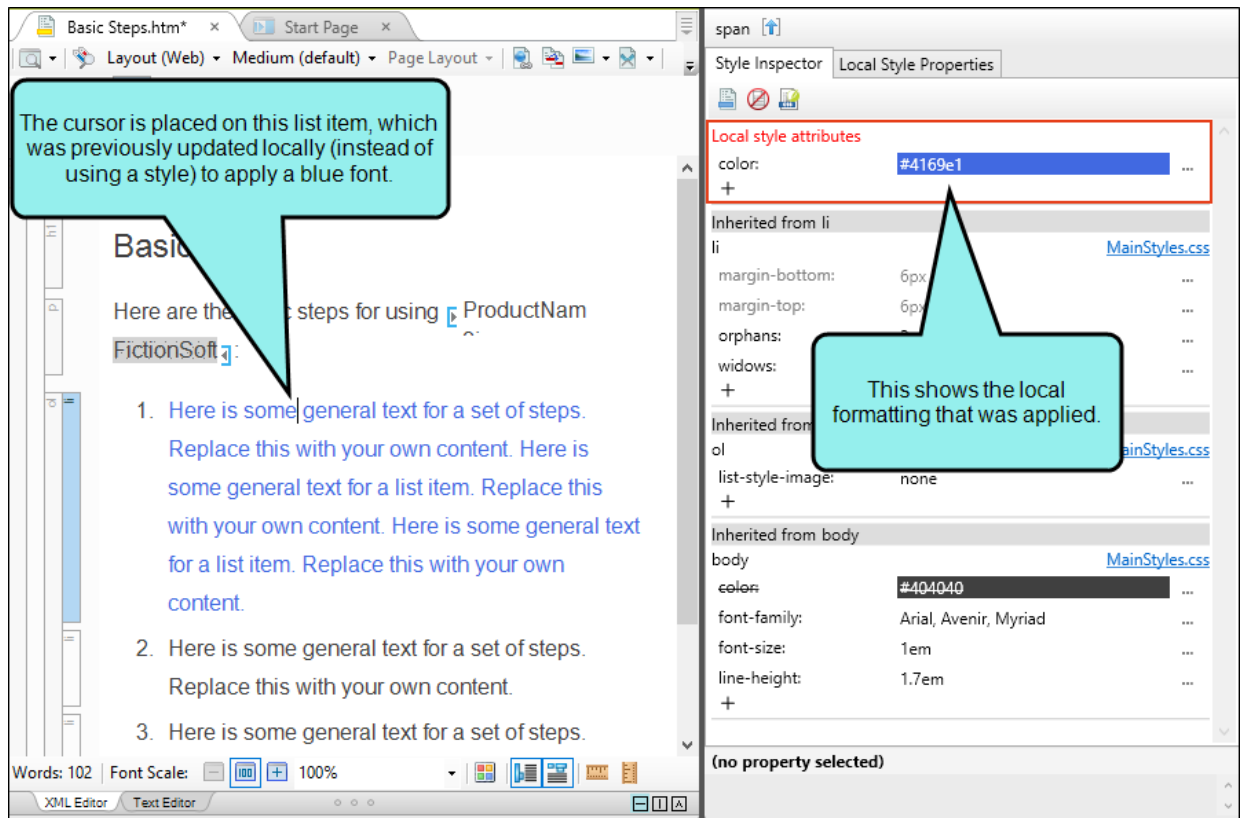
font-size: 1em


line-height: 1.7em

(no property selected)

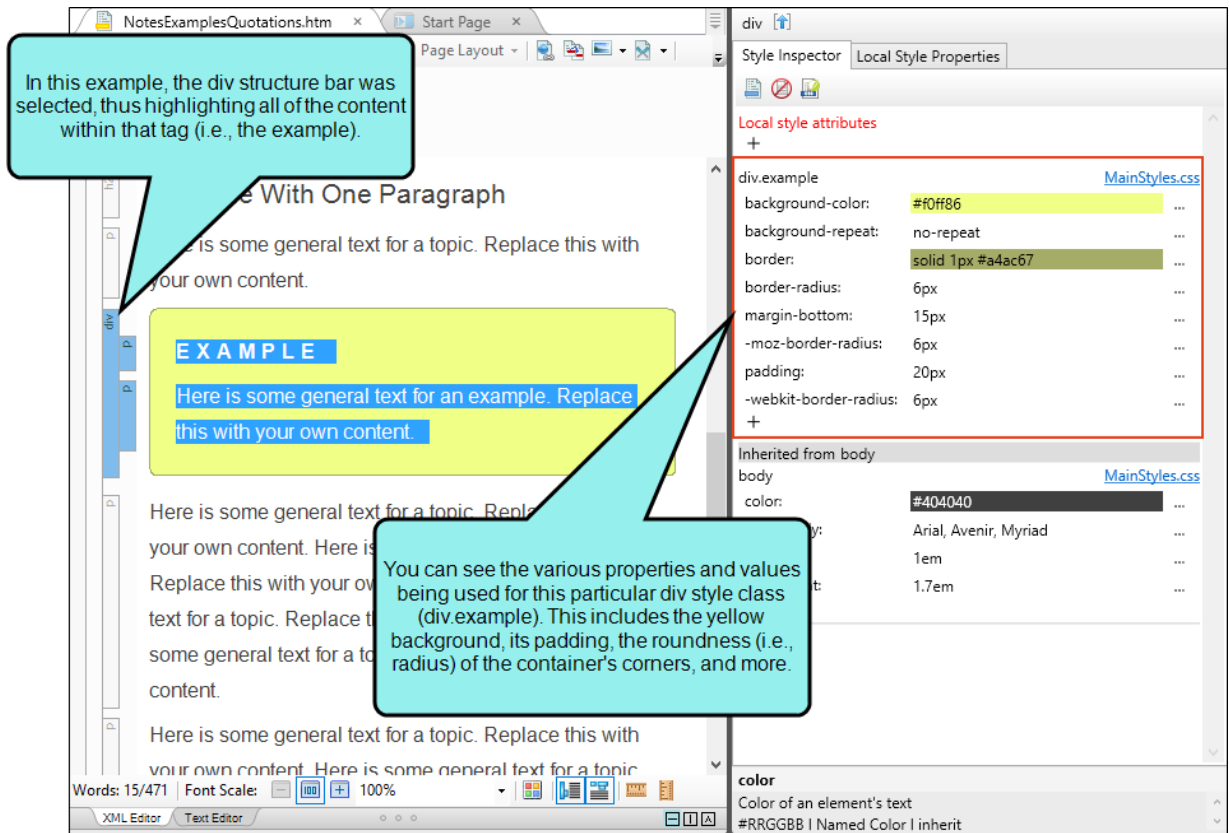
This area changed automatically to reflect the settings for the body style.

- **Local Formatting** If the selected content has any local formatting, this is indicated under the red “Local style attributes” area.

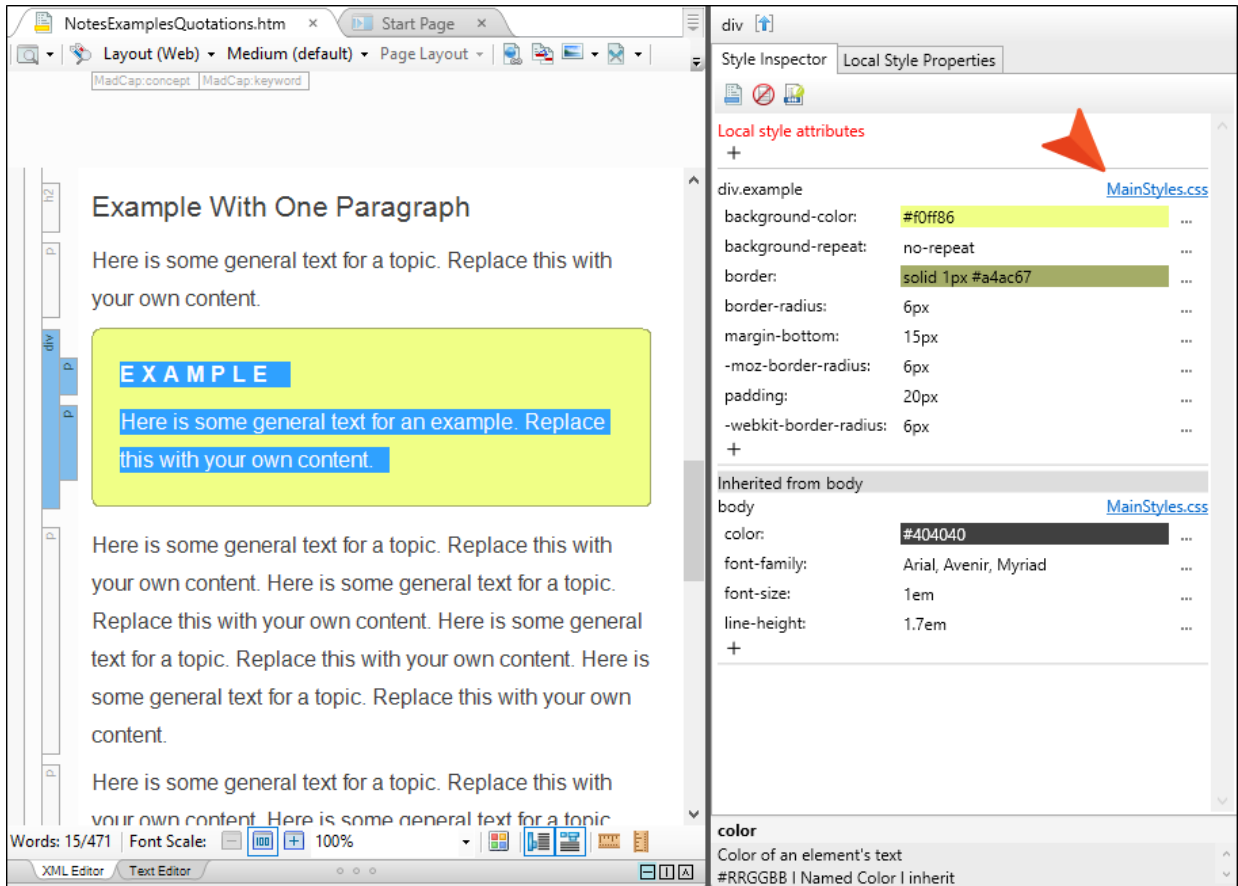


If you want to keep the look of the content, but use styles instead of local formatting, you can create a new style class based on the formatting. This can be done by using  in the local toolbar or the right-click menu. See "Creating Style Classes" on page 248.

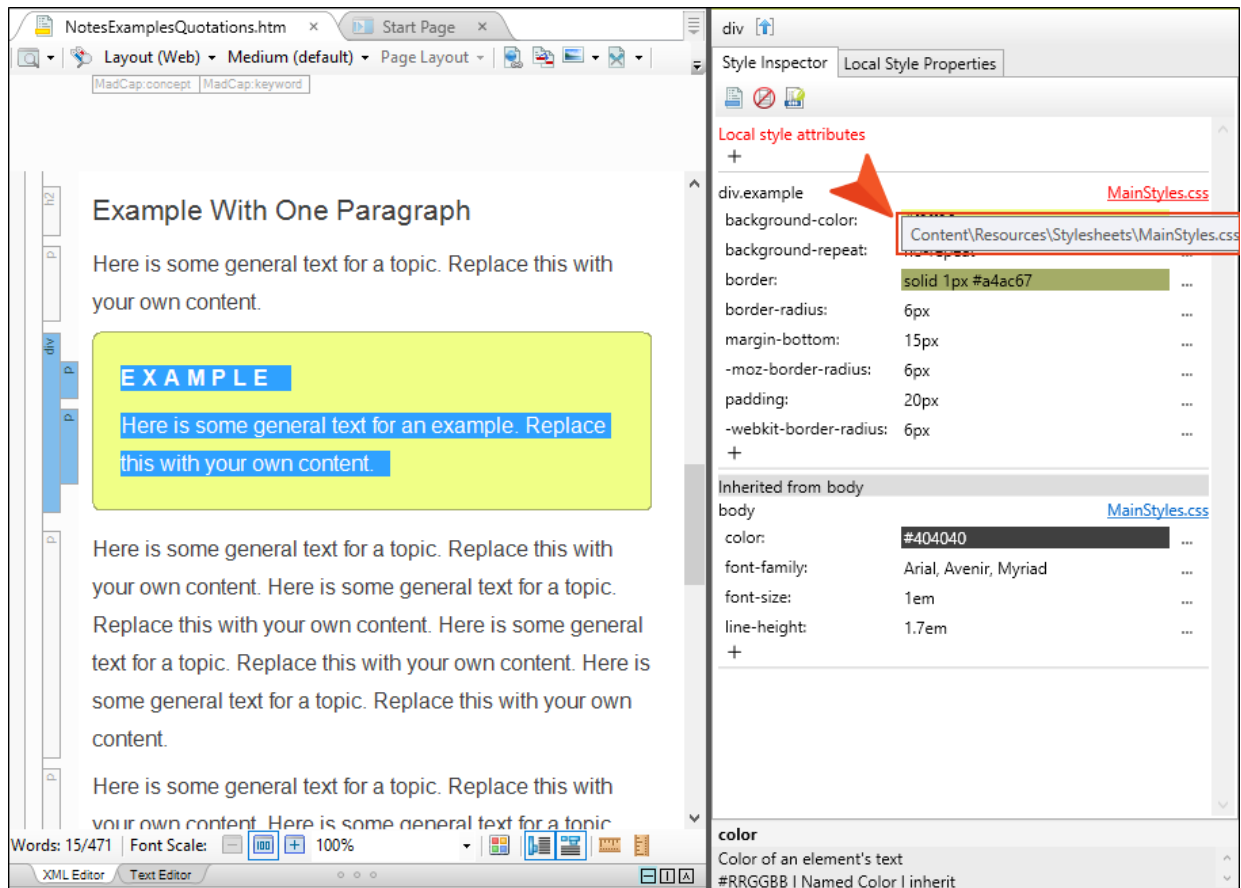
- **Style Properties and Values** Below any local attributes, the Style Inspector shows all of the properties and values that are explicitly set for the style.



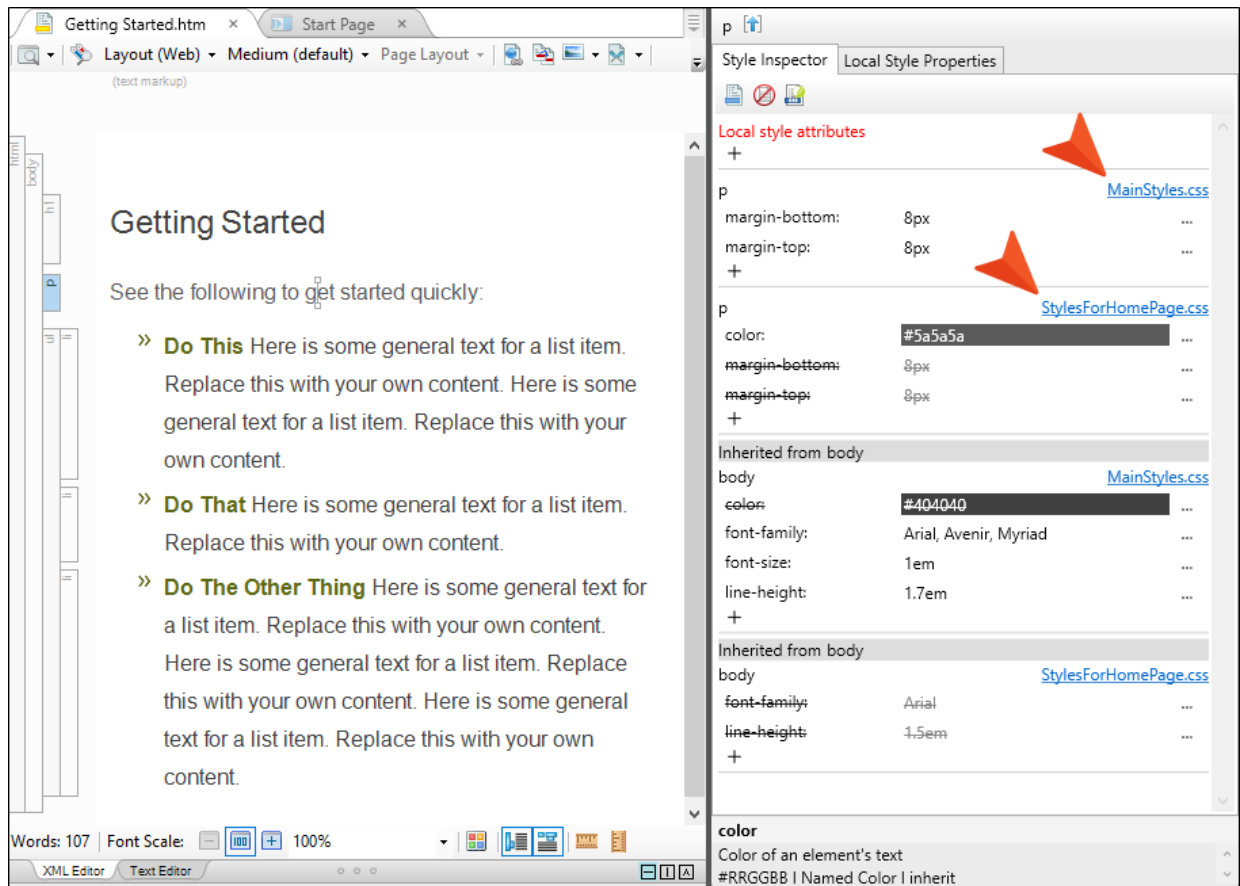
- **Stylesheet(s)** To the right of the style name, you can see the stylesheet where the properties are set.



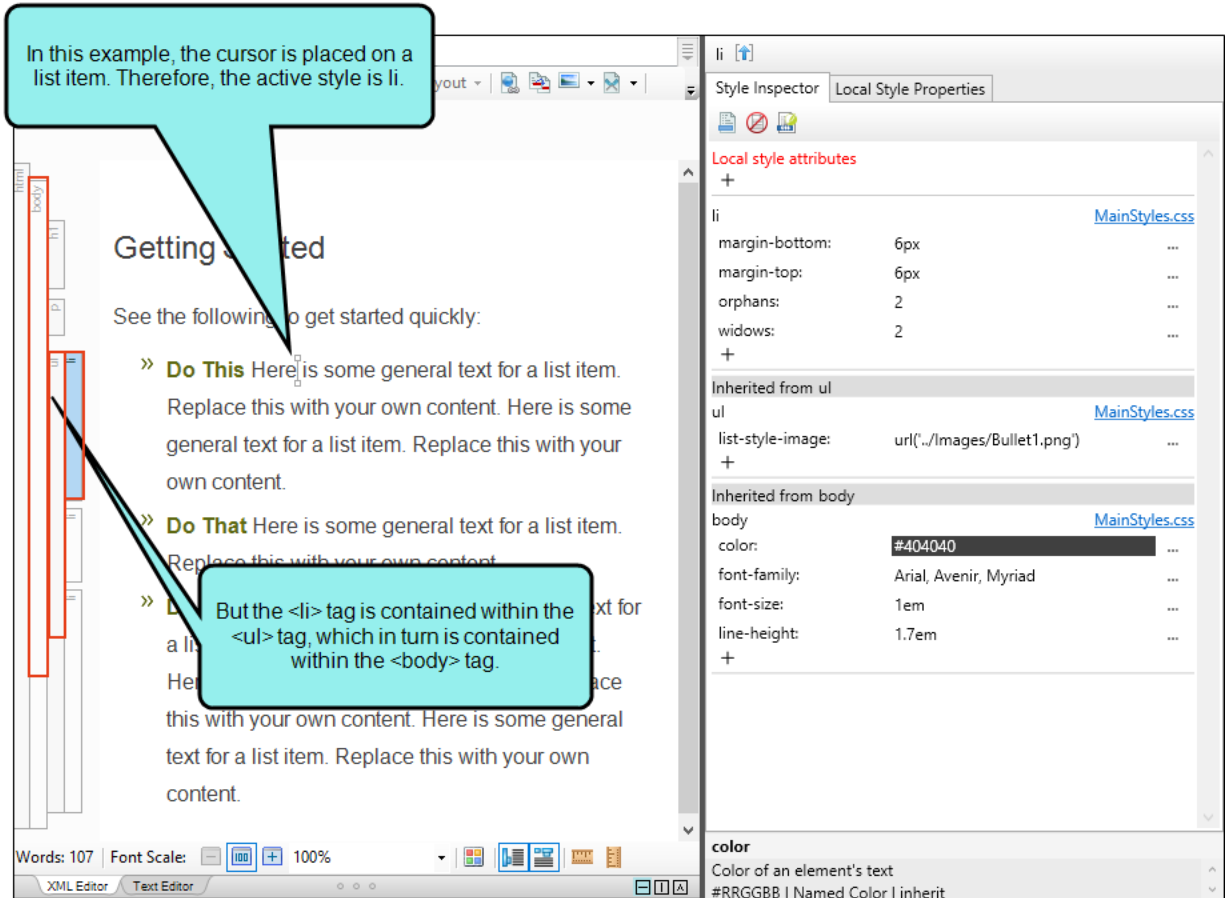
If you hover over the stylesheet name, you can see its path.

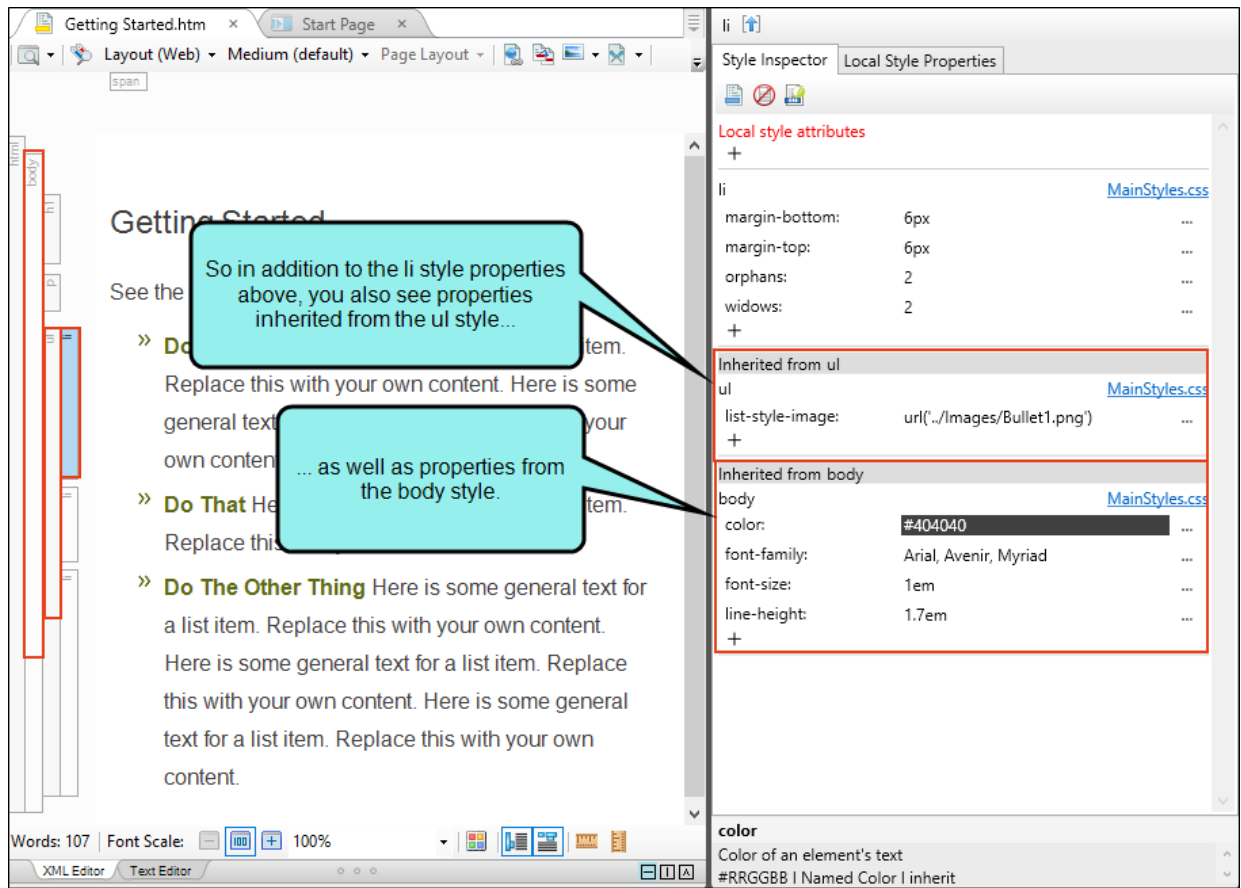


If stylesheets have been linked (see "Linking Stylesheets" on page 162), you might see properties that are coming from more than one stylesheet.



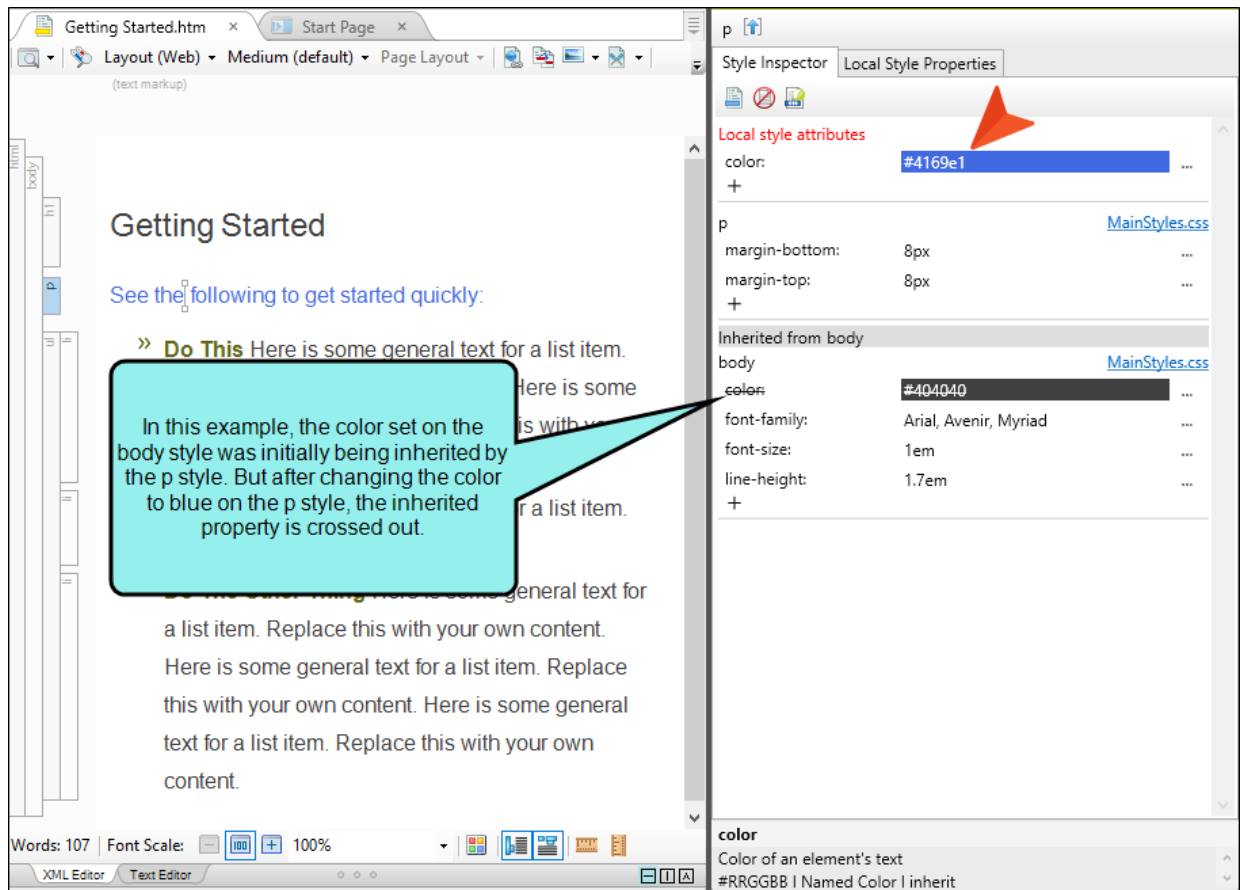
- **Inherited Properties** In addition to the style for the selected content, you will also see parent styles in the Style Inspector. These are tags that the current tag is placed within; therefore, the style for the selected content will inherit style settings from those parent styles, as well as using its own settings. Every tag you add to a content file (i.e., topic, snippet, template page) is found within the <body> tag, so at the very least, your style will be inheriting property values from the body style. But your tag might also appear within other tags, therefore inheriting from those styles as well. For more information, see "Inheritance" on page 64.







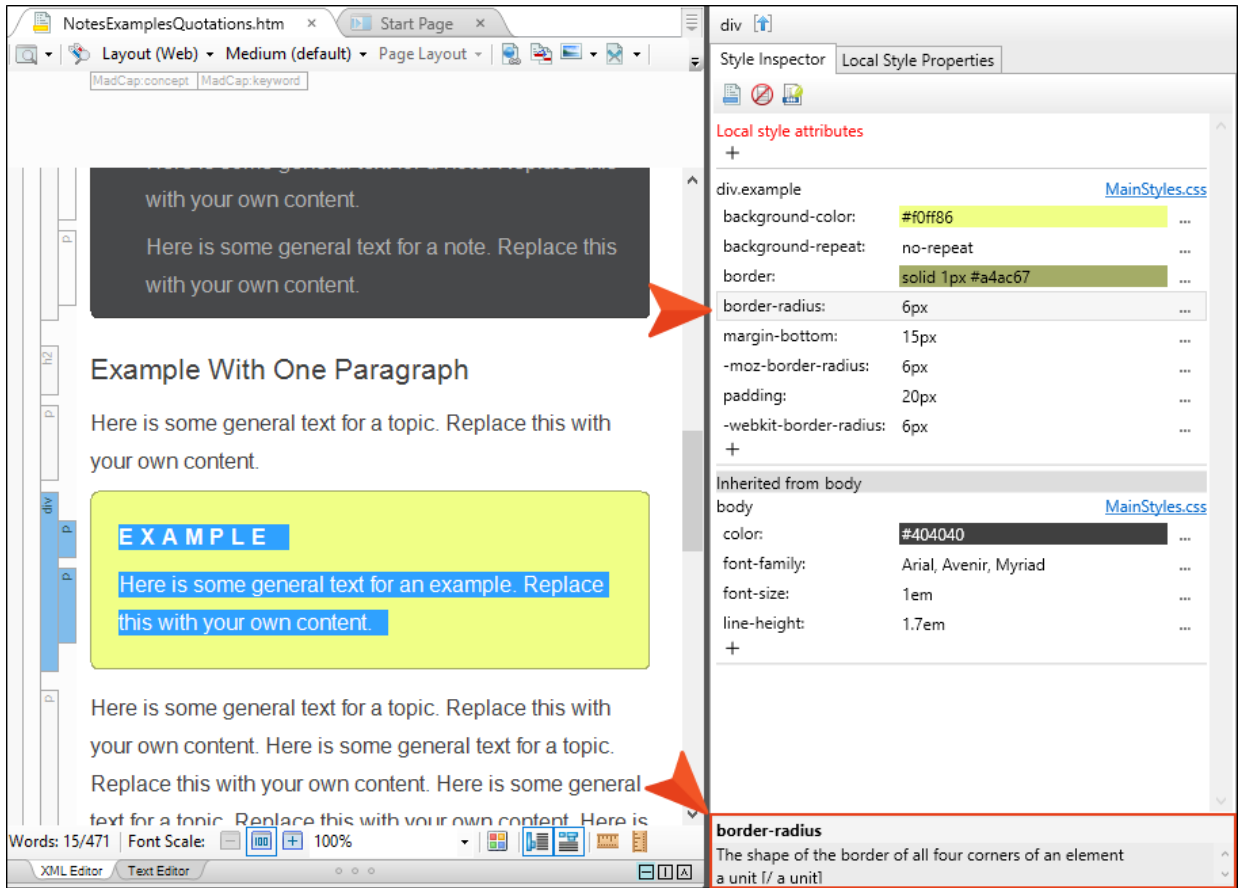
If you see a property that is crossed out, it means the value was being inherited but has been overridden in the child style.



If you see a property in gray, it means the property is inherited but is not currently being used by the content in the child tag.

The screenshot shows a web browser window with a document titled "ImageWithinList.htm". The main content area displays a list of two items, each with a heading and a paragraph of text. A callout box with a light blue background and a black border points to the list, containing the text: "In this example, a border was set on the ol style. But this border pertains to the entire container holding the list, and not to individual list items. Therefore, it is grayed out." To the right of the browser window is the "Style Inspector" panel, which shows the "Local Style Properties" for the selected "li" element. The "Local style attributes" section is empty. The "Inherited from ol" section shows a "border" property with a value of "solid 1px #808080", which is highlighted in gray. The "Inherited from body" section shows a "color" property with a value of "#404040", also highlighted in gray. The "color" section at the bottom of the panel shows the text "Color of an element's text" and "#RRGGBB | Named Color | inherit", with "#RRGGBB" highlighted in gray. The browser's status bar at the bottom shows "Words: 97", "Font Scale: 100%", and "100%".

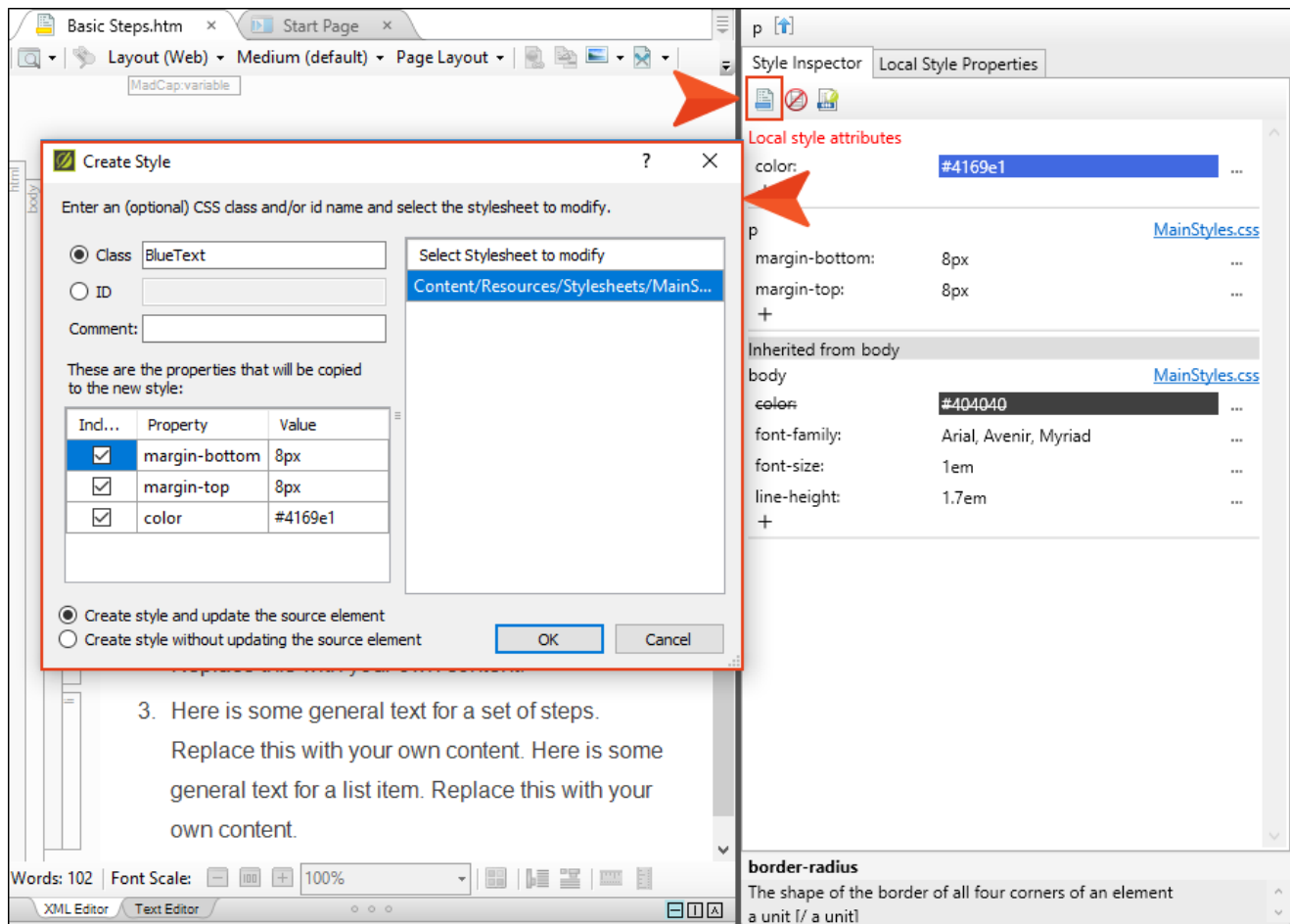
- **Property Description** Similar to the Stylesheet Editor, a description of each property is shown at the bottom when you click on it.



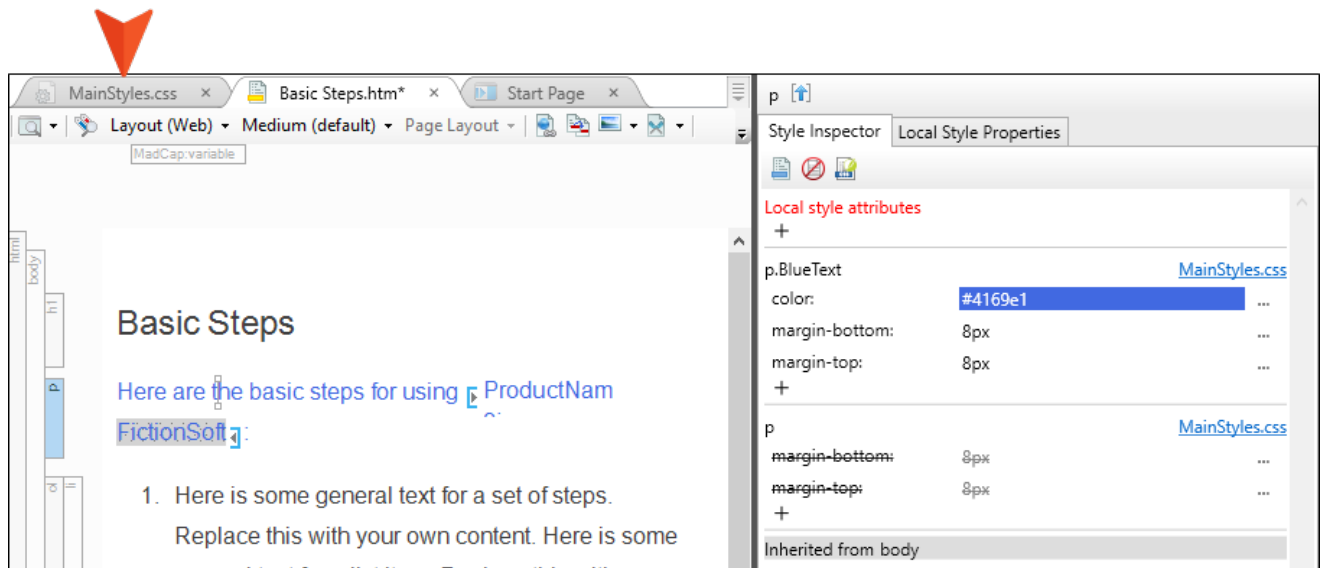
# Creating Style Classes

From the Style Inspector, you can create a new style class based on the properties of the selected style, as well as any local formatting that also might be applied to the content in the file.

When you do this, the Create Style dialog opens. From here, you can provide a name for the new class (or ID), add a comment, include or exclude any of the relevant property values, and choose whether you want to update the source document at the same time.

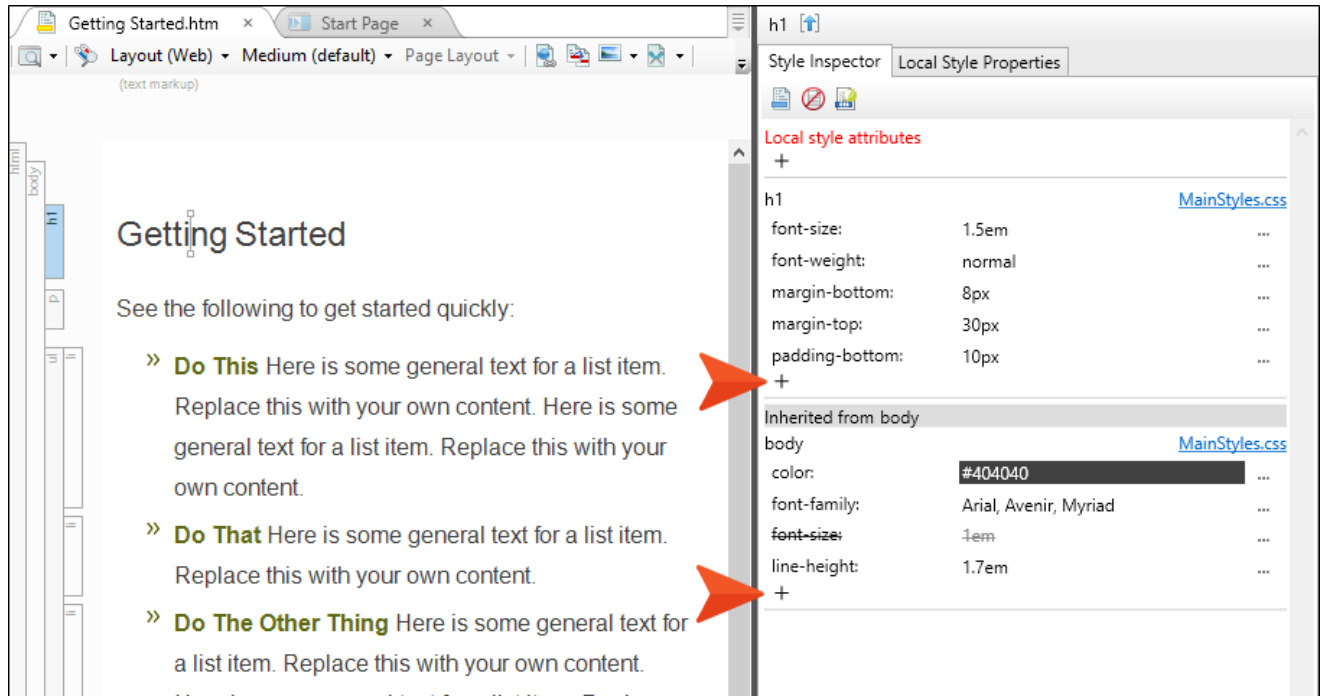


Also, when you are finished creating the new style class, the stylesheet is automatically opened (if it wasn't already). It is necessary to do this, because by creating the style class, you have made a change to the stylesheet.

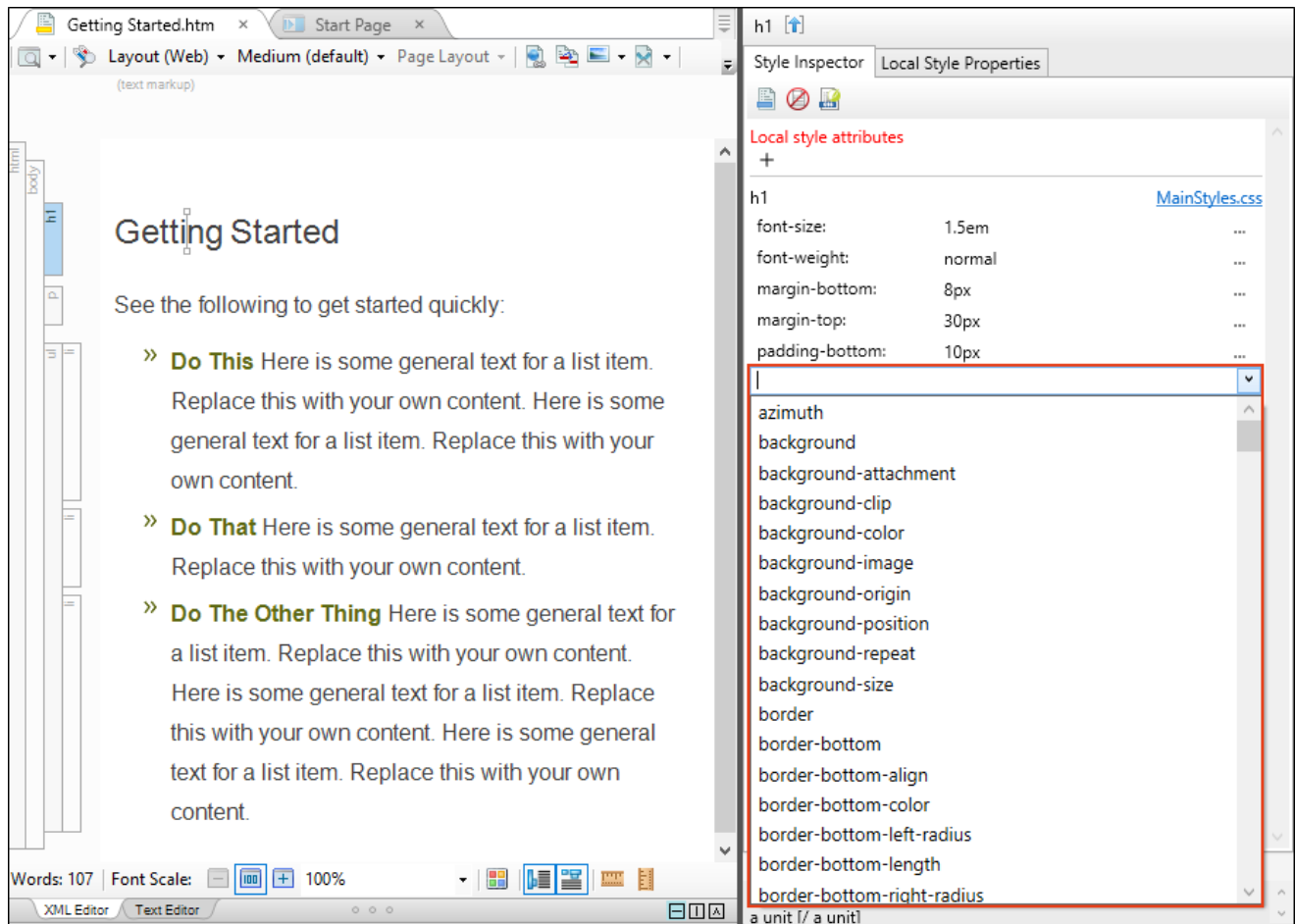


# Adding Properties to Styles

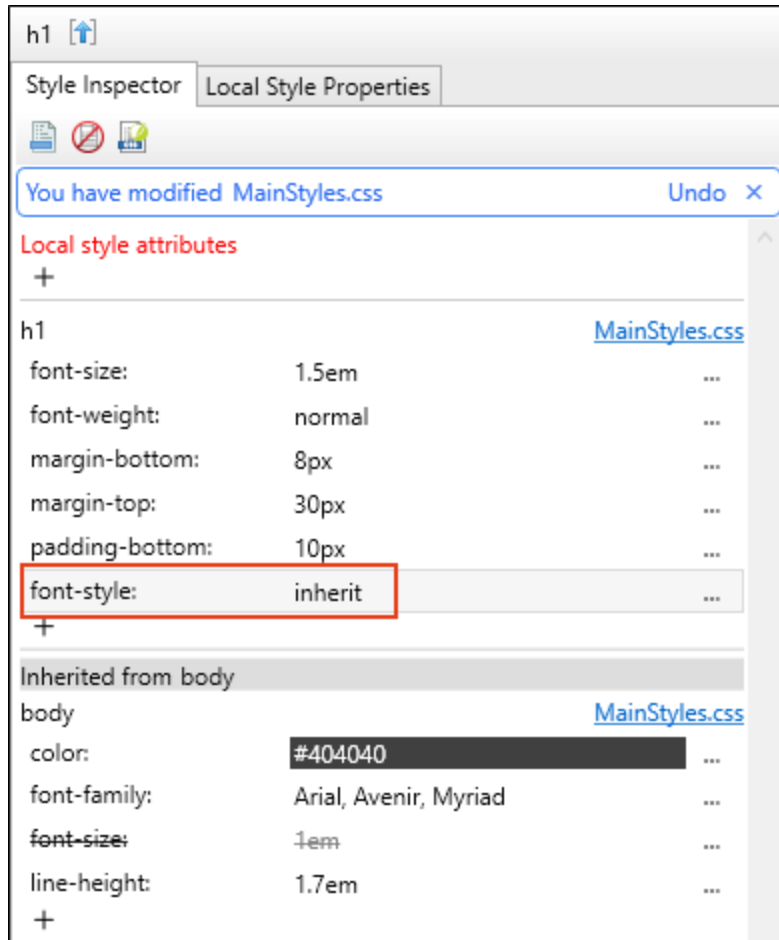
You can quickly and easily add properties to the styles shown in the Style Inspector by clicking the small plus sign at the bottom of the list of existing properties.



This opens a field at that location. You can either type the name of the new property in that field or select it from the drop-down.



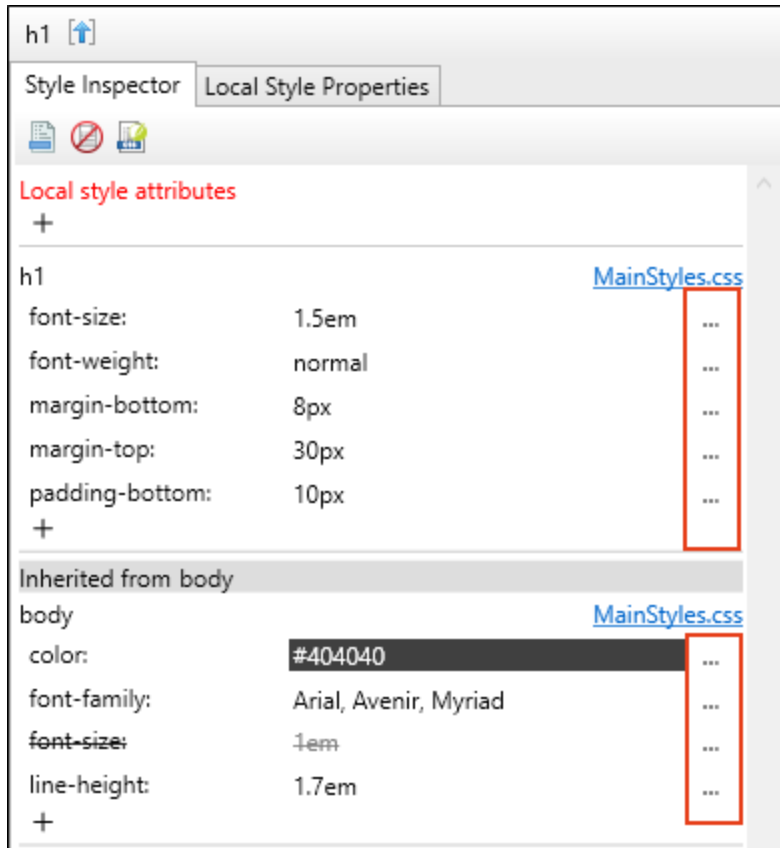
Until you edit a value for the property, it will show the value as inherited.





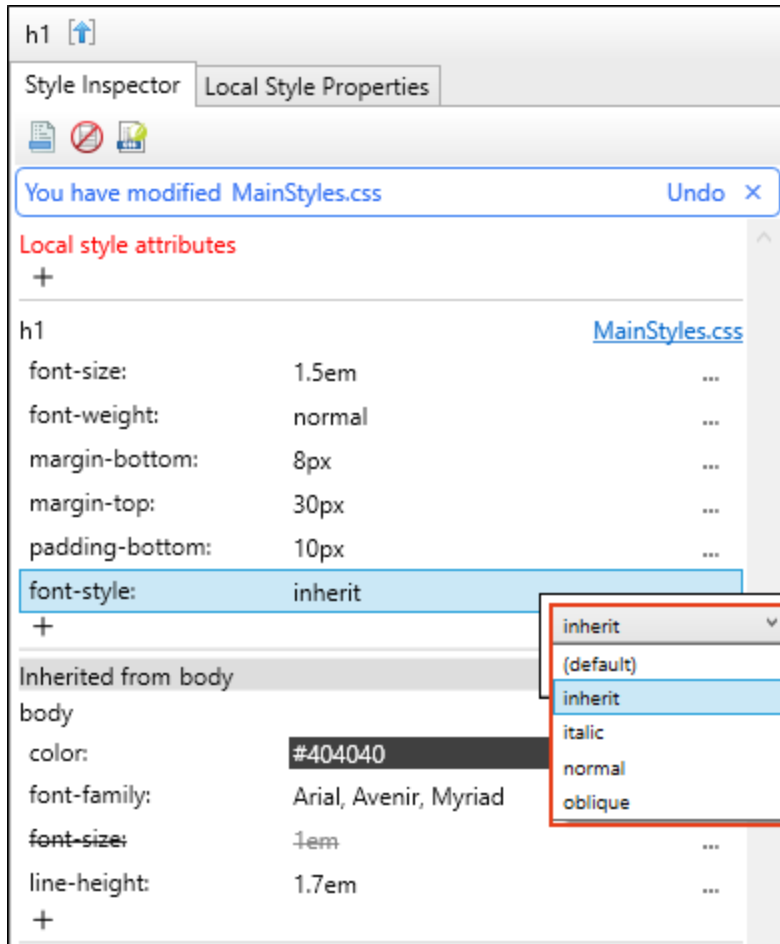
# Editing Property Values

You can change the values for most properties shown by clicking the ellipsis (...) to the right of the property.



For other properties, you can just click directly in the field and type the new value.

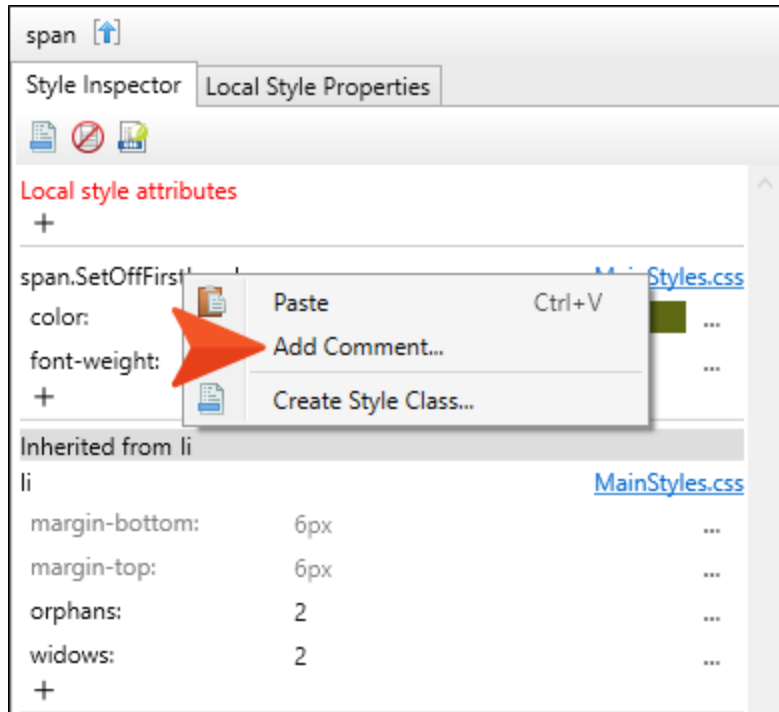
After this, enter the value in the popup. The type of popup depends on the kind of property that you are modifying.



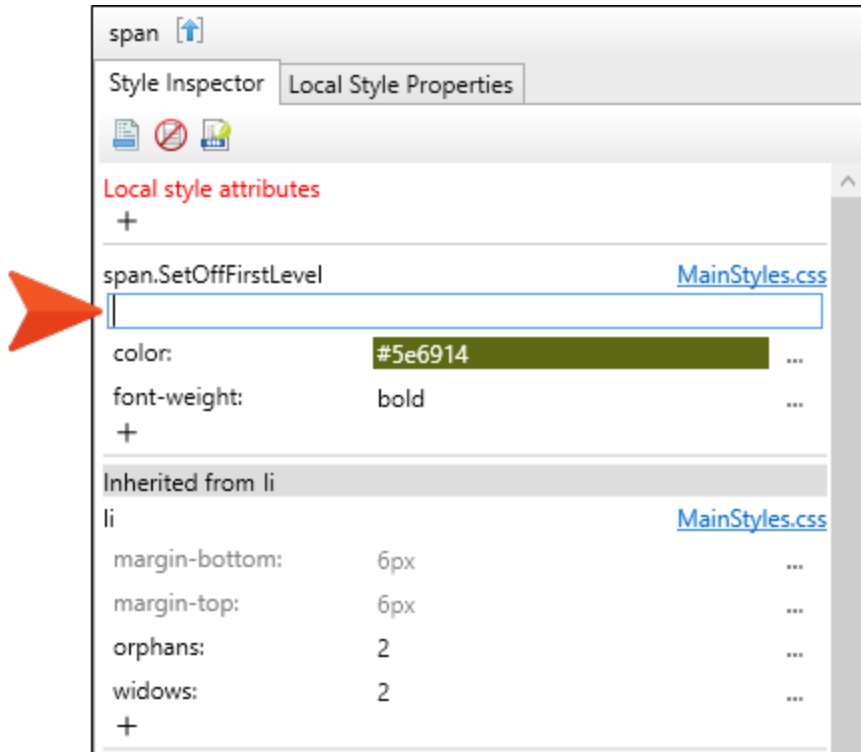
# Adding Comments to Styles

Just as you can add comments to styles in the regular stylesheet (see "Adding Comments to Styles" on page 176), you can also do this in Style Inspector.

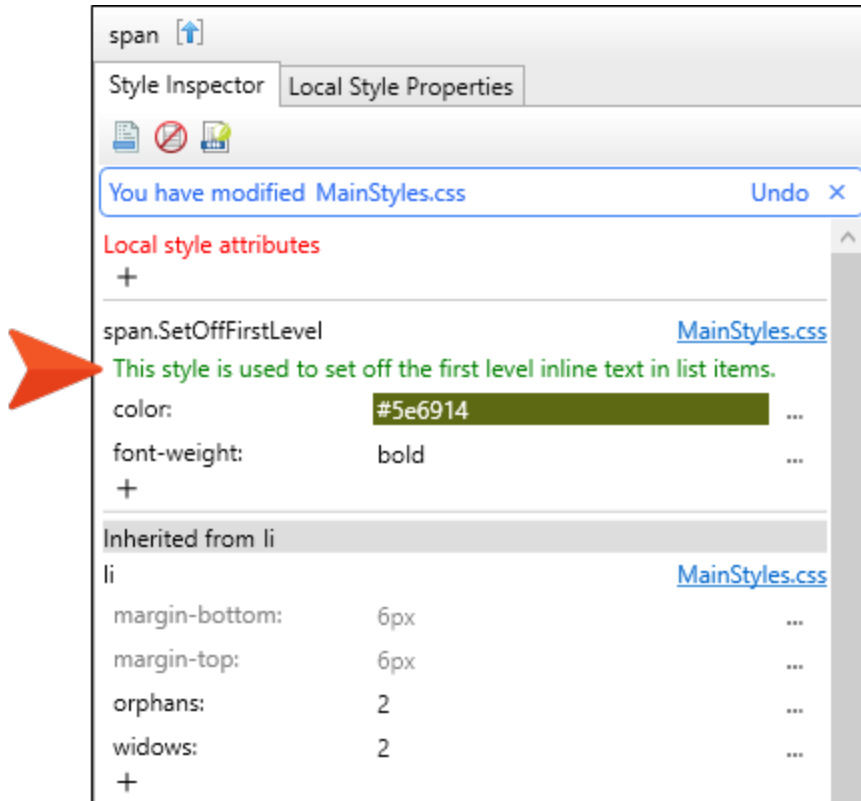
Start by right-clicking a style in the Style Inspector, then select **Add Comment**.



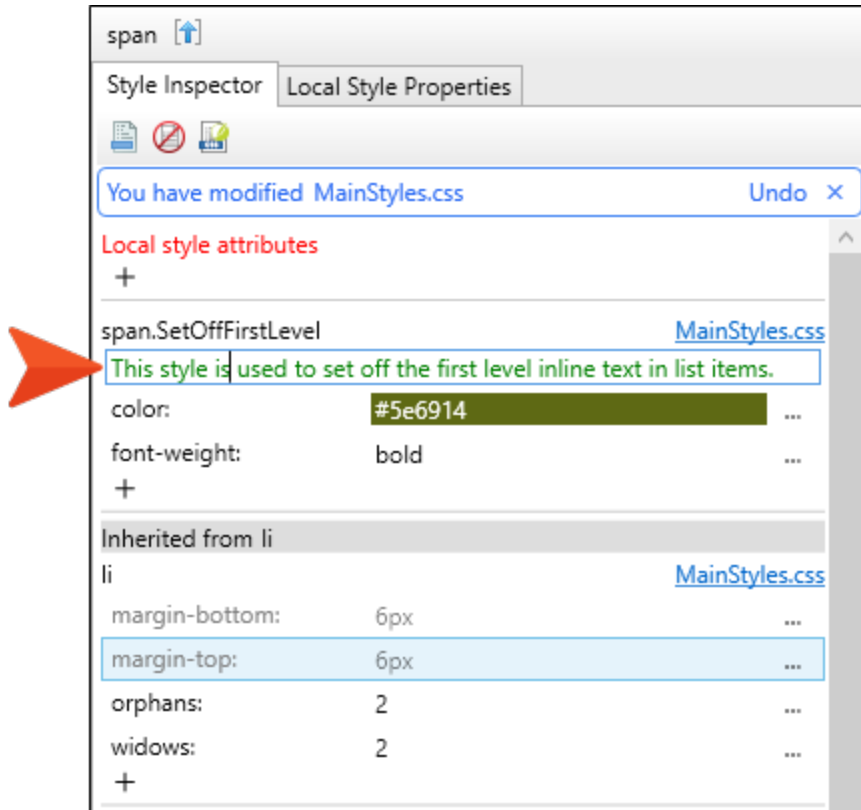
A text box displays below the style, where you can type the comment.



After you press ENTER, the comment appears in green text.

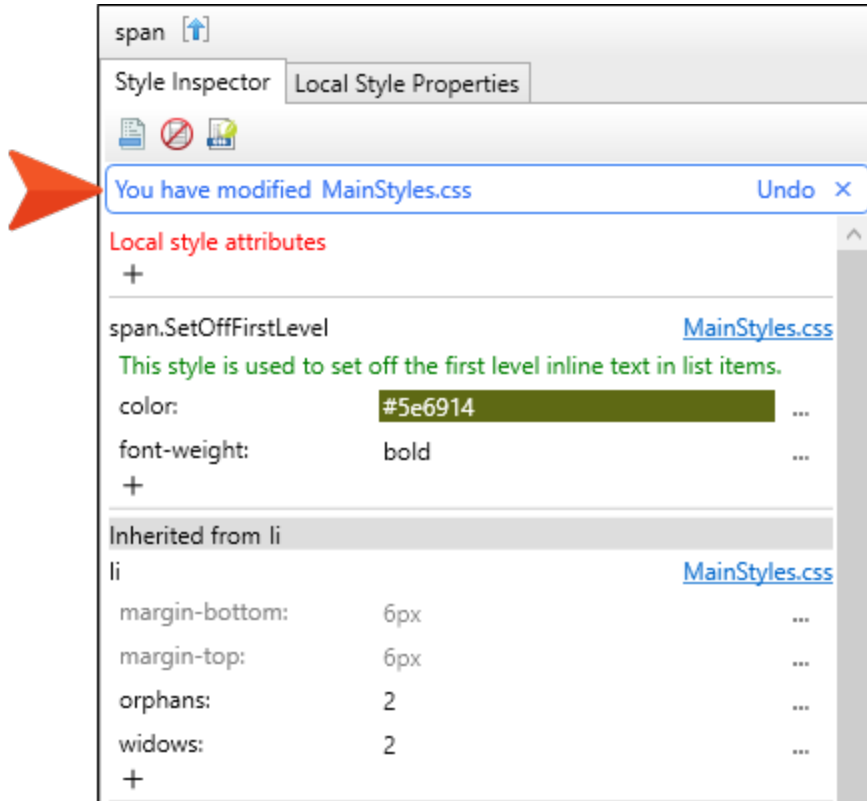



If you want to edit the comment, simply click on the comment and the text box will appear, allowing you to type in it.



# Modifying the Stylesheet—Undo or Dismiss

Whenever you make changes to styles (e.g., add properties, edit values) in the Style Inspector, you will see a message that the stylesheet has been edited. You can either undo your change or dismiss the message, in which case your edits will remain.




- ✓ **TIP** Whenever you make changes in the Style Inspector, the stylesheet is automatically opened as well, although the current content file remains the active file in the workspace. If you want to undo more than one change, first click **Undo** in the message in the Style Inspector. Then select the tab in the workspace to bring the Stylesheet Editor into focus, and click  in the Quick Access Toolbar (located in the upper-left of the Flare interface) for each change you want to undo.

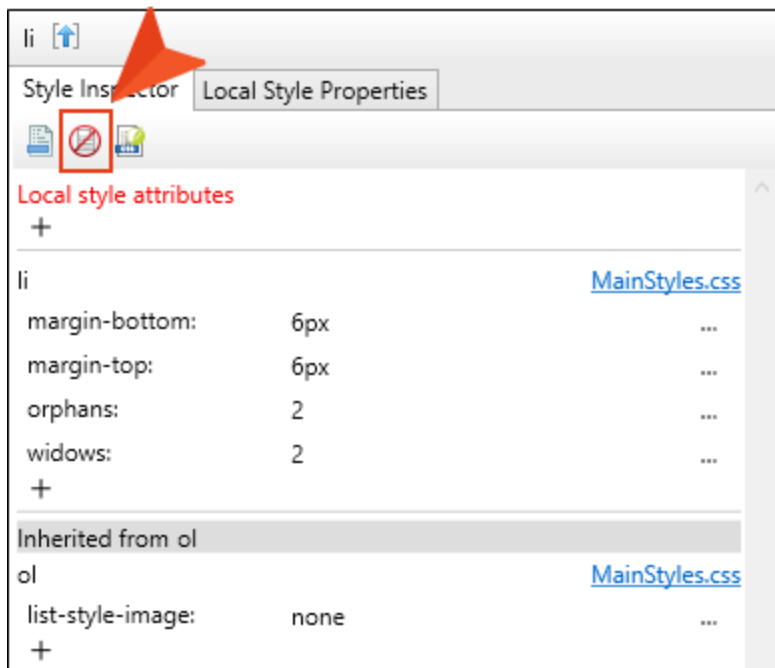
# Hiding Styles With Non-Inherited Properties

By their very nature, many style properties are considered “inherited,” while a significant number of others are “non-inherited.” This refers to how the properties act when tags are nested in a content file, such as a topic or snippet. Inherited properties for one style will trickle down to the content found in another tag within it. On the other hand, non-inherited properties will not be used by the content in the nested tags.


The Style Inspector shows both inherited and non-inherited properties by default. However, you can select an option that hides styles and their properties. This option works under either of the following conditions:


- A style contains only non-inherited properties.
- A style is empty (i.e., does not have any properties).

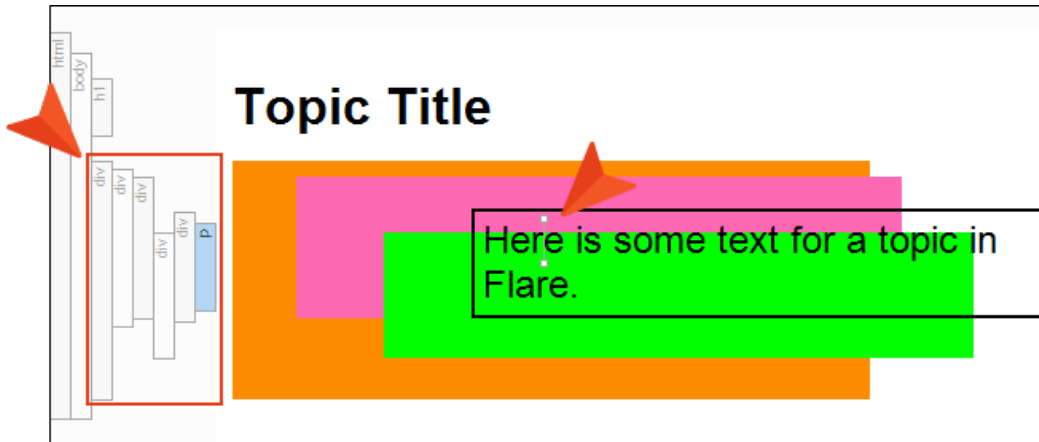
To hide such styles and properties, click  in the local toolbar. Click it again to show the styles and properties.





 **NOTE** If a style has non-inherited properties within it, but also has at least one inherited property, this option will not hide any of the properties. They will all be shown whether you select the “hide” option or not.

 **EXAMPLE** Consider the following topic. Its wild design is not really anything someone would create in a real-world situation, but it does help to illustrate this feature. Notice that the cursor is located on a paragraph, whose tag is nested within five different <div> tags.

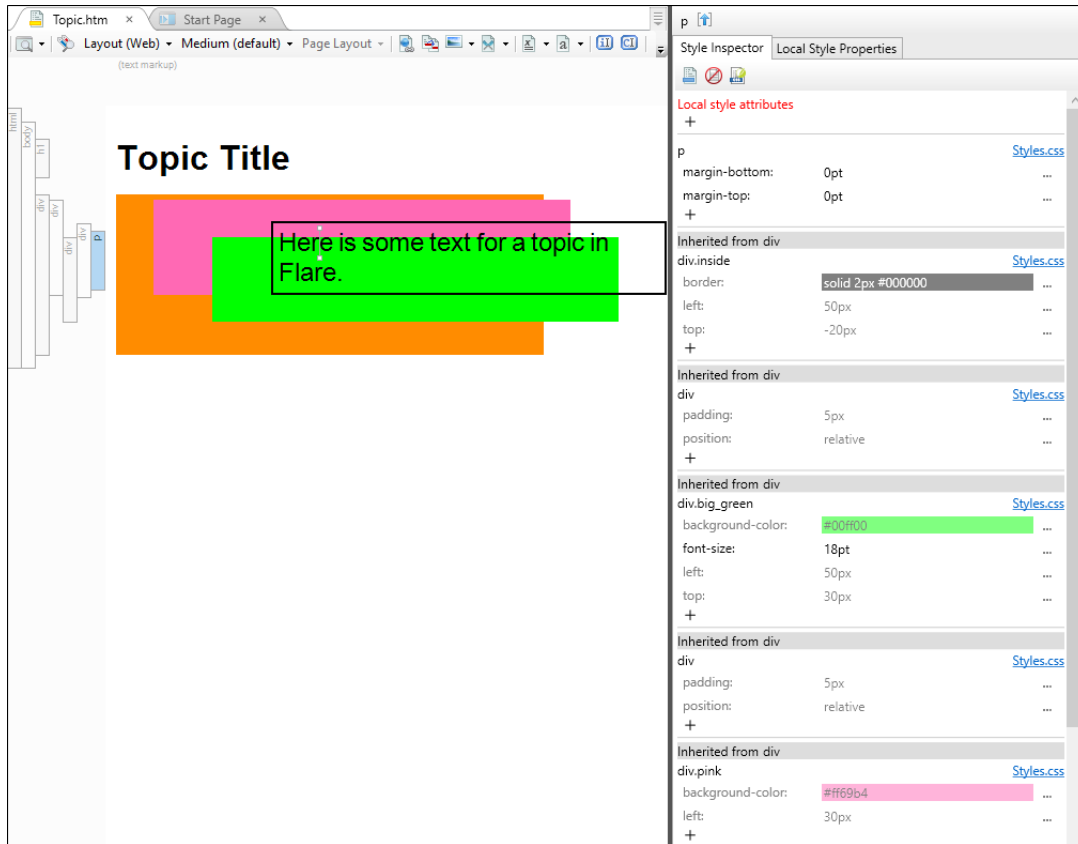


The markup looks like this:

```
<h1>Topic Title</h1>
<div class="orange">
  <div class="misc">
    <div class="pink">
      <div class="big_green">
        <div class="inside">
          <p>Here is some text for a topic in Flare.</p>
        </div>
      </div>
    </div>
  </div>
</div>
</div>
```

Due to the properties set on these div classes, you can see a few different background colors, a border, and the font values for the text.

☆ After opening the Style Inspector, you will initially see all of these styles and their properties.

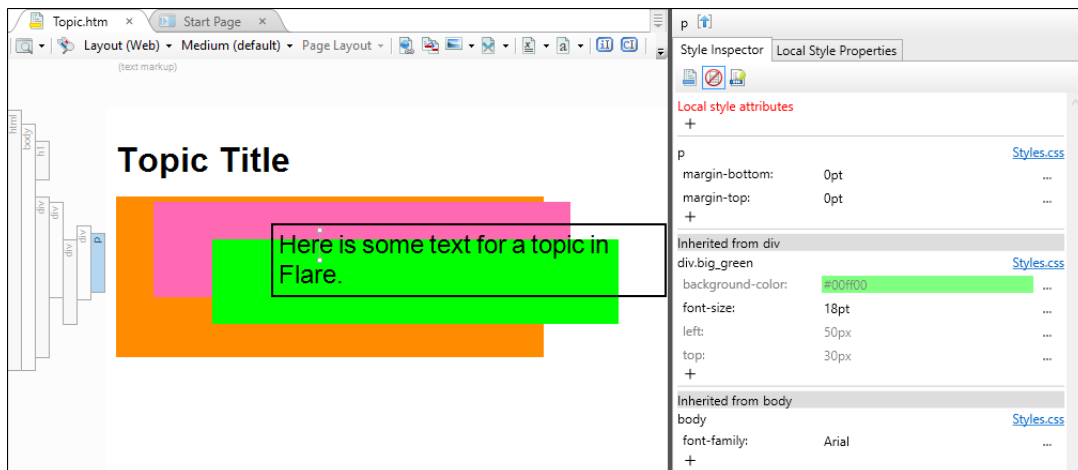


Notice that many of these properties are gray. These are non-inherited properties, which means that they will not affect the paragraph directly.

For example, the background color property is non-inherited. That is why none of the colors are actually applied to the paragraph. At first glance, they might appear to be applied to the paragraph to a certain extent, but they are actually associated with the <div> tags around the paragraph. And the positions of these various <div> tags have been staggered to show this.

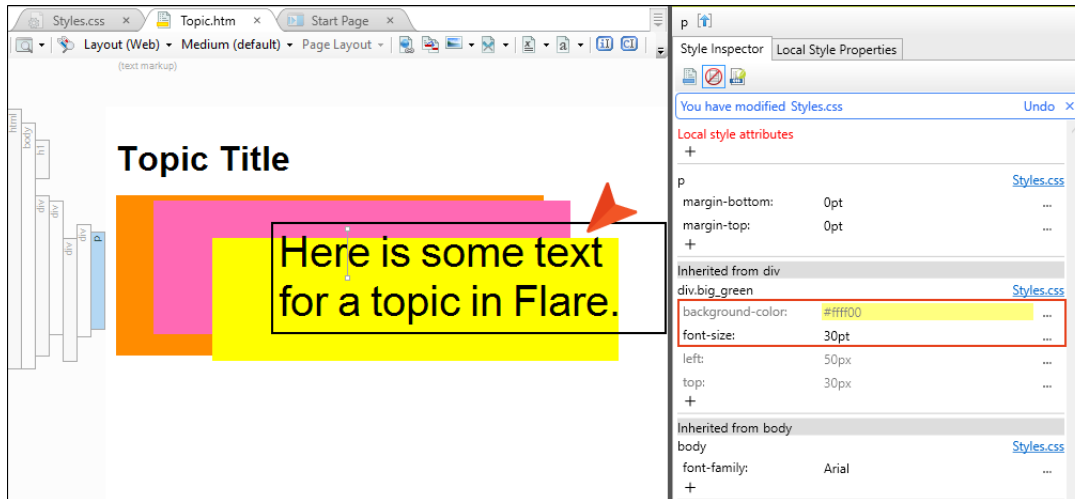
☆ The border is another example. While the border might appear to be applied to the paragraph, it is actually the <div> tag outside of the paragraph where the border actually resides.

If you select the option to hide the non-inherited styles and properties, you will see this:

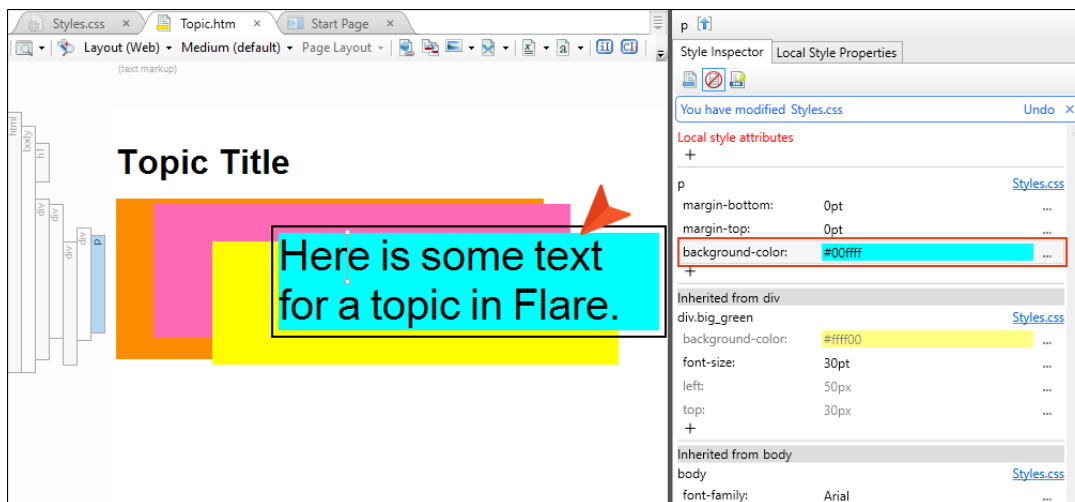


Most of the styles are now hidden, because they contained only non-inherited properties. However, notice that `div.big_green` style class is still shown, along with its properties, some of which are non-inherited (they are gray). The reason this style remains in view is that it has a specified font size, which is an inherited property. This prevents the style and properties (even non-inherited) from being hidden in the Style Inspector.

- ☆ If you were to change the green background to yellow, the paragraph would not really be affected. But if you increase the font size to 30 pt, the paragraph will change accordingly (because you've edited an inherited property).




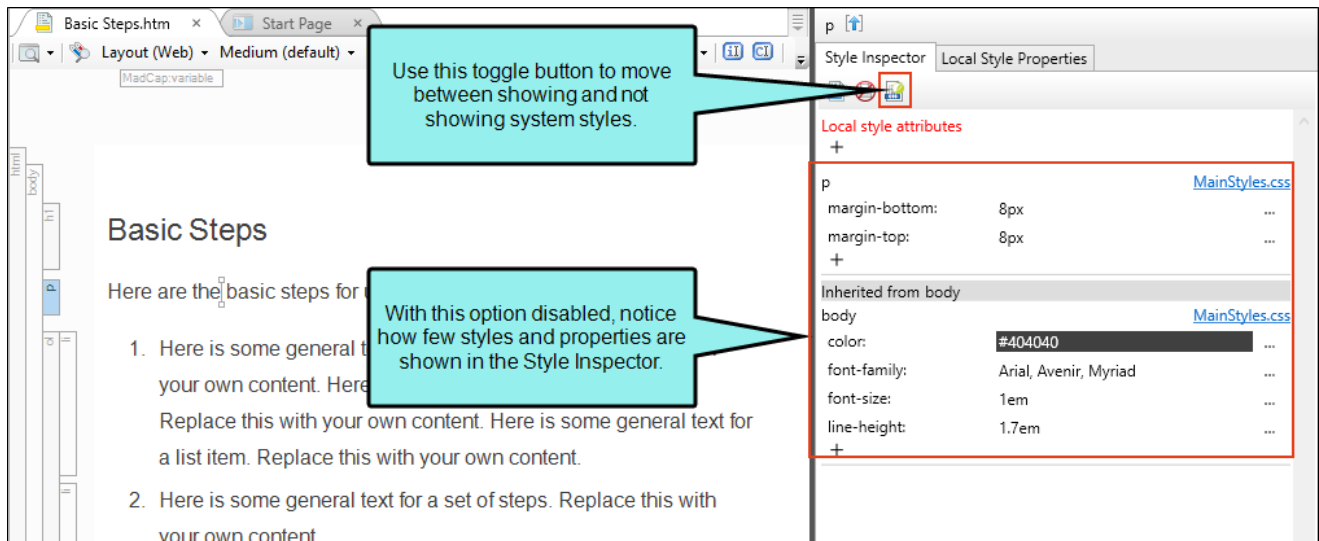
If you wanted to set the background color for the paragraph itself (let's say, to light blue), you would need to do this on the paragraph style.



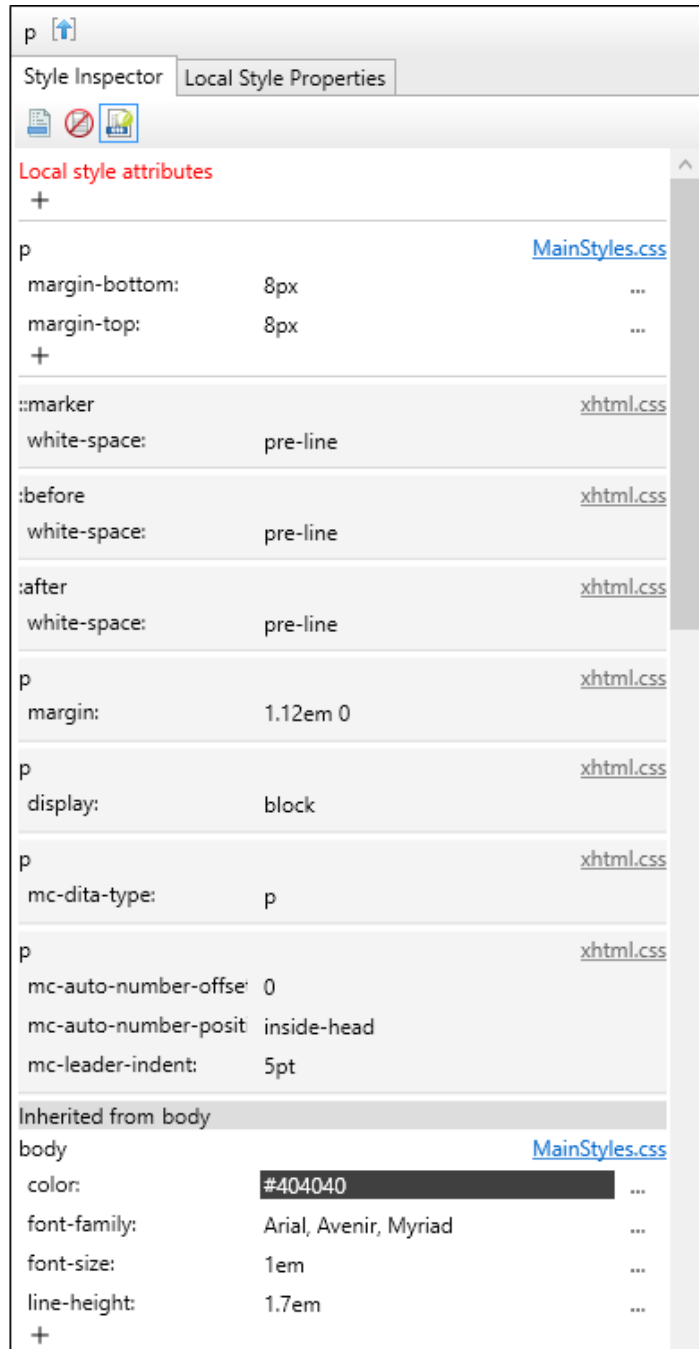
# Including System Styles

By default, the Style Inspector shows styles and properties from your local stylesheet(s). However, there might also be many other styles and properties at work in the file that you have open; these other styles and properties are inherited from system (or “factory”) stylesheets that are located where you installed Flare. See “Inheritance” on page 64.

If you want to see these other styles and properties, click  in the local toolbar.



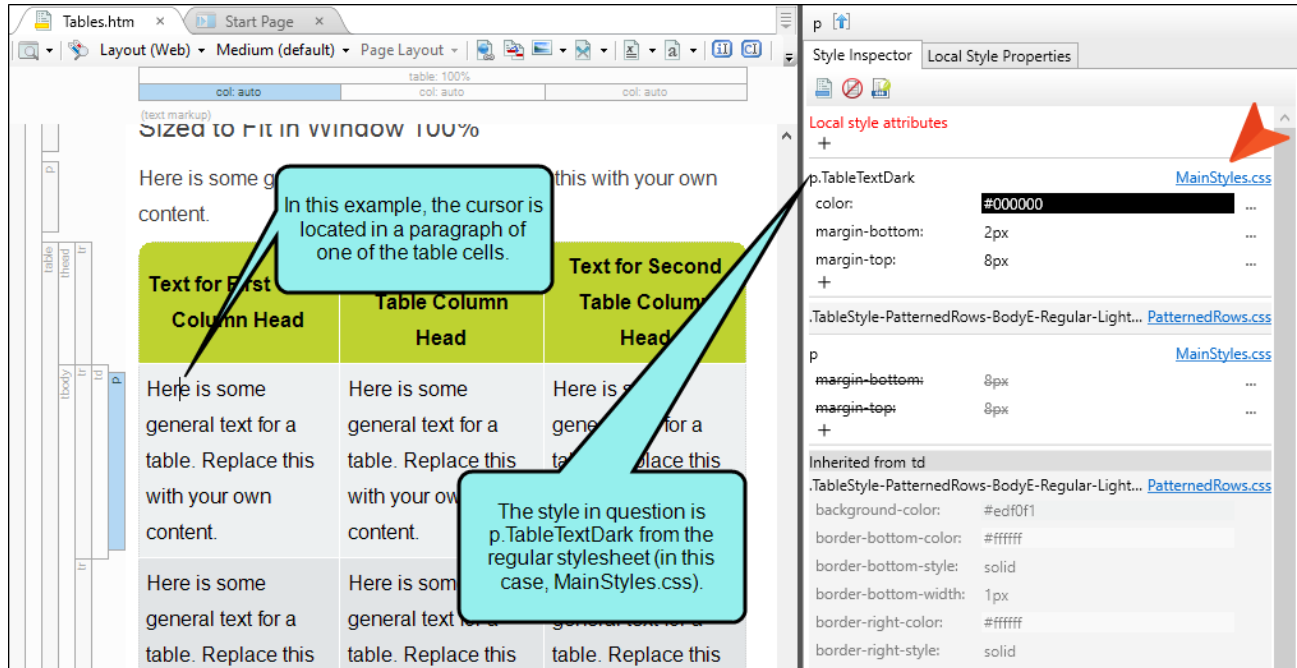
With the option enabled, notice how many more styles and properties are shown. Those with a gray background are coming from system stylesheets.



 **NOTE** Although you can view styles and properties from system stylesheets, you cannot edit them in the Style Inspector.

# Table Stylesheets in the Style Inspector

If you place your cursor in a table that is using a special table stylesheet, the appropriate style from the regular stylesheet is featured in the Style Inspector. Although styles and properties from the table stylesheet are also displayed, they are treated much like those from factory stylesheets. In other words, the background is gray and you cannot edit those properties. Instead, you should open the table stylesheet separately and edit its formatting within it.



Styles and properties are also shown from the related table stylesheet (in this case, PatternedRows.css).

The screenshot shows the Style Inspector interface with the following content:

- Local style attributes:** A plus sign (+) indicating expandable local styles.
- p.TableTextDark (MainStyles.css):**
  - color: #000000
  - margin-bottom: 2px
  - margin-top: 8px
- .TableStyle-PatternedRows-BodyE-Regular-Light... (PatternedRows.css):** A plus sign (+) indicating expandable styles.
- p (MainStyles.css):**
  - margin-bottom: 8px
  - margin-top: 8px
- Inherited from td (.TableStyle-PatternedRows-BodyE-Regular-Light... PatternedRows.css):** A red box highlights this section, which includes:
  - background-color: #edf0f1
  - border-bottom-color: #ffffff
  - border-bottom-style: solid
  - border-bottom-width: 1px
  - border-right-color: #ffffff
  - border-right-style: solid
  - border-right-width: 1px
  - padding-bottom: 5px
  - padding-left: 5px
  - padding-right: 5px
  - padding-top: 5px
- Inherited from td (MainStyles.css):**
  - margin: 8px



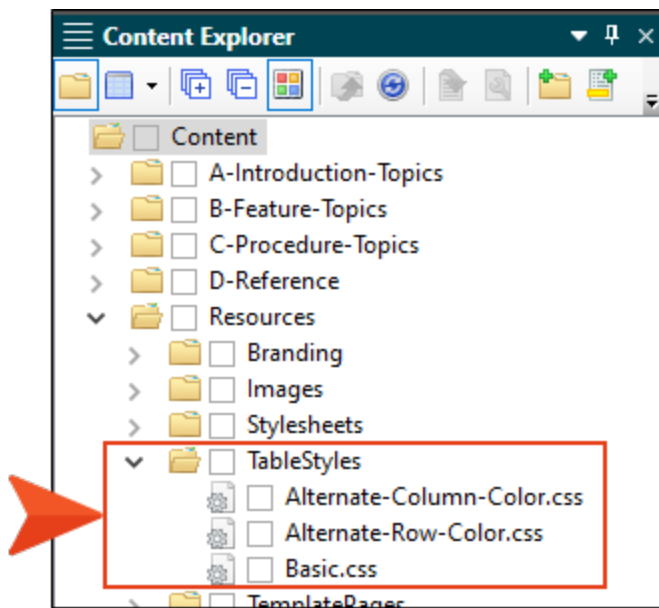
# I Style Reports

When working with projects in Flare, you can generate reports about styles. See the online Help for details.

- Viewing Duplicate Styles
- Viewing Local Style Suggestions
- Viewing New Style Suggestions
- Viewing Undefined Styles
- Viewing Unused Content Files
- Viewing Unused Styles

# Table Stylesheets

A table stylesheet lets you create a look that can be applied to several tables throughout your project. To change the look of general topic content, you can use a regular stylesheet. If you need to change branding for your project, you can use a branding stylesheet. In fact, using these stylesheets are other ways to control the look of tables.



You can create as many table stylesheets as you need. The recommended location to store a table stylesheet in the Content Explorer is in the Resources > TableStyles folder. However, you can store it anywhere in the Content Explorer that you like.

This chapter discusses the following:

Creating Table Stylesheets .....	272
Setting Table Styles for Print Output .....	275
Editing Table Stylesheets .....	279
Applying Table Stylesheets to Tables .....	294

# Creating Table Stylesheets

You can add a stylesheet to be used specifically for tables. The traditional location to store a table stylesheet in the Content Explorer is in the Resources > TableStyles folder. However, you can store it anywhere in the Content Explorer that you like.


## How to Create a Table Stylesheet


1. Do one of the following, depending on the part of the user interface you are using:
  - **(Recommended) Right-Click** In the Content Explorer, right-click on a folder and from the context menu select **New > Table Style**.


✓ **TIP** When adding a new file to the Content Explorer, the recommended method is to right-click on the folder in the Content Explorer and use the **New** menu option. This is the most efficient way to direct the new file to the folder where you want to store it. That's because the Add File dialog opens when you add a new content file, and this method ensures that the folder you want is already selected in that dialog.

- **Ribbon** Select **Project > New > Table Style**.



The Add File dialog opens.

2. In the **File Type** field at the top, make sure **Table Style** is selected.
3. In the **Source** area, choose to create the new file based on a template or an existing file.
  - **New From Template** Choose either a factory template file or one of your own custom template files as a starting point. The new file will take on all of the settings contained in the template. If you want to use the factory template provided by Flare, expand the **Factory Templates** folder and click on a template file. If you want to use your own custom template file, expand the appropriate folder and click on a file. For more information about templates, see the online Help.
  - **New From Existing** Choose an existing file of the same type as a starting point for your new file. As with template files, your new file will take on all of the settings contained in the file you select. To use this option, click , use the Open File dialog to find a file, and double-click it.


- (Optional) The **Folder** field is automatically populated with the folder that has focus in the Content Explorer. If you want to place the file into a folder that you previously created in the Content Explorer, in the **Folder** field click  and select the subfolder. Otherwise, keep the default location.

 **NOTE** If you want to place non-topic files in a recommended folder, first make sure that folder exists in the Content Explorer. If it does not exist, you can easily add it.

Non-Topic File Type	Recommended Default Folder in Content Explorer
Branding	Resources > Branding
Image	Resources > Images
Micro Content	Resources > MicroContent
Multimedia	Resources > Multimedia
Page Layout	Resources > PageLayouts
Snippet	Resources > Snippets
Stylesheet	Resources > Stylesheets
Table Stylesheet	Resources > TableStyles
Template Page	Resources > TemplatePages

- In the **File Name** field, type a new name for the stylesheet.
- (Optional) If you want to apply condition tags to the file, expand the **Attributes** section at the bottom of the dialog. Next to the **Condition Tags** field, click  and select the conditions you want to apply. Click **OK**.
- (Optional) If you want to apply file tags, expand the **Attributes** section at the bottom of the dialog. Next to the **File Tags** field, click  and select the file tags you want to apply. Click **OK**.
- Click **Add**. The table stylesheet is added and opens in its own page in the Table Style Editor.





**NOTE** You can create a new table stylesheet while inserting a table into a topic. In the Insert Table dialog, click face of the **Create New Table Style** button  (not the down arrow) and complete the options in the Select Table Style Template dialog. After you insert the table, the new stylesheet is added to your project.

# I Setting Table Styles for Print Output

A table stylesheet lets you single-source your formatting by setting the properties in one place and reusing them wherever you insert tables. But what if you want the tables in online output to look one way and the tables in your printed output to look another way? Here are two options... **Solution #1—Two mediums:** This is recommended. You can have one table stylesheet and use a medium to specify different settings for it—one medium is used for online output and another for print. **Solution #2—Two table stylesheets:** You can insert a single table at each location, using a special version of the table style for print-based output. This solution requires two table stylesheets—one for online and one for print.

# How to Create a Table Style for Print-Based Output—Two Mediums


1. From the Resources\TableStyles subfolder in the Content Explorer, open the table stylesheet.
2. Set the properties to be used for the online output.
  - a. In the local toolbar of the Table Style Editor, click in the **Medium** field and make sure the medium for the your online output is selected.
  - b. Use the various tabs in the editor to set properties for that medium.
3. Set the properties to be used for the printed output as follows:
  - a. In the local toolbar of the Table Style Editor, click the down arrow in the **Medium** field and select **Medium: print** or **Medium: [name of print medium]**.
  - b. Use the various tabs in the editor to set properties for that medium.
  - c. Click  to save your work.
4. Apply that table style to the appropriate tables throughout your project as follows:



See "Applying Table Stylesheets to Tables" on page 294.
5. Associate the online medium with your online target as follows:
  - a. Open the target to be used for online output (based on either the HTML5, Clean XHTML, Eclipse Help, Microsoft HTML Help, WebHelp, WebHelp Plus format).
  - b. In the Target Editor, select the **Advanced** tab.
  - c. In the **Stylesheet Medium** section, select the medium that you used for online output.
6. Associate the print medium with your print target as follows:
  - a. Open the target to be used for print output (Adobe PDF, Microsoft Word).
  - b. In the Target Editor, select the **Advanced** tab.
  - c. In the **Stylesheet Medium** section, select the medium that you used for printed output (e.g., print).
7. Click  to save all files.



# How to Create a Table Style for Print-Based Output—Two Table Stylesheets

1. Create one table stylesheet to be used for online output and another to be used for printed output.

 **TIP** If you want both tables to share most of the same settings, you can create the online table stylesheet first, make a copy of it for the print version, and then edit the settings in the copy as necessary. You can easily do this by selecting the original table stylesheet in the Content Explorer (Resources\TableStyles subfolder), pressing **CTRL+C**, pressing **CTRL+V**, and renaming the copy to reflect your needs.

2. Insert a table into a topic or edit an existing table.
3. In the Insert Table dialog (if inserting a new table) or the Table Properties dialog (if editing an existing table), click **Table Style** and from the drop-down select the table style to be used for the online output.
4. Click the down arrow next to the **Create Table Style** button  and select **Print Style**. Click **OK** in the small dialog that opens. The Select Table Style dialog opens.
5. From the list, select the table style to be used for the printed output.
6. Click **OK**.
7. In the Insert Table or Table Properties dialog, click **OK**.
8. Click  to save all files.

☆ **EXAMPLE** You create a table stylesheet with a pattern design that displays the table with alternating green rows. The problem is that for printed output, you need the rows to display in light gray.

Suppose you decide to use the recommended solution (mediums). Let's say the target for online output is called Target A, and the one for print output is called Target B. The first step is to make sure you have two mediums. Suppose Target A is using the "default" medium, where the rows are set to use a green background. With the properties for Target A already set, you now need to specify style properties for Target B. Therefore, in the Table Style Editor, instead of selecting the "default" medium, you can select another medium (e.g., the "print" medium) and change the properties for the rows to light gray. It's the same table stylesheet and the same pattern that you are working with. The only difference is that one medium is telling Flare to display that table rows with a green background, and the other medium is telling it to use light gray. With Target A using the default medium and Target B using the other medium, the tables will display appropriately in each output.

If you decide instead to use the other solution (two table stylesheets), you first create a table stylesheet and specify settings in it appropriate for online output (e.g., green background for table rows). Then you create a secondary stylesheet. This extra table stylesheet will have design settings that are appropriate for printed output (e.g., light gray table rows). When you insert the table into your content (or edit an existing table), you select the original online table style and also select the special print version of the table style. If you generate any targets based on an online format (HTML5, Clean XHTML, Eclipse Help, Microsoft HTML Help, WebHelp, WebHelp Plus), the end user will see green rows in the table. However, if you generate any targets based on a print format (Adobe PDF, Microsoft Word), the end user will see light gray rows in the table.

📄 **NOTE** If you used print table styles in the past and want to remove them now in favor of the medium method, you can use an option in the Apply Table Style dialog in the Table Style Editor. When this option is enabled, print table styles will be removed from any tables updated by this dialog.

# I Editing Table Stylesheets

You can modify the look and feel of multiple tables at once by editing the properties in a custom table stylesheet. These table stylesheets let you easily and quickly create patterns and different looks for tables.

## How to Edit a Table Stylesheet

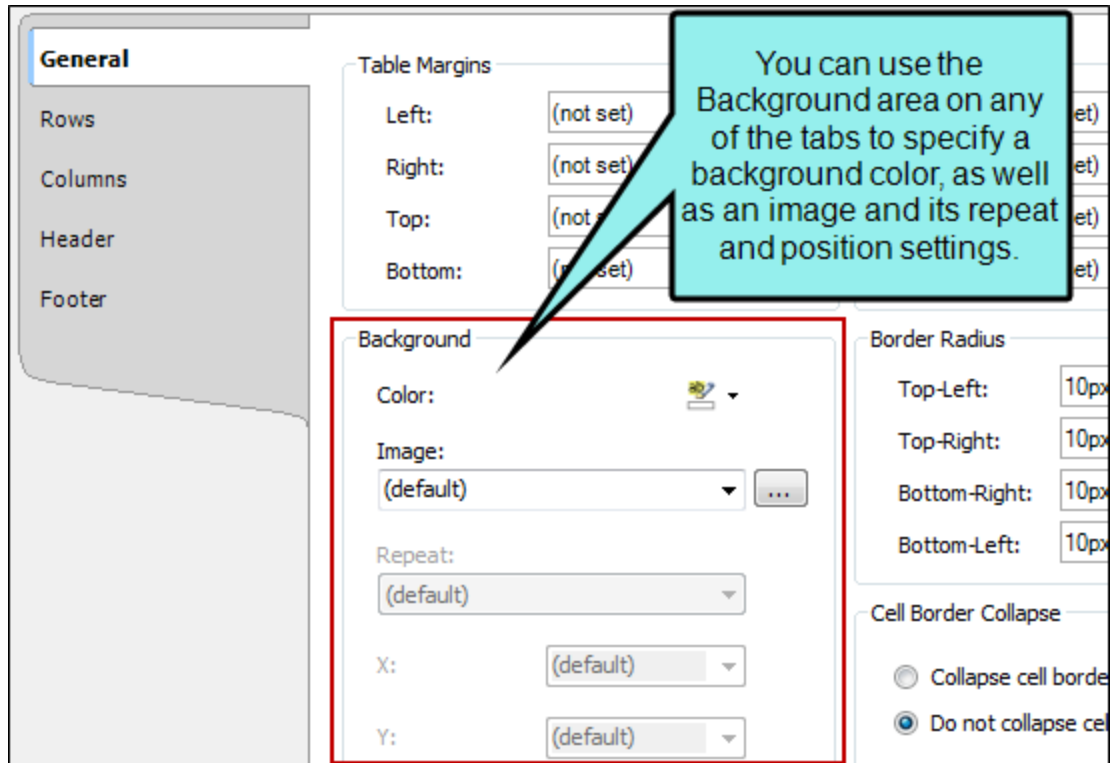
1. Open the table stylesheet that you want to edit. This is usually found in the **Resources > TableStyles** subfolder in the Content Explorer, but you can store table stylesheets in other folders in the Content Explorer if you like.
2. In the Table Style Editor, set the options for the table style on the tabs available.

### GENERAL TAB

This tab lets you set border, padding, margin, page layout breaks, and background properties for the entire table style.

- **Table Margins** Click in any of the individual fields (**Left, Right, Top, Bottom**) to specify the settings for the table margins (the amount of space around the table). In the left side of the field, enter a number for the amount of padding. In the right side of the field, select a unit of measurement (e.g., point, pixel, centimeter) for the number you entered. If you click the down arrow to the right of all the fields, the settings will be applied to all of the table margin fields. When you click that down arrow, a small popup displays. Use the lower-left area of the popup to enter a number for the amount of margin. Use the lower-right area to select a unit of measurement.

- **Background** Use this area to specify the settings that you want for the table background.



In the **Color** field, click the down arrow and select a color from the popup. For advanced color options, select **More Colors** and use the fields in the Color Picker dialog. In the Color Picker you can select a CSS variable for the background. Next to the **Image** field, click . Select an image file to insert and click **OK**.

If you want the background image to repeat, select one of the options from the **Repeat** field. You can also set the image position horizontally and vertically by using the **X** and **Y** fields.

- **Cell Padding** Click in any of the individual fields (**Left, Right, Top, Bottom**) to specify the settings for the cell padding (the amount of space between the edge of the table cell and the content in the cell). In the left side of the field, enter a number for the amount of padding. In the right side of the field, select a unit of measurement (e.g., point, pixel, centimeter) for the number you entered. If you click the down arrow to the right of all the fields, the settings will be applied to all of the cell padding fields. When you click that down arrow, a small popup displays. Use the lower-left area of the popup to enter a number for the amount of padding. Use the lower-right area to select a unit of measurement.
- **Outer Borders** Click in any of the individual fields (**Left, Right, Top, Bottom**) to specify the settings for the table border in the stylesheet. If you click the down arrow to the right of all the fields, the settings will be applied to all of the border fields. When you click that down arrow or in one of the individual fields, a small popup displays. Use the lower-left area of the popup to enter a number for the thickness of the border. Use the lower-middle area to select a unit of measurement (e.g., point, pixel, centimeter) for the number you entered. Use the upper-right area to select a color for the border. Note that in the Color Picker you can select a CSS variable. And use the lower-right area to select a line type (e.g., solid, double, dashed) for the border. When you are finished, click **OK** in the small popup.
- **Border Radius** These fields let you create rounded corners on the table. Click in any of the individual fields (**Top-Left, Top-Right, Bottom-Right, Bottom-Left**) to specify the settings for a particular corner of the table. If you click the down arrow to the right of all the fields, the settings will be applied to all of the fields. When you click that down arrow or in one of the individual fields, a small popup displays. This popup has two halves. You can complete only the left side of the popup if you like. This will create a curve that is equal horizontally and vertically. If you want a border to have more of a curve either horizontally or vertically, you can complete the fields in the right half of the popup as well, so that you have two values (e.g., 10px 15px) instead of one. For more information on using two sets of border radius properties, see [css3.info/preview/rounded-border/](https://css3.info/preview/rounded-border/). Use the lower-left area of the popup to enter a number for the amount of curve. The greater the number, the more curve that is applied. Use the area to the right of the number field to select a unit of measurement (e.g., point, pixel, centimeter). If you want to provide a second value for the rounded border, complete the same fields on the right half of the popup. When you are finished, click **OK** in the small popup.

- **Cell Border Collapse** Select whether you want to collapse the cell borders in the stylesheet. If you collapse the cell borders, the row and cell borders of a table are joined in a single border. If you do not collapse the cell borders, the row and cell borders of a table are detached. If you use the border radius properties to create rounded borders, this must be set to "Do not collapse cell borders."
- **Cell Border Spacing** Use this area to increase or decrease the amount of spacing for a cell border.
- **Overflow** This determines what happens if content overflows the table.
  - **Visible** The overflow is not clipped. It renders outside the table. This is default.
  - **Hidden** The overflow is clipped, and the rest of the content will be invisible. If you are using border-radius properties on the table, you must select this option for the rounded corners to be seen properly.
  - **Scroll** The overflow is clipped, but a scroll-bar is added to see the rest of the content.
  - **Auto** If overflow is clipped, a scroll-bar should be added to see the rest of the content.
  - **Inherit** The value of the overflow property is inherited from the parent element.
- **Print Options** Click this button to open the Breaks dialog and set page, column, and breaks for tables.

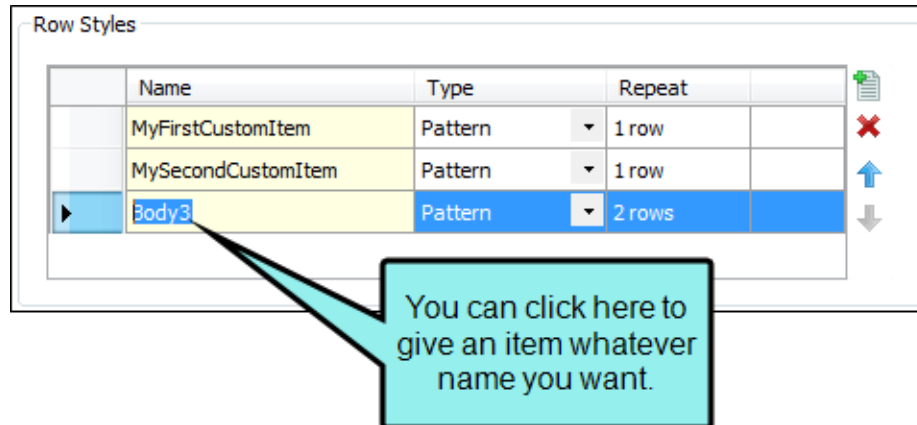
## ROWS, COLUMNS, HEADER, AND FOOTER TABS

These tabs let you set properties for the various elements of the table. For any of these elements, you can add multiple repeatable patterns with different colors and text properties. Following are descriptions for the fields that appear on each tab.

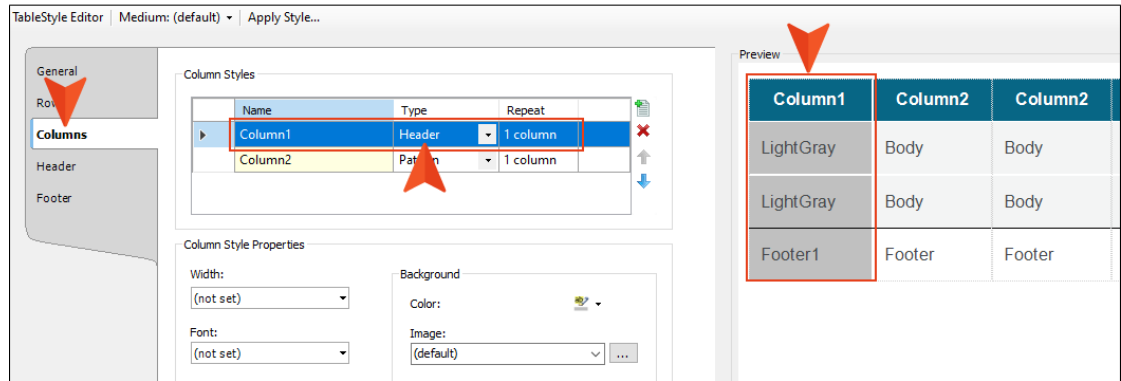
### ROW/COLUMN/HEADER/FOOTER STYLES

Displays the patterns for the row, column, header, or footer in the stylesheet. Each line represents a different pattern and how many times it is repeated in a table before the next pattern occurs.

- **Name** Displays the name of each pattern. Depending on which tab you are working on, the default names of the patterns may be Body1, Body2, Body3, etc.... Column1, Column2, Column3, etc.... Header1, Header2, Header3, etc.... Footer1, Footer2, Footer3, etc. You can click in the cell, press F2, and enter a custom name for each pattern if you like.



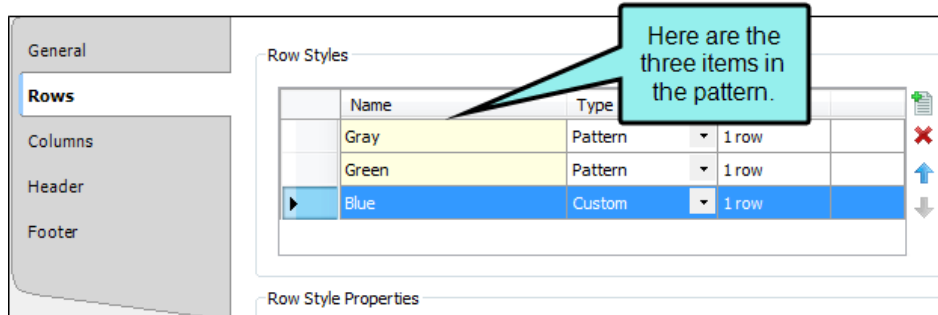
- **Type** Select a type for the item. You can switch the type for an item using the drop down menu.
  - **Header** This is available only in the Columns tab. It can be used to indicate that the entire column acts as a header, carrying more weight in a sense than the rest of the columns, which carry equal weight.



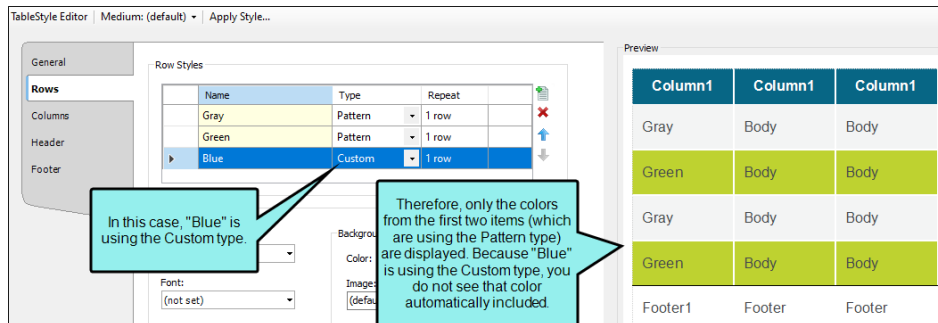
- **Pattern** This lets you create automatic patterns in a table (e.g., alternating background colors).
- **Custom** This type will not be added to a table automatically. Instead, you would need to apply that item manually to the particular areas of the table where you want to use it.



☆ **EXAMPLE** You have a table stylesheet with three pattern items on the Rows tab (Gray, Green, Blue), with alternating background colors.



However, only the first two items are using the Pattern type. The third item is using the Custom type. Therefore, when you insert a table and use this stylesheet, the rows alternate between gray and green only.



- ☆ This particular pattern was created on the Rows tab, which means that it displays only in your body rows, not in any header or footer rows. By right-clicking on the **tbody** structure bar or any of the **tr** structure bars within it, you can select **Row Style** from the context menu. From there, you can select any of the available items in the pattern to override what you already have in the table.

You can override the automatic pattern and manually apply an item's style to the entire collection of regular rows by right-clicking the `<tbody>` tag or to a specific row by right-clicking on a particular `<tr>` tag.

The screenshot shows a table editor interface. On the left, a structure bar displays a tree view of the document: `html` (root), `body` (child), `p` (child), `p` (child), `table` (child), `tbody` (child of table), `tr` (child of tbody), `tr` (child of tbody), `tr` (child of tbody), `tr` (child of tbody). A callout box points to the `tbody` tag in the structure bar. The main content area shows a table with a caption "Table 1-1 Table caption." and a 4x3 grid of cells. The first row is light blue, the second row is light green, the third row is light blue, and the fourth row is light green. Each cell contains the text: "Here is some general text for a table. Replace this with your own content."

Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.
Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.
Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.
Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.



Tables

Here is some general text for a table. Replace this with your own content.

<tr>

- Select
- Copy
- Paste (Above)
- Paste (Below)
- Delete
- Clear
- Move
- Insert New (Above)
- Insert New (Below)
- 0
- Reset Height
- Conditions...
- Row Style

- (default)
- Gray
- Green
- Blue

Words: 195

XML Ed

In this example we right-clicked on the first <tr> tag within the <tbody> tag. This opened a context menu that will let us manually override the style for the first regular table row.

Currently that row is using the Gray item style. But we can choose either of the other items to override it. We will select Blue.

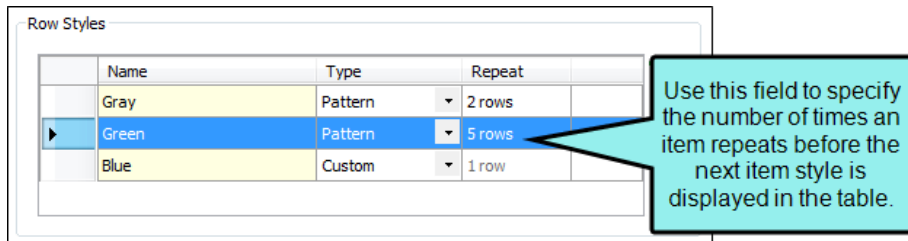
As a result, the background color for the Blue item is manually applied to that row, overriding what the pattern would normally show.

Table caption.



Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.
Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.
Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.
Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.	Here is some general text for a table. Replace this with your own content.

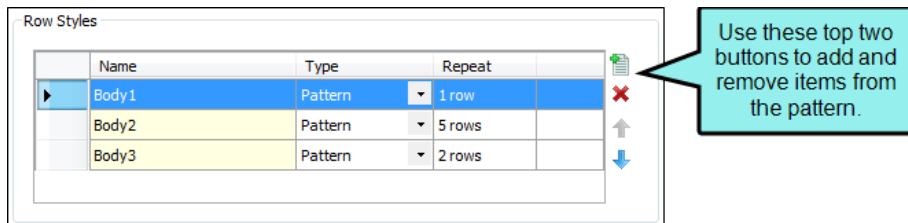
☆ The only difference between the first two items (Gray and Green) and the third item (Blue) is that Blue can be applied only from this context menu manually. Gray and Green are applied automatically, but can be applied manually from the context menu too.

- **Repeat** Click the up or down numbers to increase or decrease the number of times the pattern occurs in a table before the next pattern is displayed.





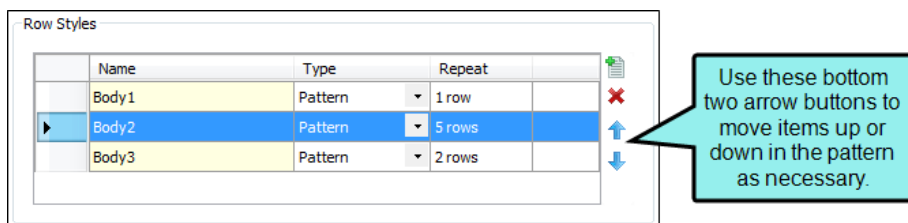
Name	Type	Repeat
Gray	Pattern	2 rows
Green	Pattern	5 rows
Blue	Custom	1 row

-  Adds a new pattern. The new pattern is initially set to repeat just once, but you can change that in the "Repeat" cell.
-  Removes the selected pattern from the list.



Name	Type	Repeat
Body1	Pattern	1 row
Body2	Pattern	5 rows
Body3	Pattern	2 rows

-  Moves the selected pattern up in the list.
-  Moves the selected pattern down in the list.



Name	Type	Repeat
Body1	Pattern	1 row
Body2	Pattern	5 rows
Body3	Pattern	2 rows

## HEIGHT

Select a pattern from the section above. Then click this field to open a small popup, which lets you set properties for the height of the row or width of the column. In the lower-left field enter a number. In the lower-right field, select a unit of measurement (e.g., points, pixels, centimeters) for the number you entered. Then click **OK** to accept the settings, or click **Cancel** to close the window without accepting them.

## FONT

Select a pattern from the section above. Then click this field to open a small popup, which lets you set font properties.

- **Weight** Select an option to change the weight of the font (e.g., bold). The numbers from 100 to 900 represent different levels of darkness. The number 400 is the same as a "normal" weight, and the number 700 is the same as the standard "bold" option. "Bolder" means the next weight that is assigned to a font that is darker than the inherited one. "Lighter" means the next weight that is assigned to a font that is lighter than the inherited one.
- **Style** Select an option to change the style of the font (e.g., italic).

According to the World Wide Web Consortium (w3.org):

*The font style specifies whether the text is to be rendered using a normal, italic, or oblique face. Italic is a more cursive companion face to the normal face, but not so cursive as to make it a script face. Oblique is a slanted form of the normal face, and is more commonly used as a companion face to sans-serif. This definition avoids having to label slightly slanted normal faces as oblique, or normal Greek faces as italic.*

- **Color** Click this field and select a color for the text. For advanced color options, select **More colors** and use the fields in the Color Picker dialog. In the Color Picker, you can select a CSS variable.
- **Size** In the top field, select **Length**. Then in the lower-left field enter a number for the size of the text, and in the lower-right select a unit of measurement (e.g., points, pixels, centimeters) for the number.
- **Family** Click in this field and select a font family (e.g., Arial) for the text.

## RULER

Select a pattern from the section above. Then click this field to open a small popup, which lets you set properties for a rule (i.e., horizontal line) between the rows or columns in the pattern. In the lower-left field enter a number for the size of the rule. In the lower-middle field, select a unit of measurement (e.g., points, pixels, centimeters) for the number you entered. In the upper-right field, select a color for the rule. In the lower-right field, select a type of line (e.g., solid, double, dashed) for the rule. Then click **OK** to accept the settings, or click **Cancel** to close the window without accepting them.

## SEPARATOR

Select a pattern from the section above. Then click this field to open a small popup, which lets you set properties for a separator (i.e., a horizontal line) between the final row or column in the pattern and the first row or column in the next pattern. In the lower-left field enter a number for the size of the separator. In the lower-middle field, select a unit of measurement (e.g., points, pixels, centimeters) for the number you entered. In the upper-right field, select a color for the separator. In the lower-right field, select a type of line (e.g., solid, double, dashed) for the separator. Then click **OK** to accept the settings, or click **Cancel** to close the window without accepting them.

## ALIGNMENT

Select an option for aligning text in the row or column horizontally.

- **Left** The text aligns at the left edge of each cell.
- **Center** The text aligns in the center of each cell.
- **Right** The text aligns at the right edge of each cell.
- **Justify** The text aligns both at the left and right edges of each cell.

## VERTICAL ALIGNMENT


Select an option for aligning text in the row or column vertically.

- **Top** The text aligns at the top of each cell.
- **Middle** The text aligns in the middle of each cell.
- **Bottom** The text aligns at the bottom of each cell.

## PRINT OPTIONS (ROWS ONLY)

Click the **Print Options** button to open the Breaks dialog and set page and column breaks for table row elements.

## BACKGROUND

Use this area to specify the settings that you want for the background. In the **Color** field, click the down arrow and select a color from the popup. For advanced color options, select **More Colors** and use the fields in the Color Picker dialog. Next to the **Image** field, click . Select an image file to insert and click **OK**. If you want the background image to repeat, select one of the options from the **Repeat** field. You can also set the image position horizontally and vertically by using the **X** and **Y** fields.

## CELL PADDING

Click in any of the individual fields (**Left, Right, Top, Bottom**) to specify the settings for the cell padding (the amount of space between the edge of the table cell and the content in the cell). In the left side of the field, enter a number for the amount of padding. In the right side of the field, select a unit of measurement (e.g., point, pixel, centimeter) for the number you entered. If you click the down arrow to the right of all the fields, the settings will be applied to all of the cell padding fields. When you click that down arrow, a small popup displays. Use the lower-left area of the popup to enter a number for the amount of padding. Use the lower-right area to select a unit of measurement.

## CELL CONTENT STYLE

When you insert a table, it is set up by default to use standard table tags in the individual cells (e.g., <th> for table headers, <td> for regular table text). However, if you press ENTER at the end of a line, a <p> tag is added within the standard tag. Therefore, in order to keep all of the content in your table cells looking consistent, there are a couple of things you can do. First, if you are editing table styles in a regular stylesheet, you can create advanced selectors (e.g., "td p"). Second, you may want to create a special style class of the p style to be used for table content (e.g., p.tabletext) and apply that style to all of your cells when you first create a table. You can manually apply specific styles to tables by selecting the table cells, clicking **Table > Cell Content Style**, and choosing the style to be used for those cells.

However, rather than repeating all these steps each time you create a table, the easiest way to accomplish this is to set a default cell content style. You can do this in a couple of ways: globally or using a table stylesheet.


The fields in this section let you set default styles in the table stylesheet for whatever tab you're on (Rows, Columns, Header, Footer). In the **Tag** field, select the parent style (usually p). Then in the **Class** field, select any class that is available for that parent style (e.g., TableRowText).

You can have different defaults for each table stylesheet in your project.

When you insert a new table using a particular table stylesheet, the various parts of the table (e.g., header, row, footer) will automatically start out with the appropriate styles so that you don't have to set any of them manually.


This feature automatically applies the selected style class only in new tables (and in new cells within existing tables) that are associated with the table stylesheet. It does not affect existing tables.

If you also have a style set in the Table ribbon using the global method, your settings in a table stylesheet override that style.

 **NOTE** As an alternative to the Cell Content Style feature, you could use advanced selectors to automatically control the look of content in tables.

## PATTERN EXAMPLE


Let's say you want the rows in the table to alternate between having no background color and a green background color. In addition, you want a header row to have a blue background. To do this, you would complete the following steps:

- a. Select the **Row** tab.
- b. In the **Row Styles** section, click . There should now be two patterns (Body1 and Body2).
- c. To make the patterns more identifiable, click in the **Name** cell (where "Body1" is shown) and press **F2**. Then replace the existing text and type `NoColor`.
- d. Click in the **Name** cell (where "Body2" is shown) and press **F2**. Then replace the existing text and type `Green`.
- e. The **Type** cell should already be set to "Pattern" for each, and the **Repeat** cell should already be set to "1" for each. Keep those fields set as they are.
- f. Select the **Green** pattern row.



- g. In the **Background/Color** field, select a green color.
- h. Select the **Header** tab.
- i. Click in the **Name** cell and press **F2**. Then replace the existing text and type **Blue**.
- j. In the **Background/Color** field, select a blue color.

The Preview section lets you see how the table elements look as you make changes.


- 3. Click  to save your work.

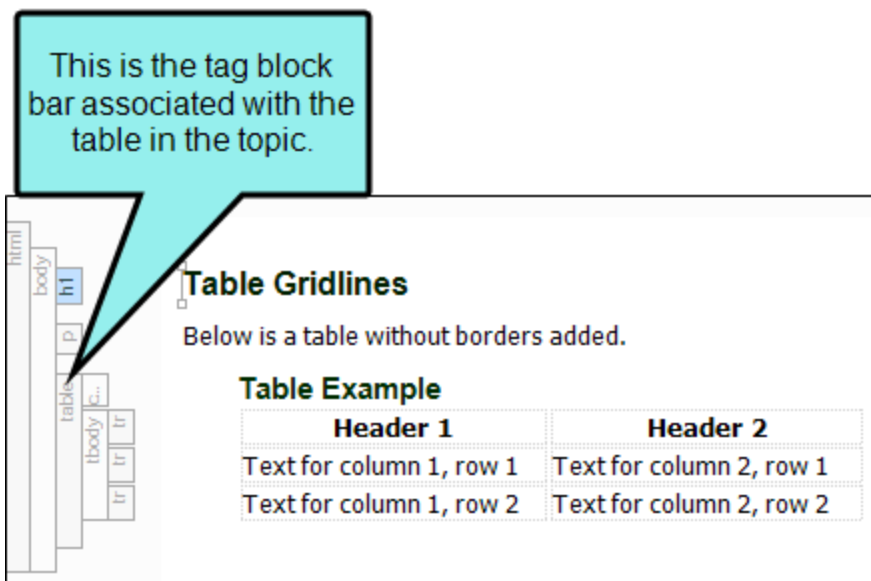
# Applying Table Stylesheets to Tables

After you edit a table stylesheet, you can apply the stylesheet to a table. You can do this to one table at a time, or to multiple tables at the same time.

## How to Apply a Table Stylesheet—One Table at a Time

The following steps show how to apply a table stylesheet to one table at a time. Use the next set of steps if instead you want to quickly apply a table stylesheet to many tables in different files.

1. Open the topic where a table has been inserted.
2. If the tag block bars are not shown to the left of the content, click  at the bottom of the editor.
3. Right-click the `table` tag bar.



This is the tag block bar associated with the table in the topic.

**Table Gridlines**  
Below is a table without borders added.

**Table Example**



Header 1	Header 2
Text for column 1, row 1	Text for column 2, row 1
Text for column 1, row 2	Text for column 2, row 2

4. Do one of the following:

- In the context menu select **Table Style** and then select the stylesheet in the submenu.

OR

- a. In the context menu, select **Table Properties**. The Table Properties dialog opens.
- b. Click **Table Style**, and from the drop-down select the stylesheet.

 **NOTE** If you have not yet created a table stylesheet yet, you can do so by clicking  to the right of the field. Click the face of the button, not the down arrow.

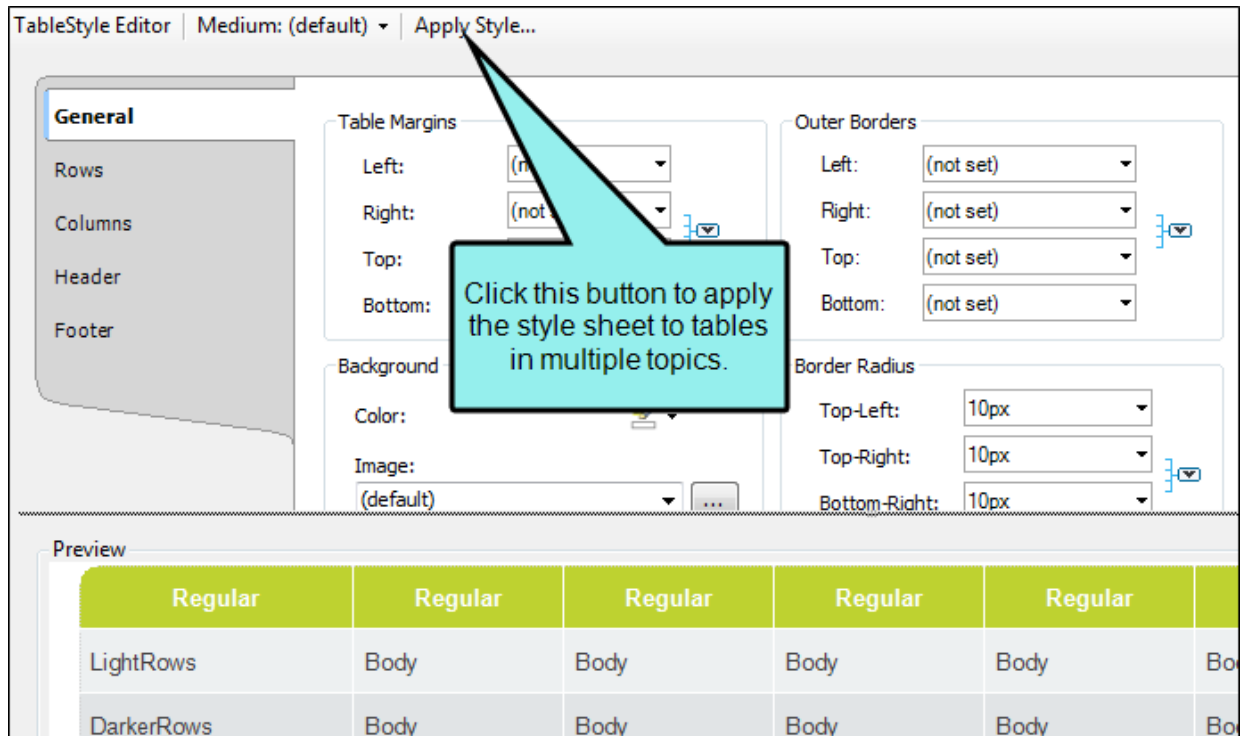
- c. Click **OK**.

5. Click  to save your work.

# How to Apply a Table Stylesheet—Multiple Tables and Files

The following steps show how to quickly apply a table stylesheet to many tables in different files. Use the previous set of steps if instead you want to apply a table stylesheet to one table at a time.

1. Open the table stylesheet that you want to apply.
2. In the local toolbar of the Table Style Editor, click **Apply Style**.

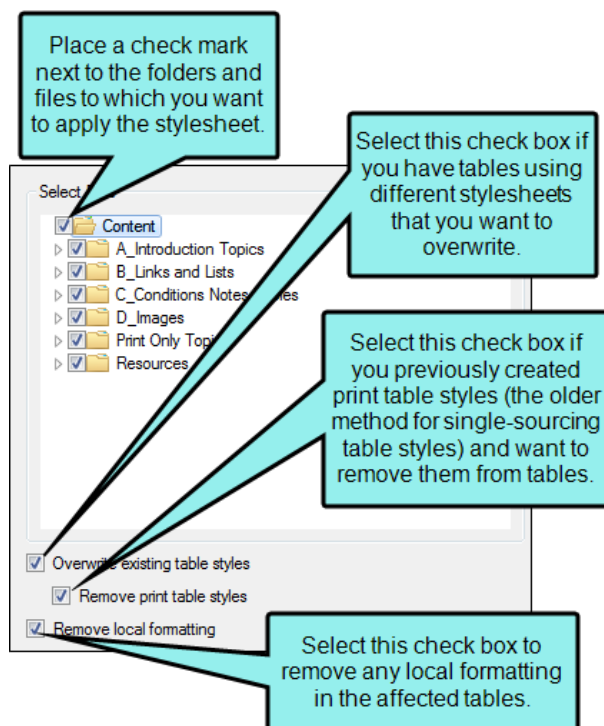


The Apply Table Style dialog opens, displaying all folders and topic files in your project.

3. Click in the check box next to each folder and/or file containing tables that should use the stylesheet. If you click next to a folder, the stylesheet will be applied to all files and subfolders under it.

You can also select either of the following options in the dialog.

- **Overwrite existing table styles** Select this if you want to overwrite existing table styles in those topics. If you choose this option, you can also select **Remove print table styles**. As of Flare V7, print table styles are no longer the preferred method for single-sourcing tables in online and print output; mediums are now the recommended solution. Therefore, if you used print table styles in the past and want to remove them now, you can use this option. When this option is enabled, print table styles will be removed from any tables updated by this dialog.
- **Remove local formatting** Select this if you have local formatting in tables and want to remove it when the table style is applied.



4. Click OK.

# Mediums and Media Queries

You can use mediums and media queries in Flare to produce various outputs with different appearances. These are similar concepts; in fact, you will see mediums and media queries side by side in different places in Flare's user interface. However, they are not the same.

This chapter discusses the following:

Mediums .....	299
Media Queries .....	309
General Information for Mediums and Media Queries .....	312
Main Activities for Mediums and Media Queries .....	319

# I Mediums

A medium is an alternative group of settings in a stylesheet and can be very useful when you are generating multiple kinds of outputs. Unless you tell Flare otherwise, default style settings will be used for the different outputs you generate. But there may be times when you want to override a default style setting for a particular output; that's why you would use a medium. *You need to explicitly tell Flare which medium you want a particular target to use.* This is done from the Advanced tab of the Stylesheet Editor.

One use for a medium is to have one group of style settings for online formats and a different group of settings for print-based formats; therefore, you could use one medium for your online targets and another medium for print-based targets. Another example is if you need to generate multiple PDFs in different color themes; most of the styles might be the same in each PDF, but you can use mediums to separate the color and other stylistic differences.

## General Information

- "Default and Print Mediums" on the next page
- "Multiple Medium View" on page 301
- "Table Styles and Mediums" on page 307
- "Print Medium and Page Layouts" on page 307
- "Medium Organization and Printing" on page 308

## Main Activities

- "Creating Mediums" on page 320
- "Selecting Mediums and Media Queries" on page 323
- "Associating a Medium With a Target" on page 335
- "Renaming Mediums and Modifying Media Queries" on page 335

☆ **EXAMPLE** You want to build both online and print-based outputs from your project. In your stylesheet, the default style for a text hyperlink is blue with an underline. That may be fine for your online outputs, but maybe you want this style to appear in black font with no underline for print-based output.

Therefore, in your stylesheet you leave the default setting as it is, so that your online outputs can use it. But then you open the print medium in the stylesheet and change the text hyperlink style to black font with no underline.

Your online targets are set to use the default styles, and your print-based targets are set to use the print medium. Therefore, when you build the output for the print target, it will still use the default styles for many pieces of content, but the text hyperlinks will be black with no underline.

## Default and Print Mediums

Flare provides you with a default and print medium. These are enough to satisfy the needs of many authors, but you can create additional mediums if necessary.

- **default** This is the standard medium. Any settings that you specify in the default medium will "trickle down" automatically to the other mediums. However, you can override any setting in a specific medium.
- **print** This medium is designed to be used for print-based output types (Adobe PDF, Microsoft Word). When you create a print-based target, this medium is automatically associated with it, although you can select a different medium for the target if necessary.

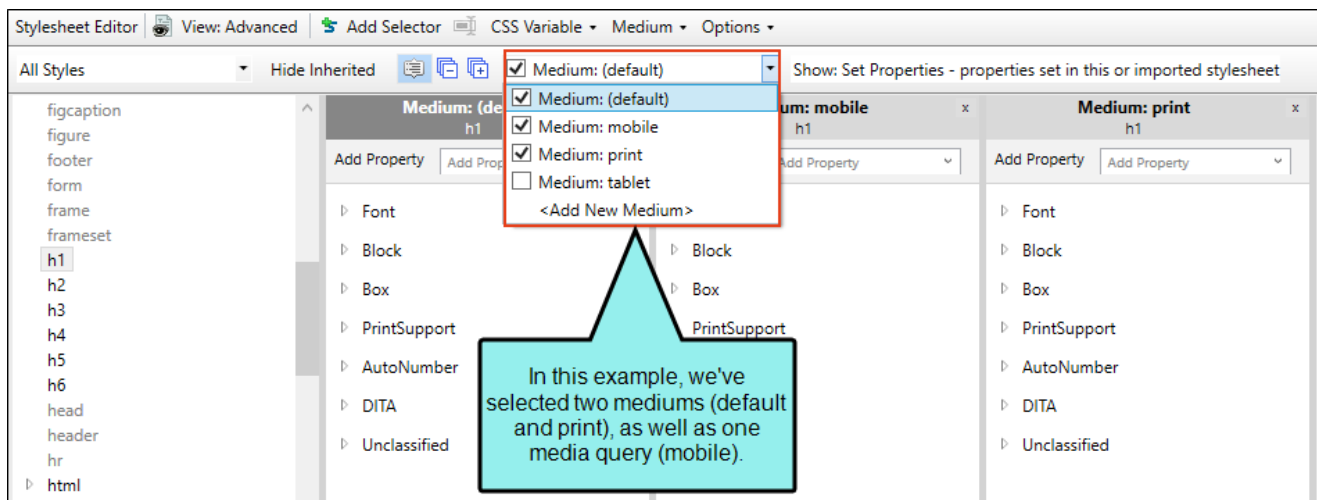


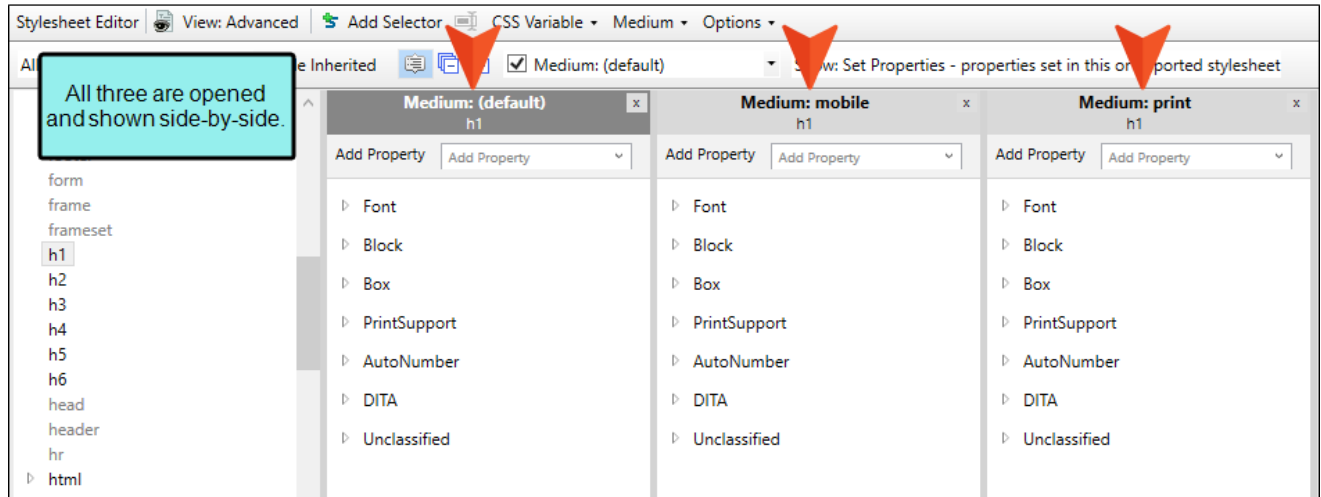
# Multiple Medium View

If you are working in the Advanced View of the Stylesheet Editor, you can open and edit multiple mediums at same time. You can do this by clicking the **Medium** drop-down field in the local toolbar and choosing the mediums you want to see so that they have a check mark next to them. Media queries are also listed in the drop-down and can be opened at the same time as well.

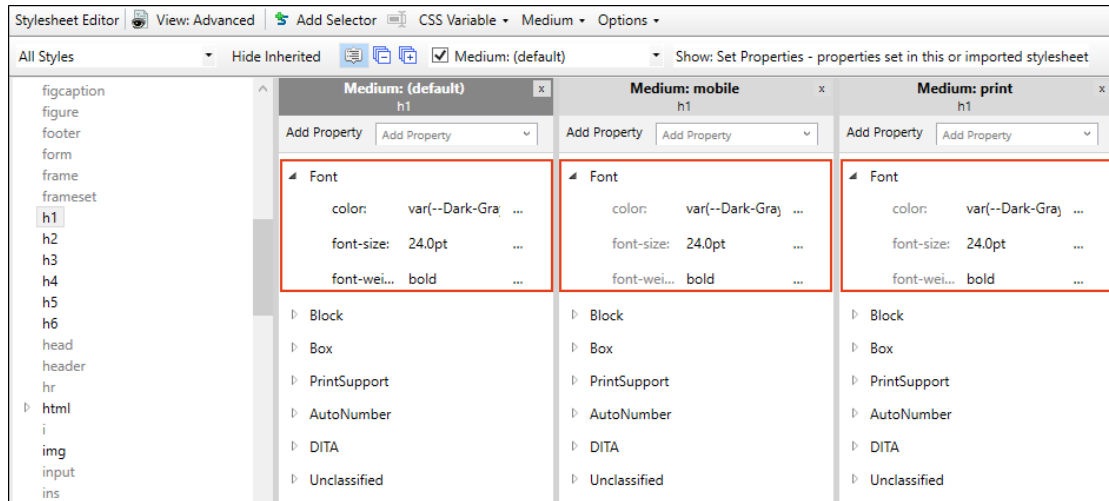
When choosing a medium/media query from the drop-down, you can click on the check mark or the name of the medium/media query. What happens next depends on how many mediums/media queries are currently open in the editor.

- **If One Medium or Media Query is Open** If only one medium/media query is open and you click a *check box* next to another medium/media query, that second medium/media query will open next to the first one. However, if you select the *name* of the medium/media query in the drop-down, it will open and the first medium/media query will close.
- **If Multiple Mediums or Media Queries are Open** If two or more mediums/media queries are open, the next medium/media query you select will open next to the others. This is true whether you select the check box or the name of the medium/media query.

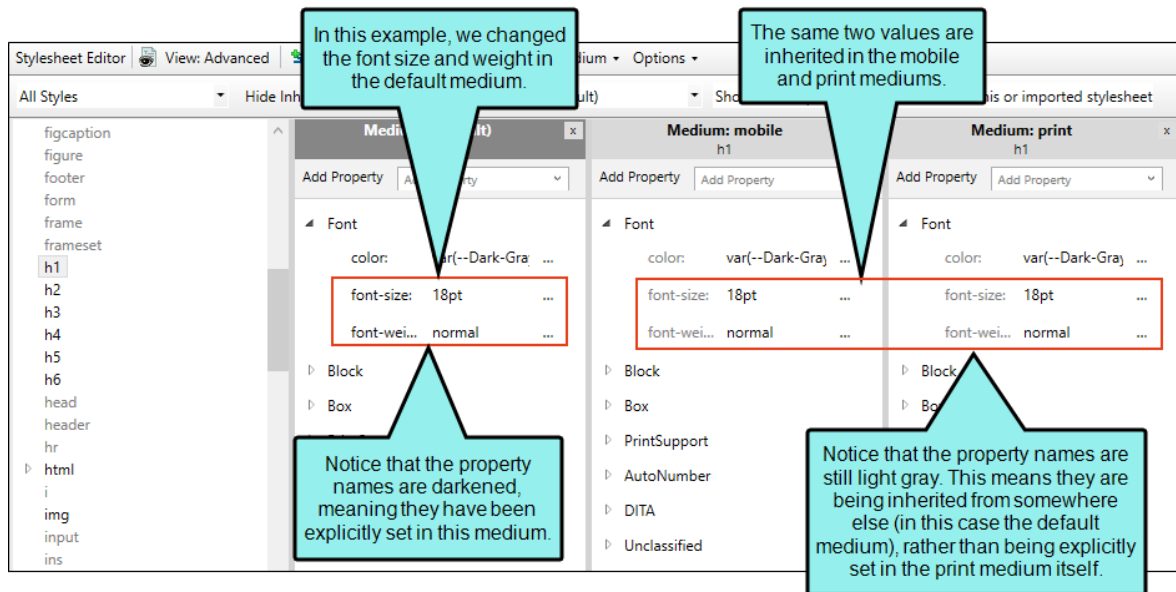




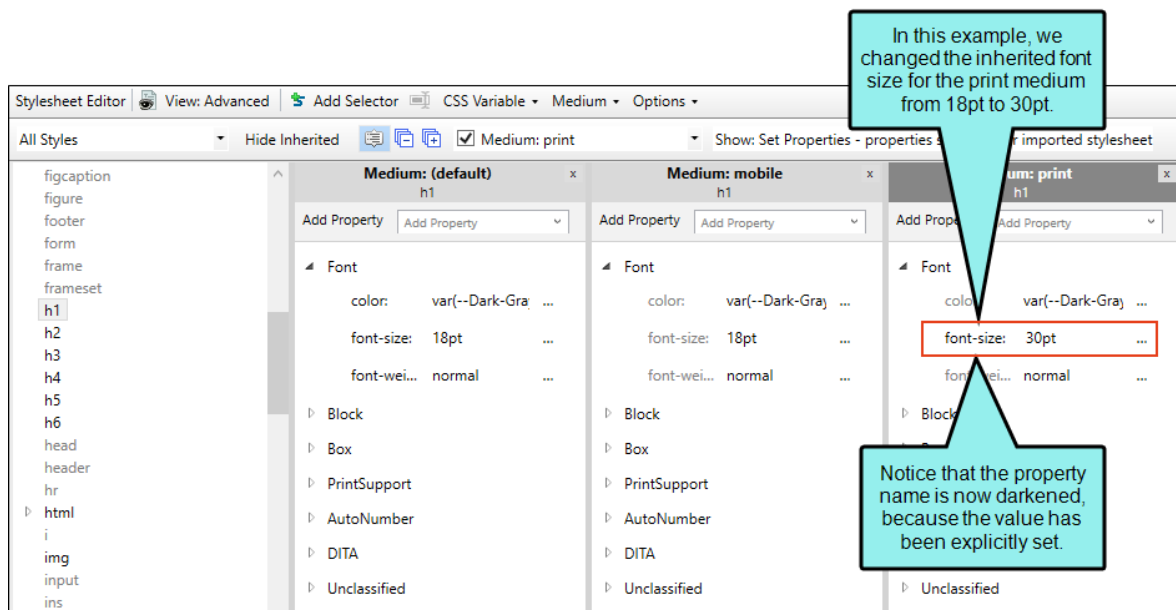
When you expand a property group in one medium or media query, the same property group also expands in any others that are open.



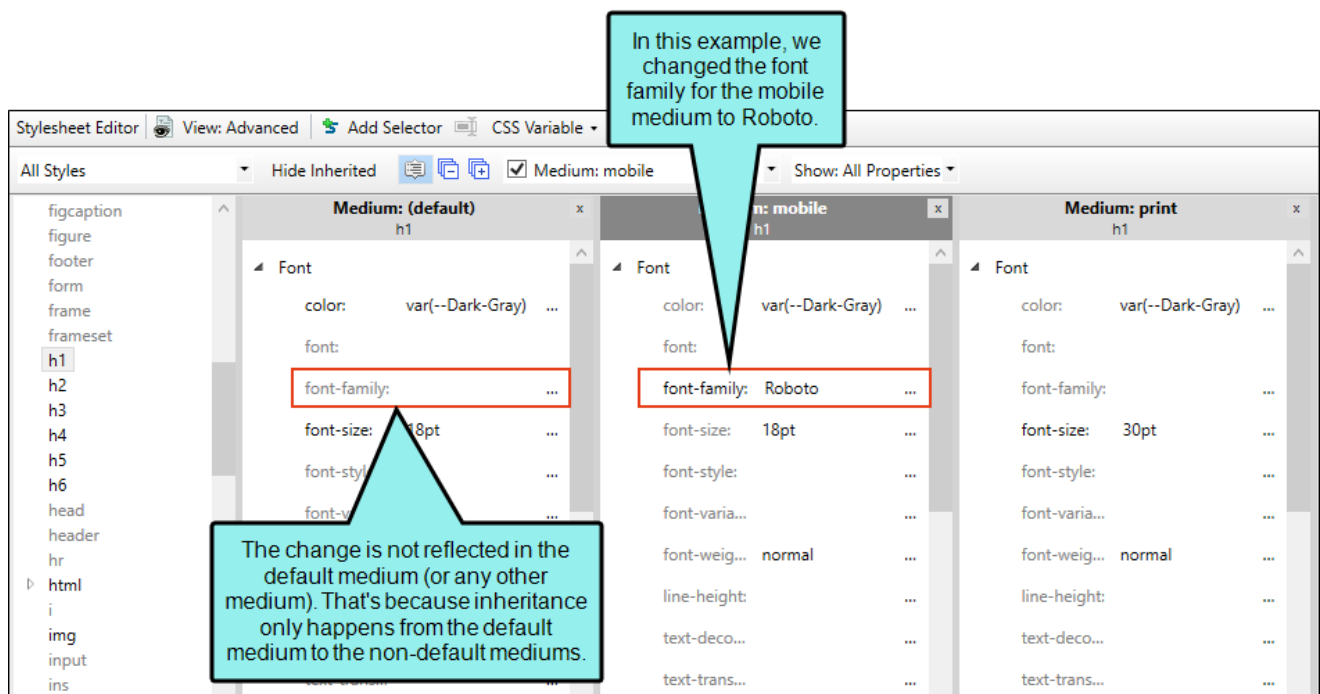
If you make a change in default medium, you see it applied also to the other mediums and media queries, because they inherit whatever is added in the default medium.



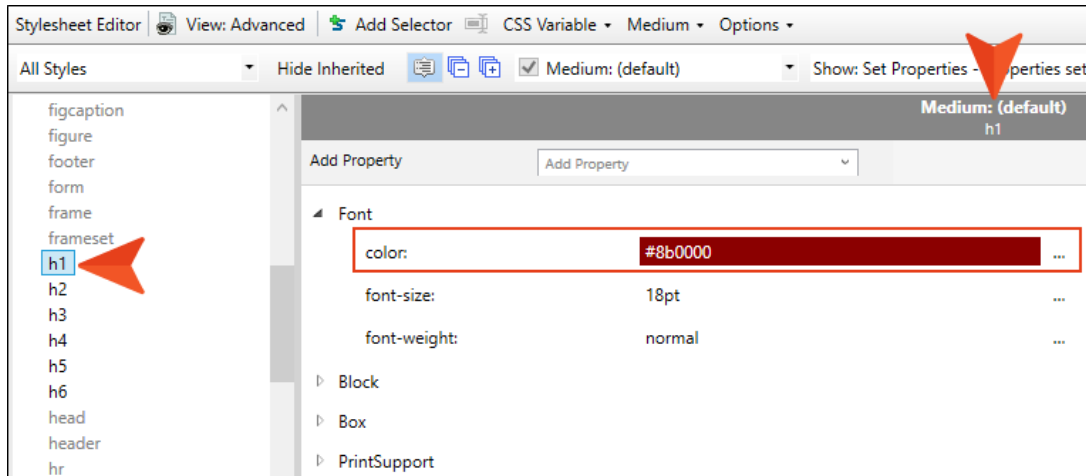
Of course, you can override any inherited property value in a specific medium or media query.



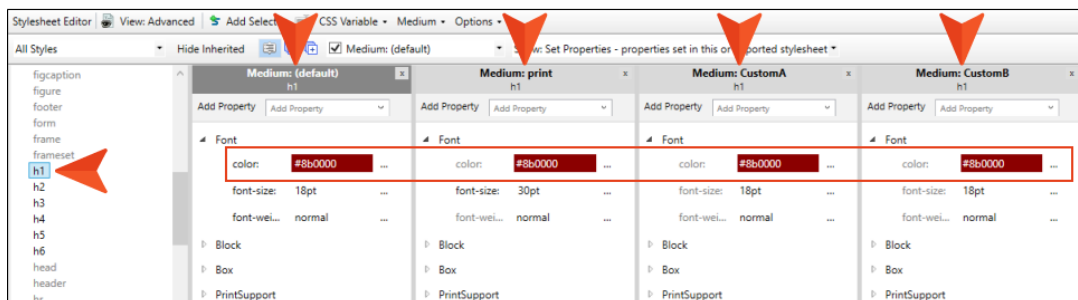
If you make a change in a non-default medium or in a media query, you see it only in that place.



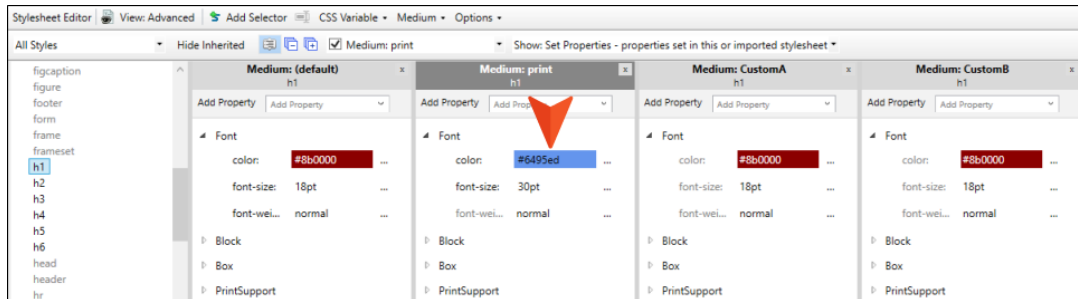
☆ **EXAMPLE** You are using four different mediums (default, print, CustomA, CustomB) in your project. Let's say you specify that the font color for the h1 style in the default medium should be dark red.



If you were to then open any of the other three mediums, you would see that the font color for the h1 style in each of those is also dark red.



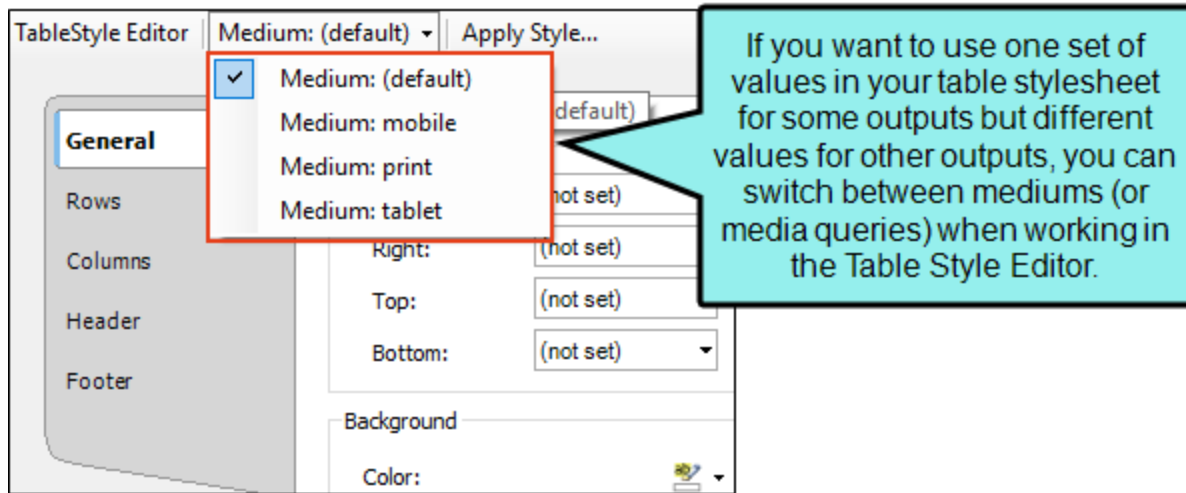
- ☆ If you were to change the font color for the h1 style to blue in the print medium, that color will be used for that medium. However, the h1 style will continue to be displayed in dark red for the default, CustomA, and CustomB mediums.



- 📄 **NOTE** To access the mediums and media queries provided by Flare (print, tablet, mobile), you might need to make sure the **Hide Inherited** option in the local toolbar of the Stylesheet Editor is *not* selected. However, this is not necessary once you make an explicit change in one of those mediums or media queries; after that, it will show up in the Medium drop-down whether you use the Hide Inherited option or not.

# Table Styles and Mediums

Not only are mediums and media queries accessible from the Stylesheet Editor, you can also use them in the Table Style Editor when working with table stylesheets. See "Table Stylesheets" on page 270 and "Creating Table Stylesheets" on page 272.



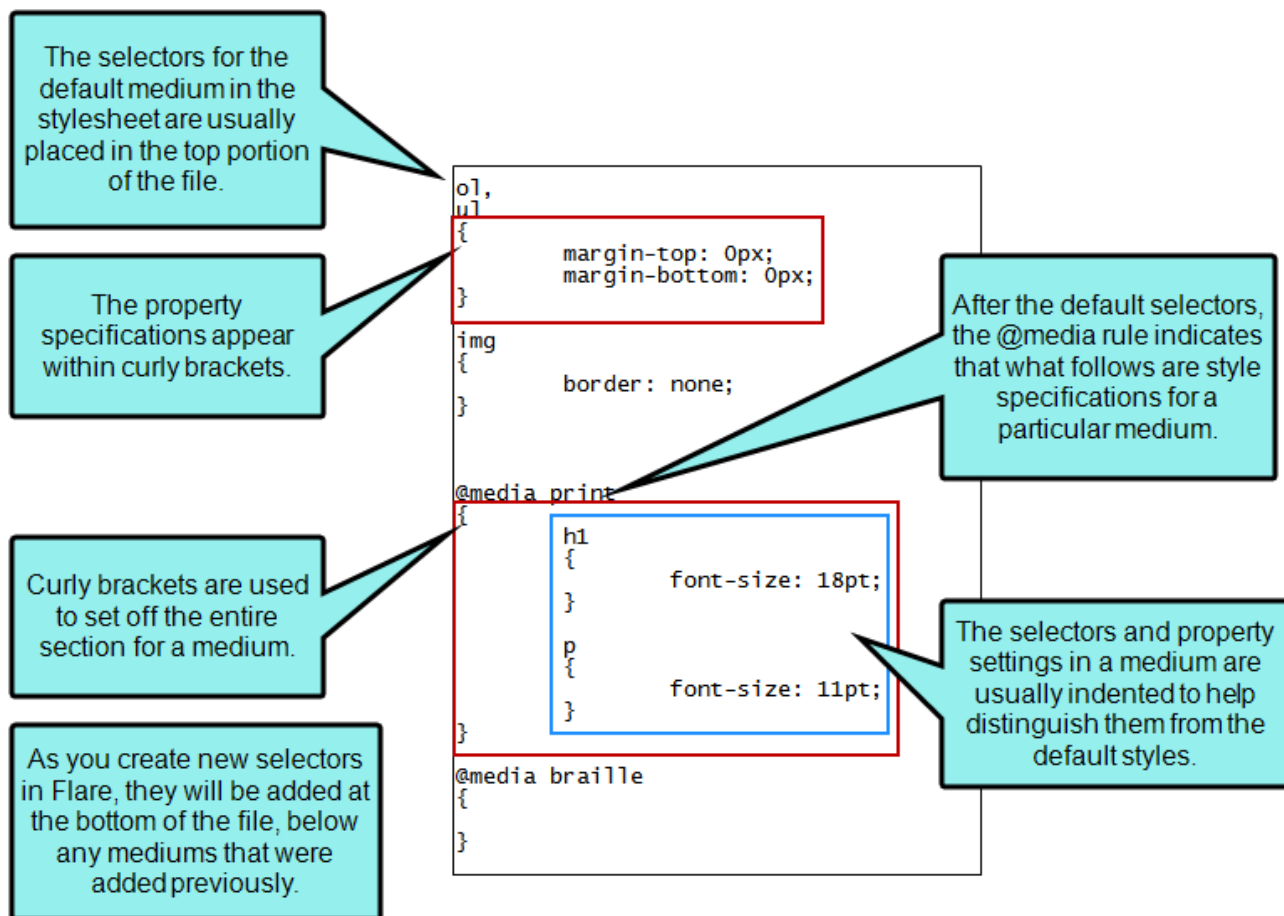
## Print Medium and Page Layouts

Page layouts render content using the print medium settings in the stylesheet.

☆ **EXAMPLE** You have a style class called "p.Section" and you apply it to the text you have in a header frame in a page layout. In the default style medium, the font color for this style is red, but in the print medium it is black. It makes more sense to display the font for a page layout based on the print medium, because that medium will most likely be used in the print-based output. Therefore, the text displays as black in the page layout.

# Medium Organization and Printing

Different mediums (and media queries) in a stylesheet are set apart by the @media rule and sets of curly brackets. This can be seen when you open the stylesheet source file; to do this, navigate to the CSS file in the Content Explorer, right-click on it, and select **Open with > Internal Text Editor**.

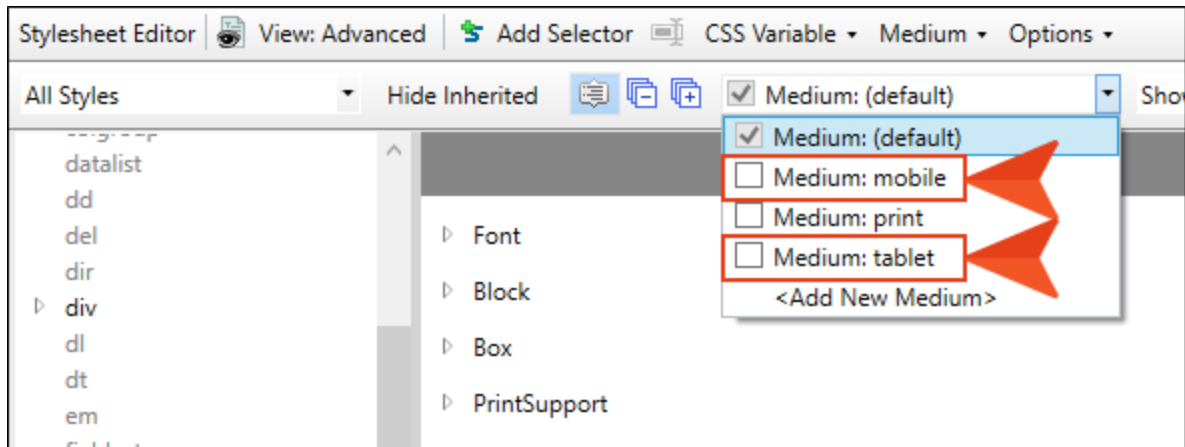


The placement of the medium section in the stylesheet affects which styles are used when you print topics. For example, if you are using the print medium and want those styles to be used when printing a topic from a web browser, the @media print section should be placed in the stylesheet after your default media section, because style properties lower in a CSS take precedence over properties placed above them. You can always move the medium sections around in the stylesheet (if you want them to be placed higher or lower in the file) by cutting and pasting them in the Internal Text Editor.



# I Media Queries

A media query is an alternative group of settings in a stylesheet. These settings are automatically used under certain conditions, such as when a screen of a certain size is displaying the output. Media queries are able to do this because they are configured with specific criteria (e.g., maximum width of the screen, orientation, resolution). When the criteria are met, the style settings in the media query are used to display the output. *You do not tell a Flare target to use a media query; it just happens automatically.*



## General Information

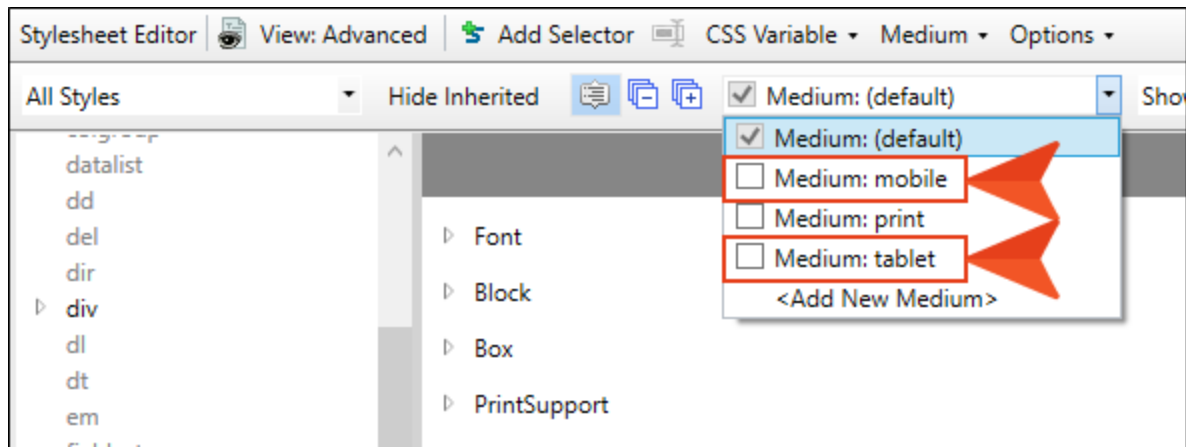
- "Tablet and Mobile Media Queries" on the next page
- "Uses for Media Queries" on page 311

## Main Activities

- "Creating Media Queries" on page 321
- "Selecting Mediums and Media Queries" on page 323
- "Renaming Mediums and Modifying Media Queries" on page 335

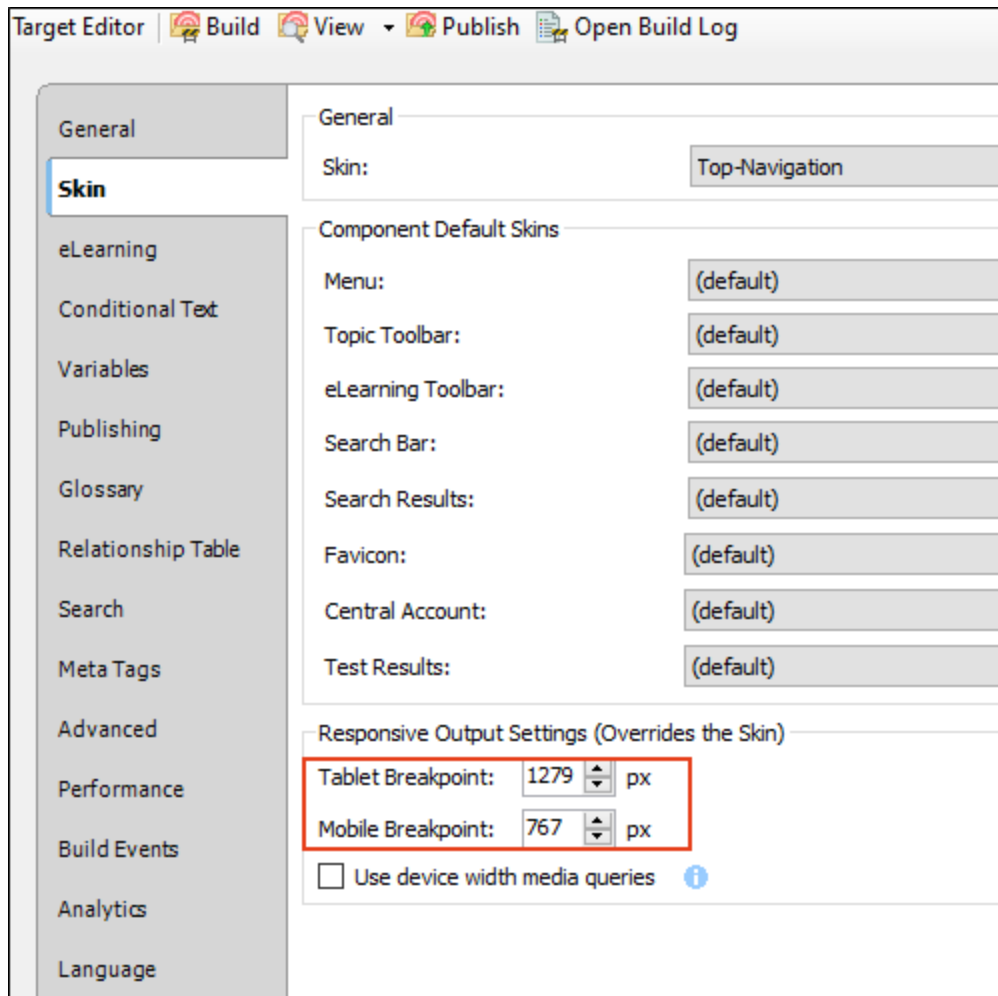
# Tablet and Mobile Media Queries

A couple of media queries (tablet and mobile) are already provided in Flare.



You can create additional media queries from the Stylesheet Editor if you want, but most authors will be able to do everything they need with just the tablet and mobile media queries. The tablet media query is designed to be used on medium-sized screens, such as iPads. The mobile media query is designed to be used on smaller screens, such as smart phones.

The tablet and mobile media queries are tied to the responsive output settings on the Skin tab of the Target Editor for HTML5 targets. The breakpoints provided in the target determine the point at which your media queries will become active in displaying the different style settings.



## Uses for Media Queries

You can place any valid CSS style settings in a media query, just as you can in a medium. For example, if you want paragraph text to suddenly turn blue when a topic is viewed on an iPhone, you can edit the mobile media query, telling it to use a blue font for all paragraph styles.

One of the most common reasons to use a media query is to account for how the structure of content needs to shift or change when viewed on screens of different sizes. You can use Flare's Responsive Layout window pane to do exactly that.

# I General Information for Mediums and Media Queries

There are various pieces of general information you should know if you plan to use this feature.

- "Mediums and Media Queries Together" below
- "Conflicts With Mediums and Media Queries" on the next page

## Mediums and Media Queries Together

You do not need to choose between using a medium or a media query for a target. They can be used alongside one another when you generate output.

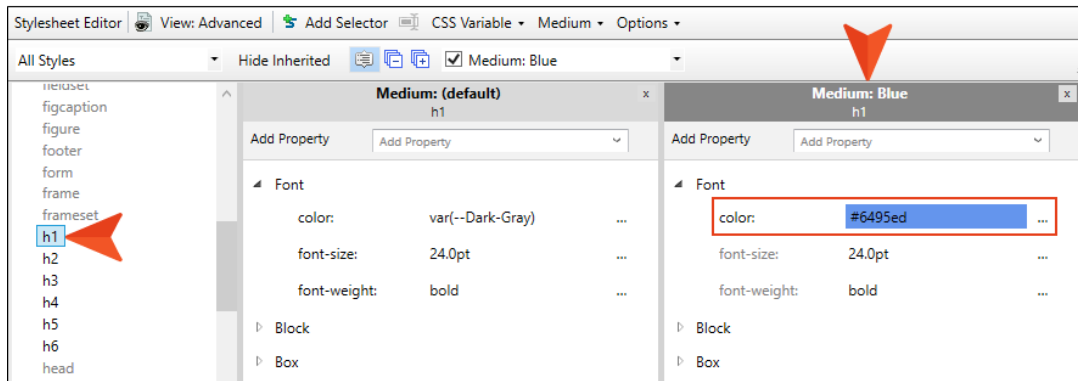
☆ **EXAMPLE** You have custom medium called "OnlineBlue," which you associate with one of your targets to show content with a blue theme on some styles. But in addition to that medium, you also make some edits to the tablet and mobile media queries in your stylesheet so that content is adjusted for smaller devices.

When you view the output on a large monitor, the content will look a certain way that makes sense on a big screen. When you view it on a tablet, the content might shift a bit so that it looks better on that smaller screen. And when you view it on an iPhone, the content might shift again to account for that device. But in all three cases, the content is still adhering to the blue theme from the OnlineBlue medium.

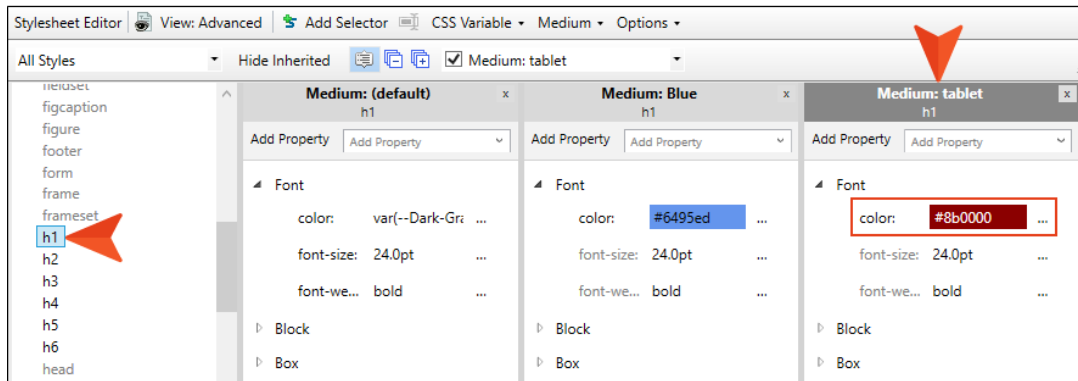
# Conflicts With Mediums and Media Queries

What if there is a conflict between your selected medium and a media query, or a conflict between media queries themselves? Media queries will always have precedence over your selected medium. As far as multiple media queries are concerned, the end result depends on the order of the media queries in your stylesheet (when viewing it in the Internal Text Editor). The general rule is that priority is given to whichever media query is listed last (i.e., the one that was added most recently) in the stylesheet. Then the next one above has the next highest priority, and so on. You can always open your stylesheet in the Internal Text Editor and change the order of the media queries.

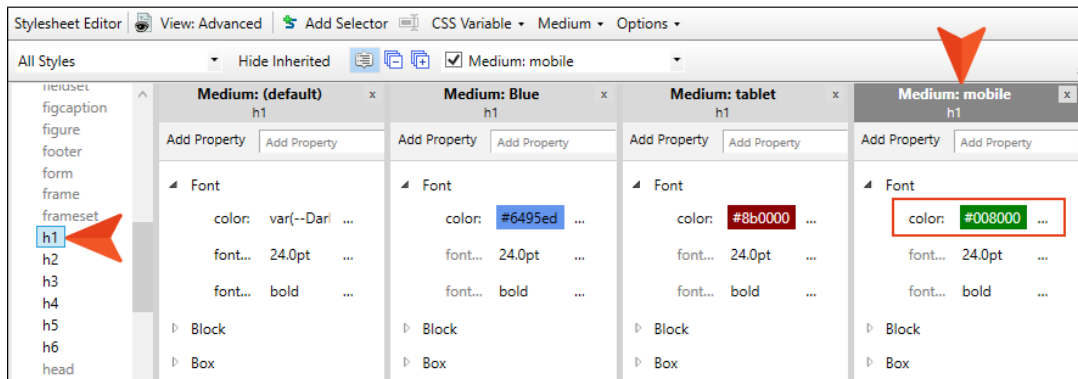
☆ **EXAMPLE** You create a custom medium, naming it "Blue." You associate this medium with your HTML5 target. And when you open your stylesheet, you select this medium and tell Flare that the h1 style should be blue.



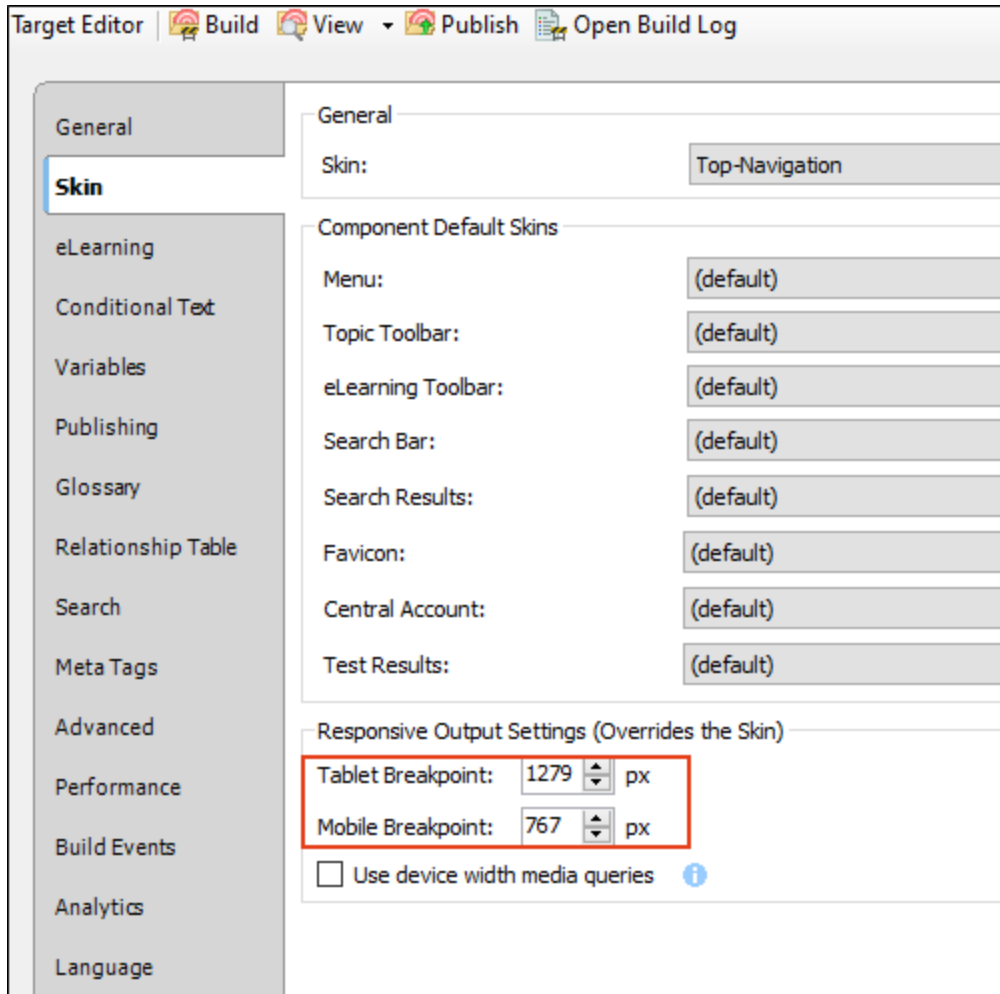
☆ Next, you edit the factory tablet media query and you specify that the h1 style should be dark red.



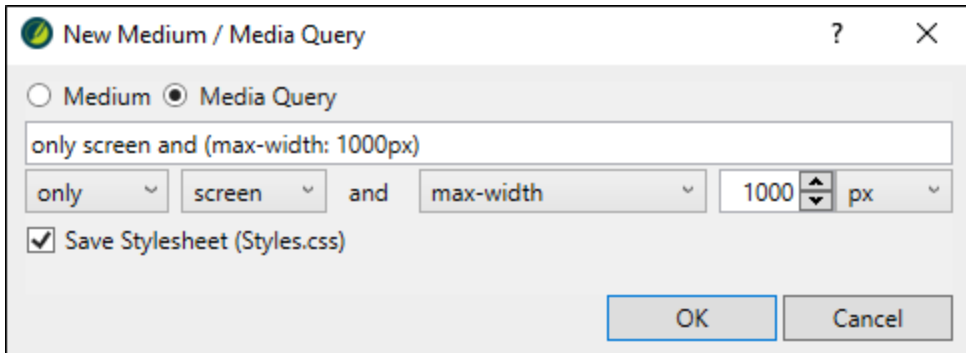
After this, you edit the factory mobile media query and tell Flare to make the h1 style green.



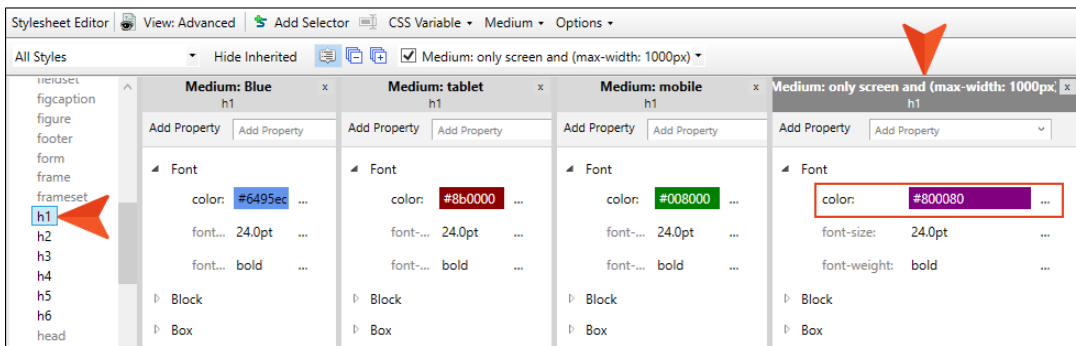
- ☆ If you open your HTML5 target and look at the **Skin** tab, you will see that the tablet maximum width is set to 1279 pixels, and the mobile maximum width is set to 767 pixels.



- ☆ You decide that you want a different look when the screen is between 1279 and 767 pixels. Therefore, you create a new media query, setting it to a maximum width of 1000 pixels, like this:



In the stylesheet, you set the h1 style for this media query to purple.





☆ If you open the Content Explorer, right-click on your stylesheet, and select **Open with > Internal Text Editor**, the stylesheet will open in that editor. When you scroll to the bottom, you will see the custom medium and media queries are in this order (with @media showing where each section of styles begins):

1. @media tablet
2. @media mobile
3. @media print (not being used in our example)
4. @media Blue
5. @media only screen and (max-width: 1000px)

Now you generate the HTML5 target and view the output. When the browser is maximized, you will notice that the h1 headings are blue, because that's what you set on the Advanced tab of the "Blue" medium.

Then you reduce the width of the browser. Once it gets to 1279 pixels, the h1 headings turn from blue to dark red, because that's what you set in the tablet media query.

You reduce the width of the browser even more. When you get to 1000 pixels, the h1 headings turn from red to purple, because that's what you set in your custom media query.

You continue to reduce the width of the browser. You're expecting the h1 headings to display in green (the color in the mobile media query), but they never do. What happened?

Remember that you added your custom media query last, so it appears at the bottom of the stylesheet. The tablet color (dark red) showed up because its width is higher than the custom media query width. As soon as it hit 1279 pixels, your browser saw that there were instructions to change the color to red. Even though the custom media query appears at the bottom of the stylesheet (therefore with the highest priority), its lower width setting meant that it wasn't in conflict with the tablet setting at that point. It was only at the point when the browser got to 1000 pixels that there was a conflict, so at that moment, the browser chose the media query that is lower in the stylesheet and displayed the text in purple.

The problem with the mobile media query is that by the time the browser was reduced to 767 pixels, it was already using a color for a media query with a higher priority. So the mobile color for h1 never got an opportunity to display.

☆ To fix this, you can cut the mobile media query in the Internal Text Editor and paste it last in the stylesheet, so that the order is like this:

1. @media tablet
2. @media print (not being used in our example)
3. @media Blue
4. @media only screen and (max-width: 1000px)
5. @media mobile

Now when you generate and view the output, it will work as you expect when you reduce the browser width: from blue to dark red to purple to green.

# I Main Activities for Mediums and Media Queries

Some activities are particularly common and important when it comes to this feature.

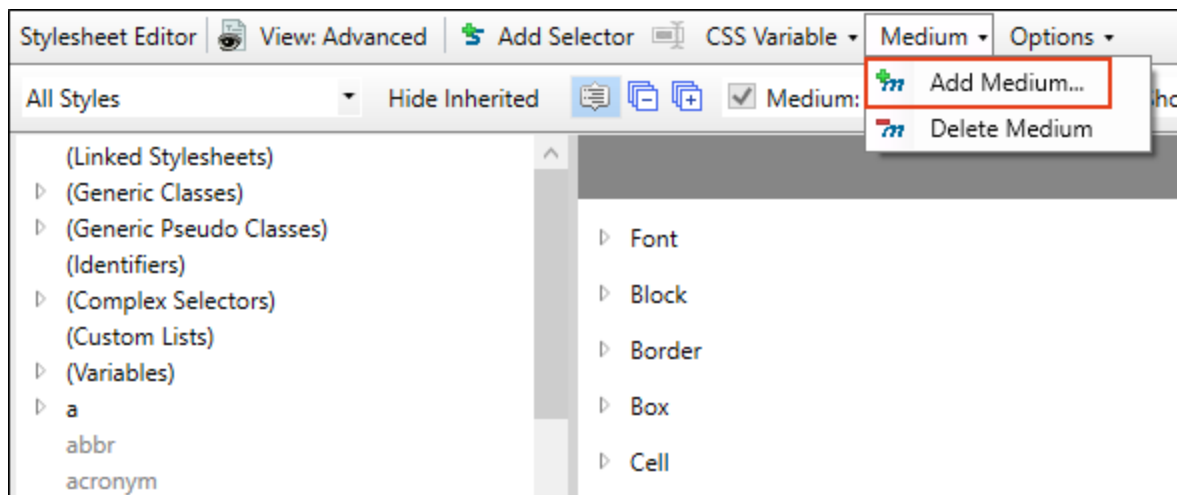
- "Creating Mediums" on the next page
- "Creating Media Queries" on page 321
- "Selecting Mediums and Media Queries" on page 323
- "Associating a Medium With a Target" on page 335
- "Renaming Mediums and Modifying Media Queries" on page 335

# Creating Mediums


In addition to the default medium, a print medium is also provided for you. If necessary, you can add more mediums to your stylesheet.

## How to Create a Medium

1. Open a stylesheet.
2. In the local toolbar of the Stylesheet Editor, click **Medium (default)** and select **Add Medium**.



The New Medium/Media Query dialog opens.

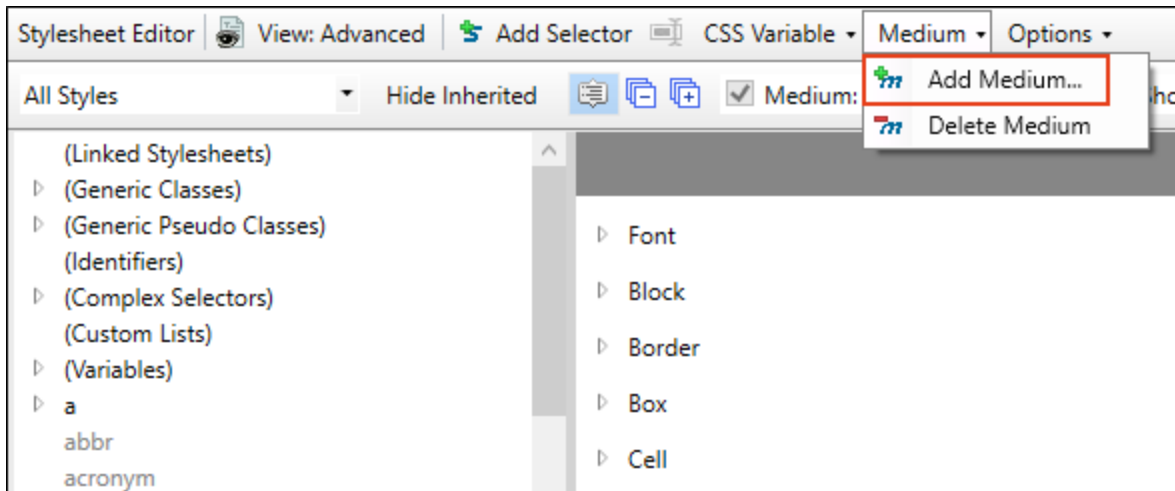
3. Select **Medium**.
4. Enter a name for the medium (with no spaces).
5. Click **OK**.
6. Click  to save your work.

# Creating Media Queries

A tablet and mobile media query are already provided for you. If necessary, you can add more media queries to your stylesheet.

## How to Create a Media Query

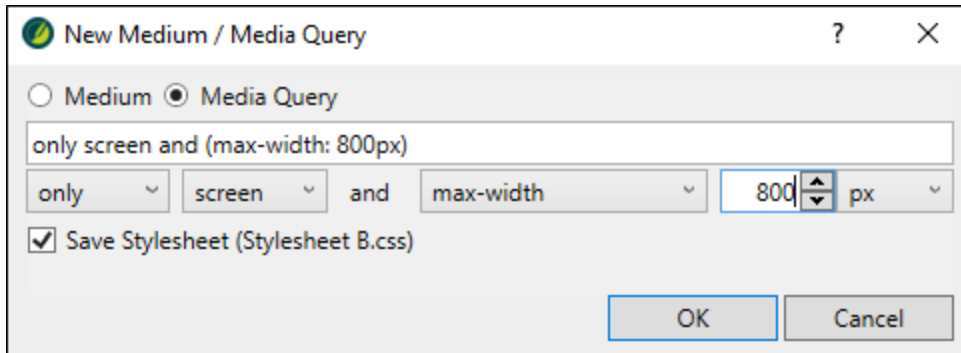
1. Open a stylesheet.
2. In the local toolbar of the Stylesheet Editor, click **Medium (default)** and select **Add Medium**.



The New Medium/Media Query dialog opens.


3. Select **Media Query**. Extra fields are shown in the dialog.


- You can enter your media query directly in the text field (as long as you use valid media query syntax). Alternatively, you can use the fields below it to choose criteria for the media query; the text field is populated accordingly.




For more information about creating custom media queries and the options available in the New Medium dialog, see:

<http://www.w3.org/TR/css3-mediaqueries/>

 **NOTE** Some fields in this dialog will change depending on the options you choose in other fields.

- Click **OK**.
- Click  to save your work.

 **NOTE** You can create and save a new custom media query from a cursor position on the layout resizer slider bar.

# Selecting Mediums and Media Queries

When you are editing a stylesheet, you can select a medium or media query in order to edit settings for it. You can also select a medium or media query when you are viewing or editing a content file.

## When Editing a Stylesheet

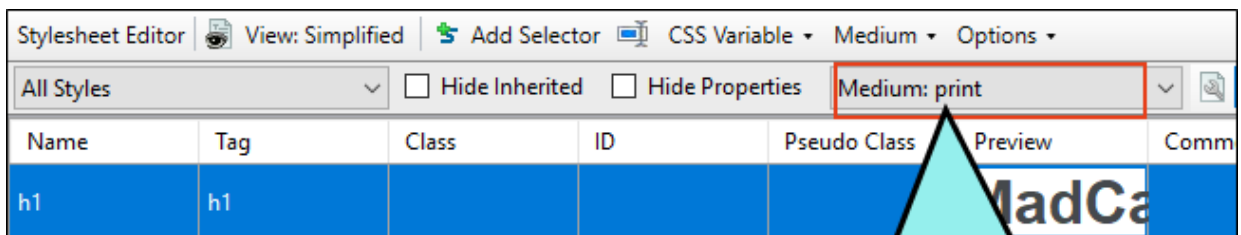
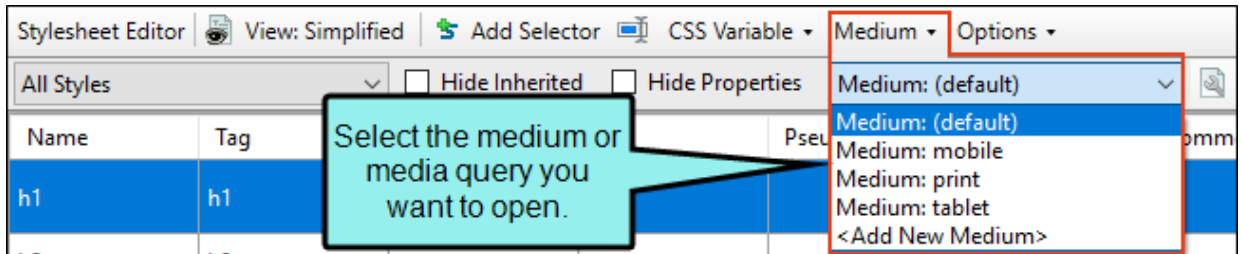
You can select a medium or media query when editing styles in either the Stylesheet Editor or the Table Style Editor. This lets you place style settings in that medium or media query so that the look and/or behavior is unique for a particular target or screen when the output is generated.

In the Simplified view of the Stylesheet Editor, as well as in the Table Style Editor, selecting a medium or media query does not change the editor, except to indicate the chosen medium or media query in the drop-down field. Whichever medium or media query appears in that field is the one you are changing when you make modifications to styles in the editor.

But if you are working in the Advanced view of the Stylesheet Editor, selecting a medium or media query opens a pane in the editor dedicated to that medium or media query. You then click inside the appropriate pane and make your changes to that medium or media query. In addition, you can have multiple mediums and media queries open at same time. See "Multiple Medium View" on page 301.

# How to Select a Medium or Media Query in the Stylesheet Editor (Simplified View)

1. From the Content Explorer, open the stylesheet that you want to modify.
2. Click in the **Medium** drop-down and select the appropriate medium or media query.



3. Edit the styles for that medium or media query as necessary.

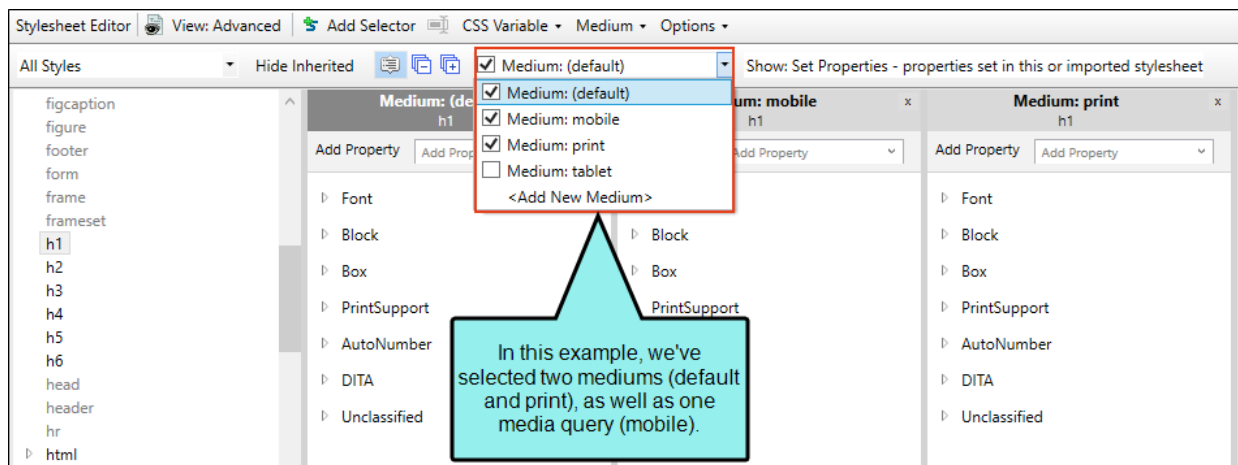


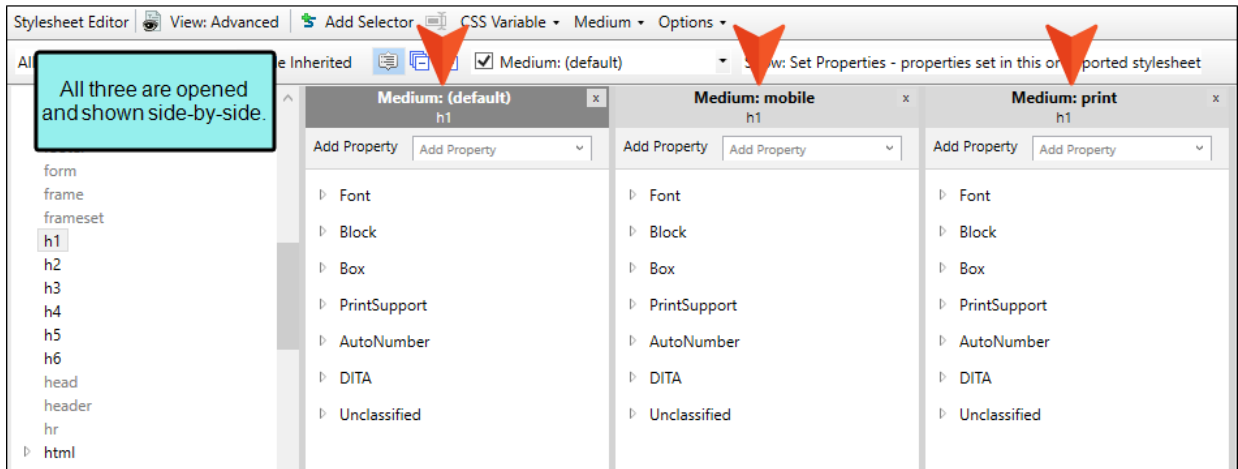
# How to Select a Medium or Media Query in the Stylesheet Editor (Advanced View)

1. From the Content Explorer, open the stylesheet that you want to modify.
2. Click in the **Medium** drop-down and select the appropriate medium or media query.

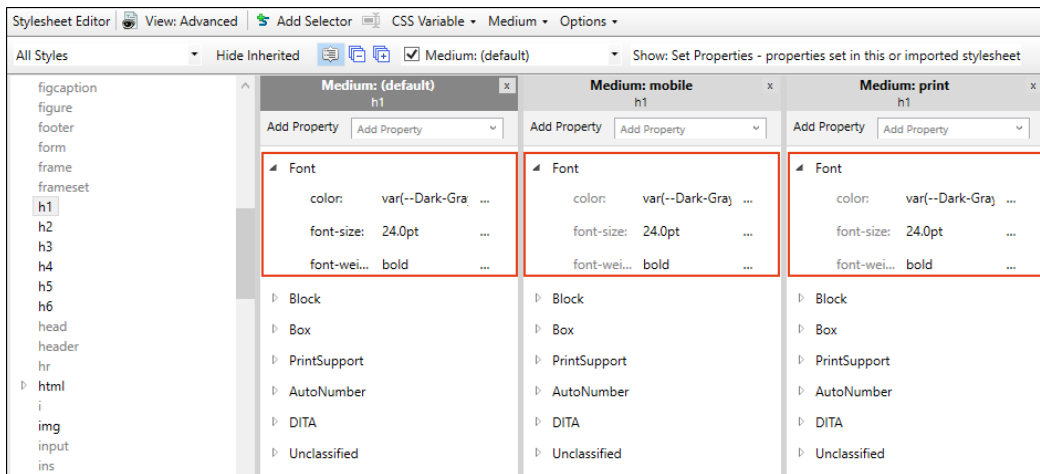
When choosing a medium/media query from the drop-down, you can click on the check mark or the name of the medium/media query. What happens next depends on how many mediums/media queries are currently open in the editor.

- **If One Medium or Media Query is Open** If only one medium/media query is open and you click a *check box* next to another medium/media query, that second medium/media query will open next to the first one. However, if you select the *name* of the medium/media query in the drop-down, it will open and the first medium/media query will close.
- **If Multiple Mediums or Media Queries are Open** If two or more mediums/media queries are open, the next medium/media query you select will open next to the others. This is true whether you select the check box or the name of the medium/media query.

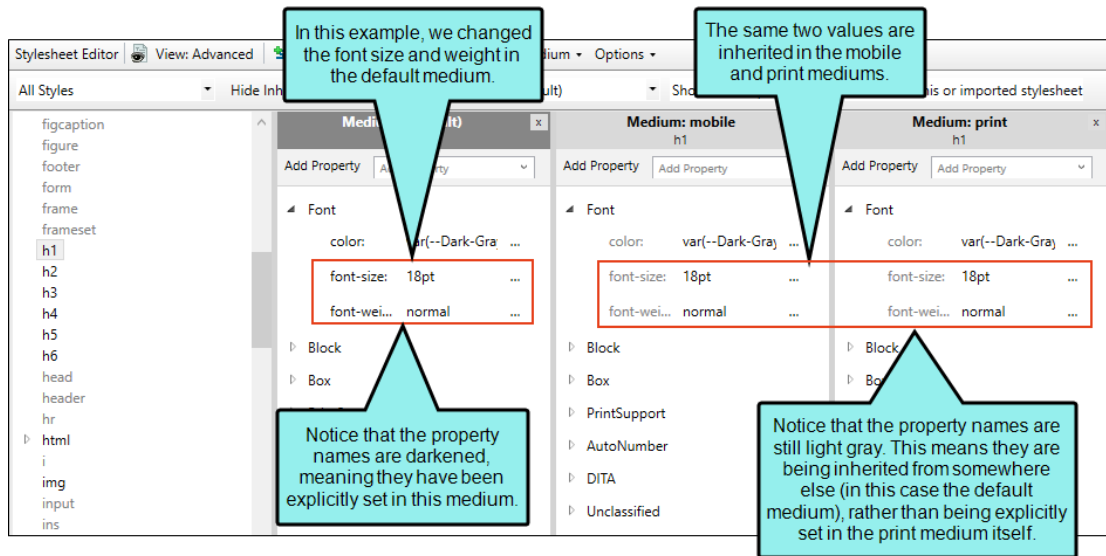




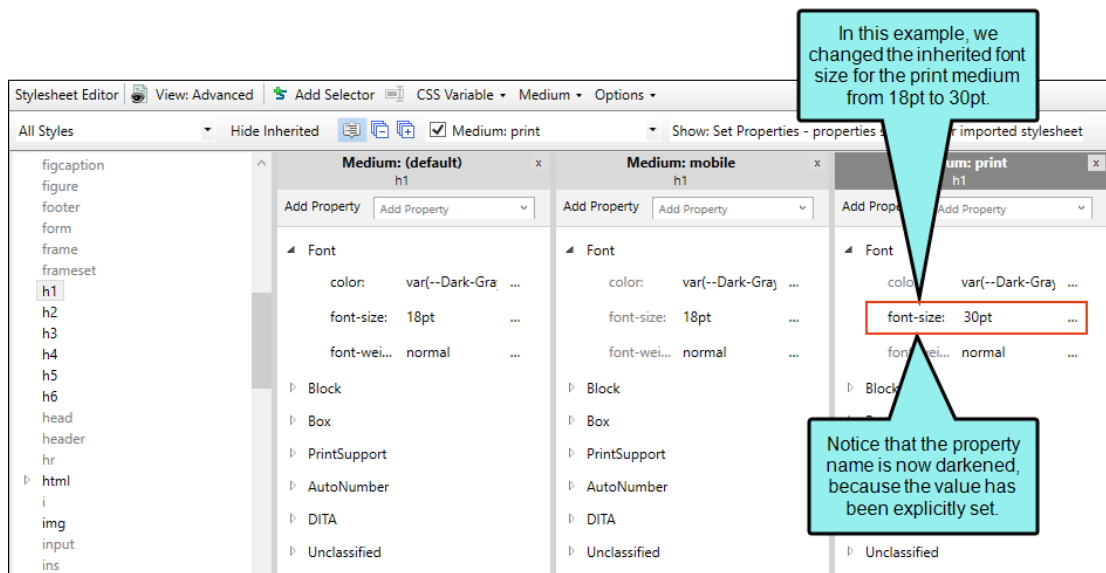
When you expand a property group in one medium or media query, the same property group also expands in any others that are open.



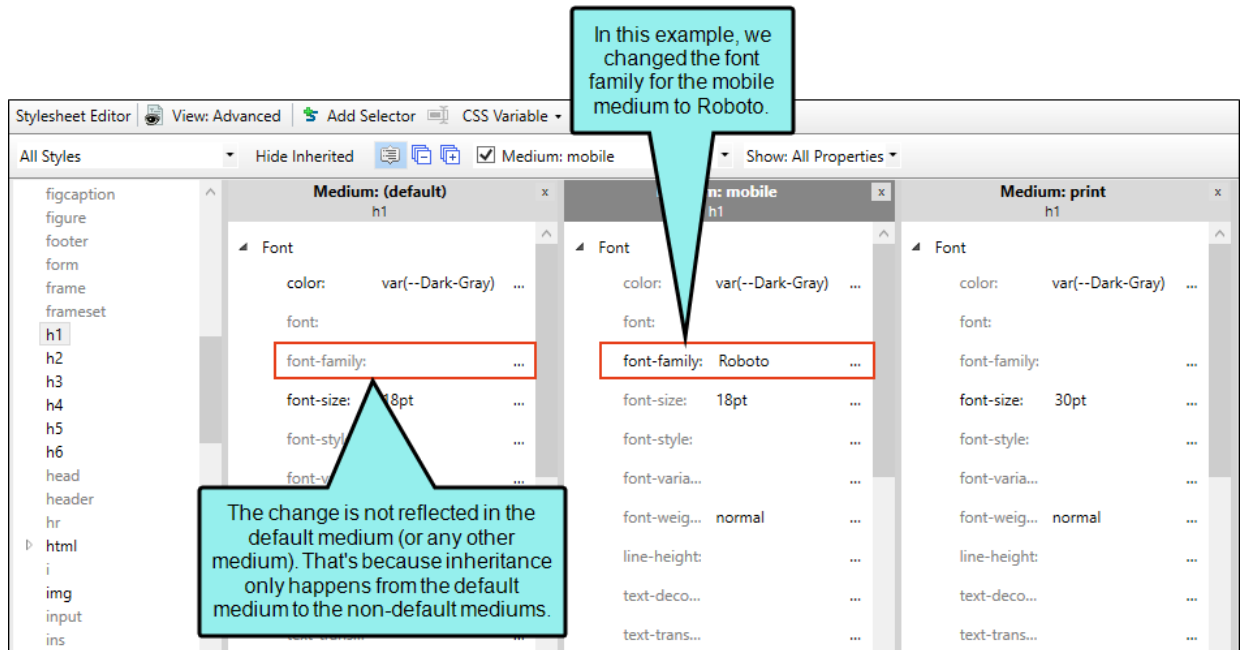
If you make a change in default medium, you see it applied also to the other mediums and media queries, because they inherit whatever is added in the default medium.



Of course, you can override any inherited property value in a specific medium or media query.



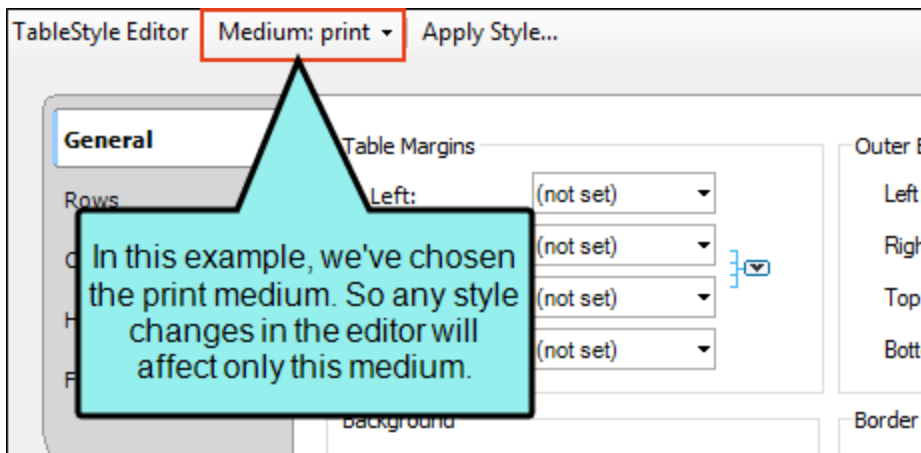
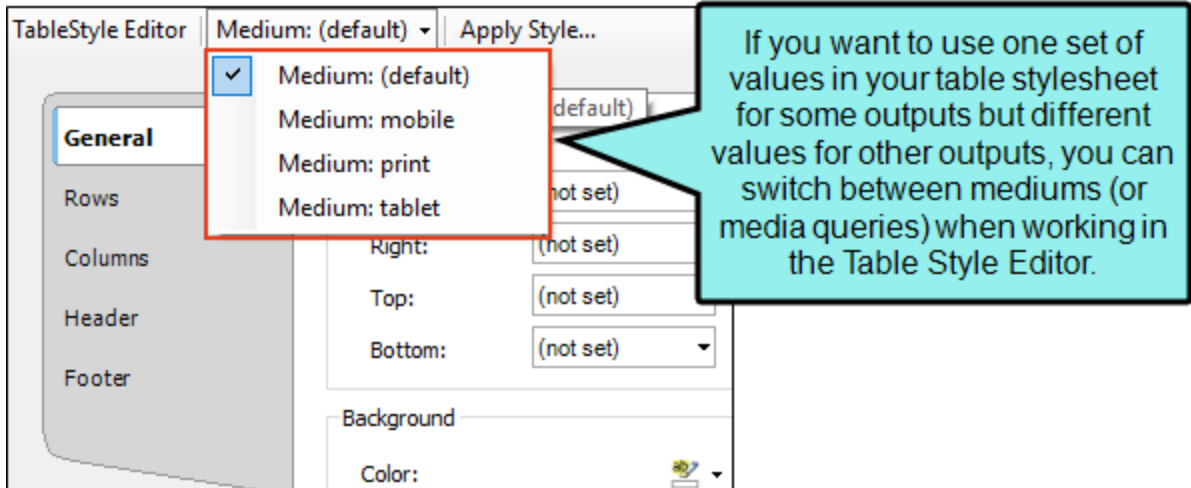
If you make a change in a non-default medium or in a media query, you see it only in that place.



3. Edit the styles for that medium or media query as necessary.

## How to Select a Medium or Media Query in the Table Style Editor

1. From the Content Explorer, open the table stylesheet.
2. Click in the **Medium** drop-down and select the appropriate medium or media query.



3. Edit the styles for that medium or media query as necessary.

# When Viewing or Editing Content

You can also select a medium or media query in the XML Editor when viewing and editing content.

## Mediums and Layout Modes

Flare provides multiple layout modes when working in the XML Editor: Web Layout, Web Layout (Tablet), Web Layout (Mobile), and Print Layout. This lets you see your content in the format that you are most likely concerned about at the moment.

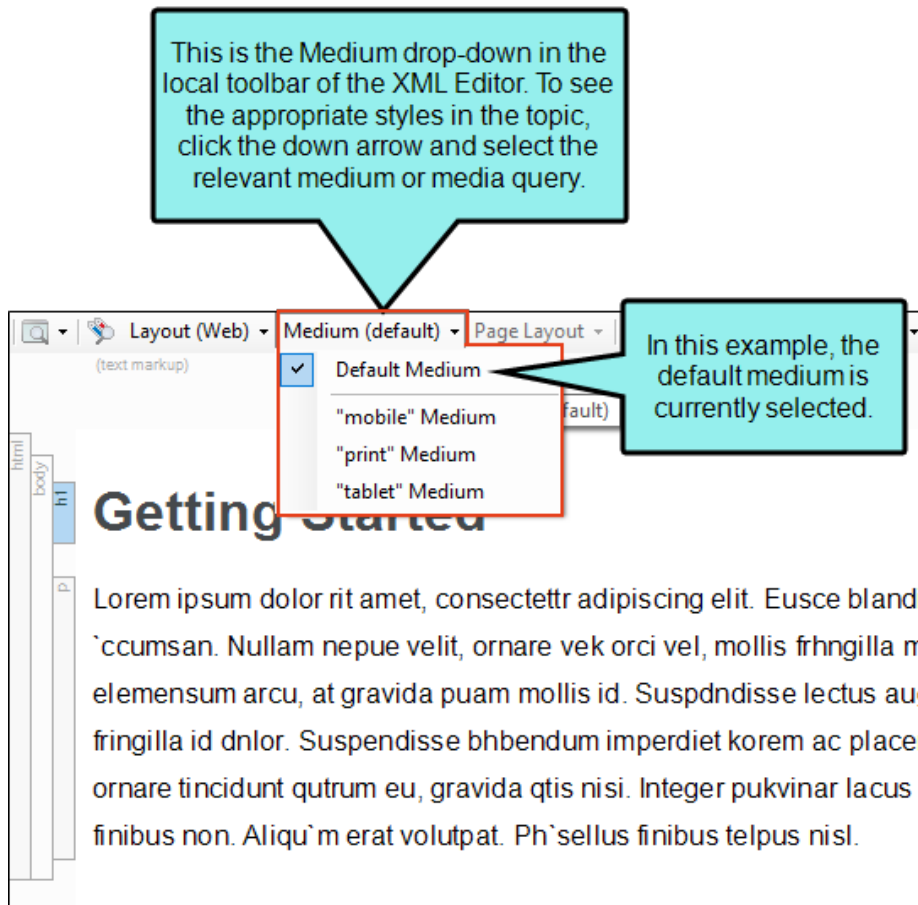
The default and print mediums are tied to the Web Layout and Print Layout modes, respectively. The tablet and mobile media queries are tied to the Web Layout (Tablet) and Web Layout (Mobile) modes, respectively.

☆ **EXAMPLE** You are using the Web Layout mode, so the XML Editor displays the styles from the default medium. But if you switch to Print Layout mode, the XML Editor automatically displays the styles from the print medium. And if you choose the Web Layout (mobile) mode, the XML Editor adjusts to show the topic as if it were displayed on a mobile device.

The key is to select the layout first. If you select the medium or media query, the layout will *not* automatically change as well.

## How to Select a Medium or Media Query in the XML Editor

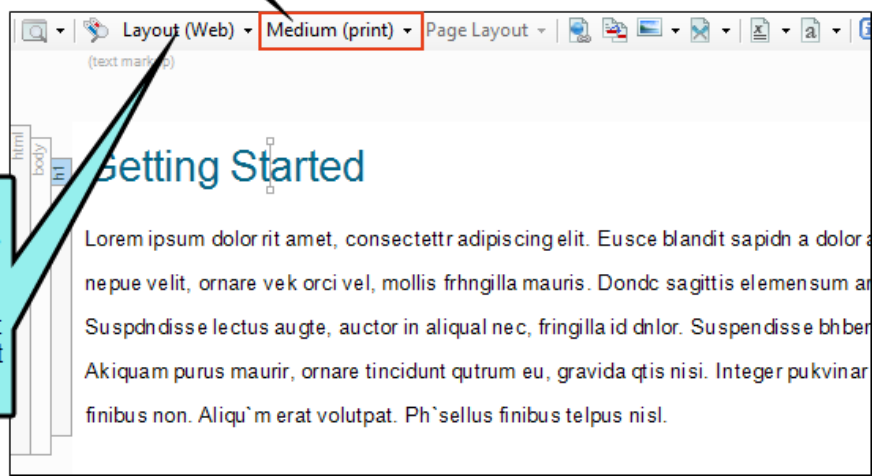
1. Open the content file.
2. In the local toolbar, click in the **Medium** drop-down and select the appropriate medium or media query.



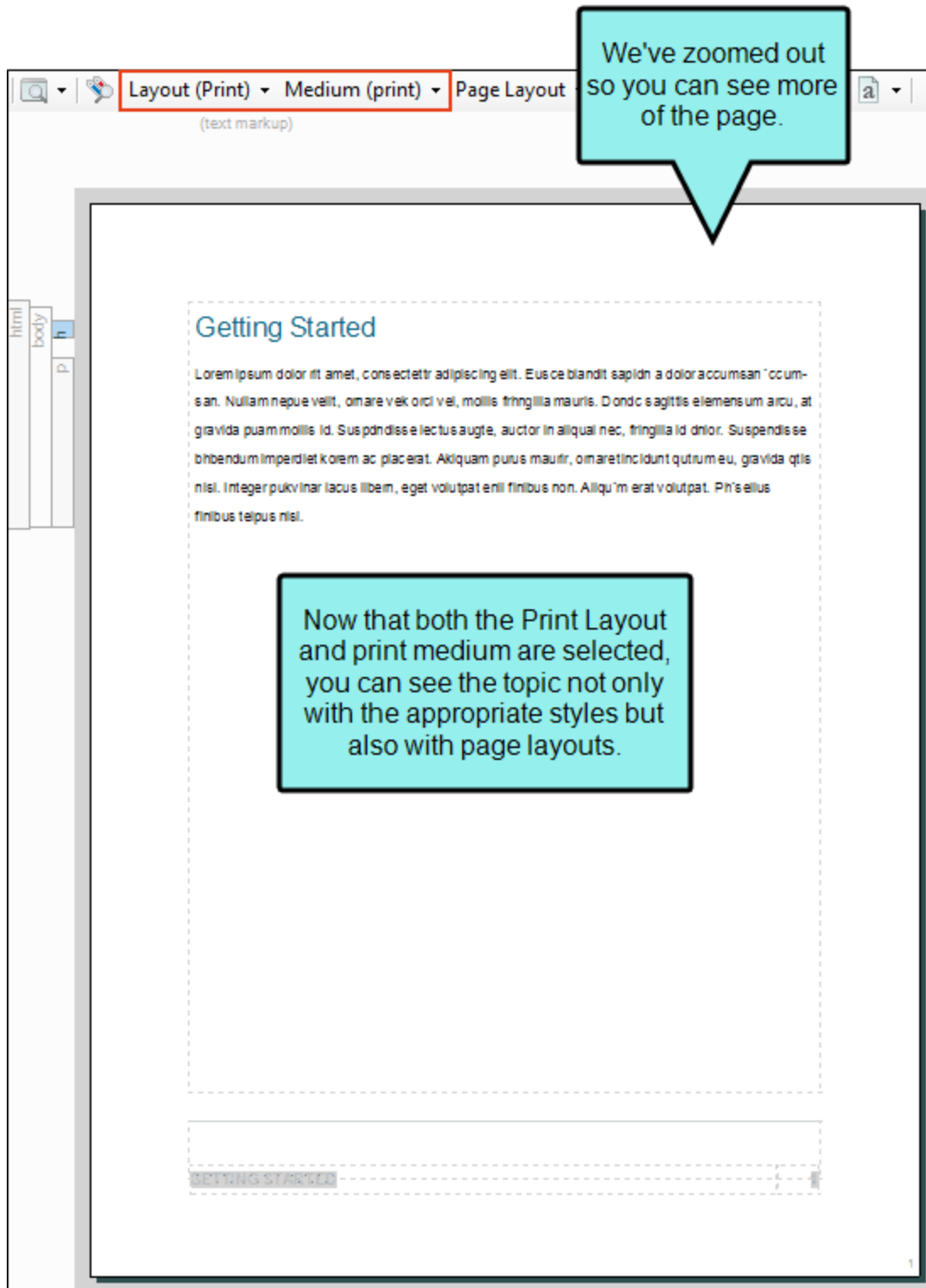
The look of the content changes to reflect the style settings for that medium or media query.

We've selected the print medium. Notice how the look of the content changed below.

Also notice that the Web Layout mode is selected. This layout mode is intended for online outputs, so to get a better feel for the topic in print output, you can switch to Print Layout mode.








This feature is simply intended to show you how particular styles make a topic look. By using this feature, you are not telling Flare to use that medium in the output for that topic. The way to do that is to associate the medium with a target. See "Associating a Medium With a Target" on the next page.

## What's Noteworthy?

 **NOTE** To access the mediums and media queries provided by Flare (print, tablet, mobile), you might need to make sure the **Hide Inherited** option in the local toolbar of the Stylesheet Editor is *not* selected. However, this is not necessary once you make an explicit change in one of those mediums or media queries; after that, it will show up in the Medium drop-down whether you use the Hide Inherited option or not.

# Associating a Medium With a Target

After you decide on a medium for an output, you need to associate it with the target. After you build the target, the medium will be used to display the correct style settings in the output.

## How to Associate a Medium With a Target

1. From the Project Organizer, open the target.
2. In the Target Editor, select the **Advanced** tab.
3. Click the drop-down arrow in the **Medium** field, and select the medium that you want to associate with the target.

 **NOTE** Only mediums can be selected. Media queries are not available from this field.

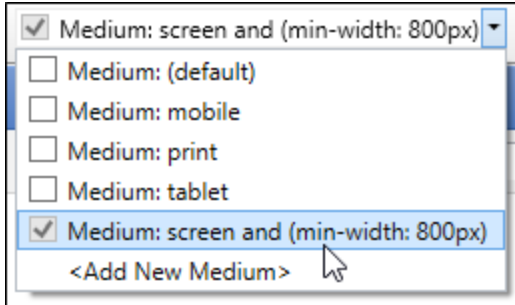
4. Click  to save your work.

## Renaming Mediums and Modifying Media Queries

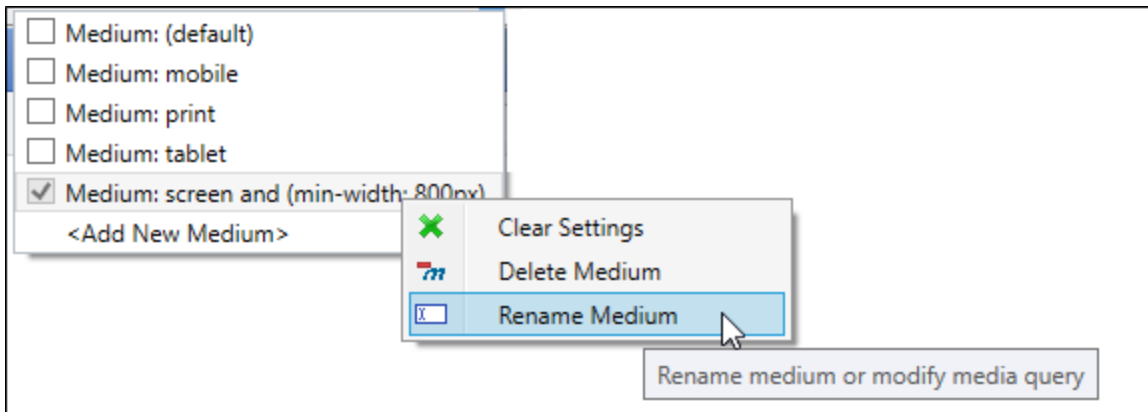
You can rename *custom* mediums and modify media queries without having to use the Internal Text Editor. However, you cannot rename *factory* mediums and media queries.

# How to Rename Mediums and Modify Media Queries

1. Open the Stylesheet Editor in the Advanced view.
2. Click the **Medium** drop-down and right-click the custom medium or media query you want to rename and/or modify.



3. From the context menu select **Rename Medium**.



The Rename Medium/Modify Media Query dialog opens.

4. If you selected a medium, change the name. If you selected a media query, you can make modifications to any of the settings, thus renaming it.
5. Click **OK**.

# MadCap-Specific Styles

In addition to the many standard styles from W3C, you might notice several unique-looking tags that begin with the word "MadCap" (e.g., MadCap|footnote, MadCap|toggler).

These special styles have been added to the Flare user interface in order to support some of the unique features available only in MadCap Software products.

Following is a list of MadCap-specific styles. For more details and steps for each, see the online Help.

### **MadCap|annotation**

Modifies the look of content to which an annotation (i.e., internal topic comment) points. For example, you might want annotated text to be displayed in the XML Editor with red font and a yellow background. This does not change the text as it will be shown in the output, but rather only as it is displayed in the XML Editor for authors. When an annotation is inserted in a content file, the MadCap:annotation tag includes the comment's creation date, user name and initials (as set in the File > Options dialog, Review tab) of the person who created or edited it, and the comment text.

---

### **MadCap|bodyProxy**

Modifies the look of the "container" holding topic content. For example, you might edit this style to add a border around all topic content.

---

### **MadCap|breadcrumbsProxy**

Modifies the look of breadcrumbs in online output.

---

### **MadCap|centralAccountProxy**

This style displays in the interface due to Flare's schema. However, it doesn't have a function, so you can ignore it. To control the look of the Central account link added via a proxy, you can use a skin component.

---

### **MadCap|codeSnippet**

Modifies the look of the entire code snippet block that has been inserted in the XML Editor.

---

### **MadCap|codeSnippetBody**

Modifies the look of the code snippet text, as well as the line numbers and vertical border to the right of the numbers.

---

### **MadCap|codeSnippetCaption**

Modifies the look of the caption used for the code snippet.

---

### **MadCap|codeSnippetCopyButton**

Modifies the look of the copy button link that can be added to code snippets for HTML5 output. If you want to change the word "Copy" to something else, you can edit the mc-label property.

---

**MadCap|concept**

Modifies the look of concepts that have been inserted in the XML Editor (when markers are turned on). This does not affect the output.

---

**MadCap|conceptLink**

Modifies the look (e.g., font, color, wording) of a concept (See Also) link heading. When you do this, the style changes for all concept links in any topics in your project.

---

**MadCap|conceptLinkControlList**

Modifies the look of the entire list (<ul> element) when concept links are displayed in a list, rather than in a popup.

---

**MadCap|conceptLinkControlListItem**

Modifies the look of individual items in the list (<li> elements) when concept links are displayed in a list, rather than in a popup.

---

**MadCap|conceptLinkControlListItemLink**

Modifies the look of links in the list (<a> elements) when concept links are displayed in a list, rather than in a popup.

---

**MadCap|conceptsProxy**

Modifies the look of the "container" holding a generated list of concepts.

---

**MadCap|conditionalText**

Modifies the look of content in the XML Editor that has a condition tag applied to it. For example, you might want conditioned content to stand out with a larger font so you can easily spot it while editing content. This does not affect the output.

---

**MadCap|correctFeedback**

Modifies the appearance of content that is shown as feedback when the eLearning question is answered correctly.

---

### **MadCap|dropDown**

Modifies the entire container holding a drop-down effect, including the image that is shown when a drop-down effect is open or closed.

---

### **MadCap|dropDownBody**

Modifies content displayed when users open a drop-down effect.

---

### **MadCap|dropDownHead**

Modifies the text in the first paragraph of a drop-down effect (i.e., the paragraph where the drop-down link is located).

---

### **MadCap|dropDownHotspot**

Modifies the specific text that you select in the first paragraph of a drop-down effect to serve as the link for opening the drop-down body. If you do not select specific text in the first paragraph to serve as the hotspot, the entire first paragraph is used as the hotspot.

---

### **MadCap|eLearningToolbarProxy**

Modifies the look of the container holding the eLearning Toolbar for adding navigation buttons to topics.

---

### **MadCap|endnoteBlock**

Modifies the container (or block) holding individual endnote comments. For example, use this if you want to add a border around each endnote comment created from an Endnotes proxy.

---

### **MadCap|endnotesBlock**

Modifies the container (or block) holding all endnote comments. For example, use this if you want to add a border around the collection of all endnote comments created from an Endnotes proxy.

---

### **MadCap|endnotesProxy**

Modifies the appearance of the text portion of the Endnotes proxy.

---

### **MadCap|equation**

Modifies the appearance of all equations.

---



**MadCap|expanding**

Modifies the entire container holding an expanding text effect, including the image that is shown when an expanding text effect is open or closed.

---

**MadCap|expandingBody**

Modifies the expanded text portion of an expanding text effect (i.e., the area that is displayed or hidden when users click the hotspot link).

---

**MadCap|expandingHead**

Modifies the hotspot portion of an expanding text effect.

---

**MadCap|faqProxy**

Modifies the FAQ proxy container holding micro content.

---

**MadCap|footnote**

Modifies both the footnote number (or symbol) where it is inserted in the topic, as well as the number and accompanying comment text (at the bottom of the page, or wherever else you specify its location).

---

**MadCap|footnoteBlock**

Modifies the container (or block) holding individual footnote comments. For example, use this if you want to add a border around each footnote comment on a page.

---

**MadCap|footnotesBlock**

Modifies the container (or block) holding all footnote comments. For example, use this if you want to add a border around the collection of all footnote comments on a page.

---

**MadCap|glossaryProxy**

Modifies the look of the "container" holding a generated glossary.

---

**MadCap|glossaryTerm**

Modifies the look of glossary term links.

---

## MadCap|helpControlList

Modifies the look of the *entire list* (<ul> element) when Help control links are displayed in a list, rather than in a popup. This is a general style that controls all three types of Help control links—concept, keyword, and related topics. Alternatively, you can set properties on each specific style—MadCap|conceptLinkControlList, MadCap|keywordLinkControlList, or MadCap|relatedTopicsControlList.

---

## MadCap|helpControlListItem

Modifies the look of *individual items in the list* (<li> elements) when Help control links are displayed in a list, rather than in a popup. This is a general style that controls all three types of Help control links—concept, keyword, and related topics. Alternatively, you can set properties on each specific style—MadCap|conceptLinkControlListItem, MadCap|keywordLinkControlListItem, or MadCap|relatedTopicsControlListItem.

---

## MadCap|helpControlListItemLink

Modifies the look of *links in the list* (<a> elements) when Help control links are displayed in a list, rather than in a popup. This is a general style that controls all three types of Help control links—concept, keyword, and related topics. Alternatively, you can set properties on each specific style—MadCap|conceptLinkControlListItemLink, MadCap|keywordLinkControlListItemLink, or MadCap|relatedTopicsControlListItemLink.

---

## MadCap|helpControlMenu

Modifies the look of links (i.e., menu items) that users see when they click a concept link, keyword link, or related topics control. This style is grouped with the "Dynamic Effects Styles" (which you can select from the drop-down list in the upper-left corner of the Stylesheet Editor). This particular style controls the *entire list* when you are using the *popup menu* method for displaying Help control links.

---

## MadCap|helpControlMenuItem

Modifies the look of links (i.e., menu items) that users see when they click a concept link, keyword link, or related topics control. This style is grouped with the "Dynamic Effects Styles" (which you can select from the drop-down list in the upper-left corner of the Stylesheet Editor). This particular style controls the *individual list items* when you are using the *popup menu* method for displaying Help control links.

---

**MadCap|incorrectFeedback**

Modifies the appearance of content that is shown as feedback when the eLearning question is answered incorrectly.

---

**MadCap|indexProxy**

Modifies the look of the "container" holding a generated index for print-based output.

---

**MadCap|keyword**

Modifies the look of index keywords that have been inserted in the XML Editor (when markers are turned on). This does not affect the output.

---

**MadCap|keywordLink**

Modifies the look (e.g., font, color, wording) of a keyword link heading. When you do this, the style changes for all keyword links in any topics in your project.

---

**MadCap|keywordLinkControlList**

Modifies the look of the entire list (<ul> element) when keyword links are displayed in a list, rather than in a popup.

---

**MadCap|keywordLinkControlItem**

Modifies the look of individual items in the list (<li> elements) when keyword links are displayed in a list, rather than in a popup.

---

**MadCap|keywordLinkControlItemLink**

Modifies the look of links in the list (<a> elements) when keyword links are displayed in a list, rather than in a popup.

---

**MadCap|knowledgeProxy**

Modifies the Knowledge proxy container holding micro content.

---

**MadCap|listOfProxy**

Modifies the look of the "container" holding a generated list of elements.

---

**MadCap|menuProxy**

Modifies the look of a menu.

---

**MadCap|microContent**

Related to micro content that you create. However, in this version, modifying the style will have no effect on the output.

---

**MadCap|miniTocProxy**

Modifies the look of the "container" holding a generated mini-TOC.

---

**MadCap|model3D**

Modifies the look of 3D models.

---

**MadCap|multipleChoice**

Modifies the look of question sections that have been inserted in the XML Editor. This consists of the MadCap|question, MadCap|multipleChoiceItem, MadCap|correctFeedback, MadCap|incorrectFeedback, and MadCap|submitButton sections.

---

**MadCap|multipleChoiceItem**

Modifies the look of the answer in the XML Editor that has been inserted within the MadCap|multipleChoice section.

---

**MadCap|namedDestination**

This style does not have any relevant style properties. Named destinations are used in PDF output to label certain locations in the document. These locations can then be linked to directly from another PDF document.

---

**MadCap|pageBreak**

Page breaks can be added to a document to break the content across pages in printed output. You can change the page break's behavior to set whether it breaks before or after content.

---

**MadCap|pageFooter**

Modifies the look of the content contained in a page footer used in template pages for Microsoft Word output.

---

**MadCap|pageHeader**

Modifies the look of the content contained in a page header used in template pages for Microsoft Word output.

---

**MadCap|popup**

Modifies the look of the container holding a text popup link. For example, you can modify this style to place a border around the link.

---

**MadCap|popupBody**

Modifies the popup text portion of an popup text effect (i.e., the area that is displayed or hidden when users click the hotspot link).

---

**MadCap|popupHead**

Modifies the hotspot portion of a popup text effect.

---

**MadCap|promotionProxy**

Modifies the Promotion proxy container holding micro content.

---

**MadCap|qrCode**

Modifies the appearance of all QR codes.

---

**MadCap|question**

Modifies the look of the question in the XML Editor that has been inserted within the MadCap|multipleChoice section.

---

**MadCap|relatedTopics**

Modifies the look (e.g., font, color, wording) of a related topics link heading. When you do this, the style changes for all related topics links in any topics in your project.

---

**MadCap|relatedTopicsControlList**

Modifies the look of the entire list (<ul> element) when related topics are displayed in a list, rather than in a popup.

---

### MadCap|relatedTopicsControlItem

Modifies the look of individual items in the list (<li> elements) when related topics are displayed in a list, rather than in a popup.

---

### MadCap|relatedTopicsControlItemLink

Modifies the look of links in the list (<a> elements) when related topics are displayed in a list, rather than in a popup.

---

### MadCap|relationshipsHeading

Modifies the look of headings used in relationship links. There are three classes of this style that you can edit. If you edit the main MadCap|relationshipsHeading style, the look of all of the classes are affected. However, you can also edit the look of each class if you want.

- **concept** Modifies the look of the heading that is displayed above any concept links (i.e., any topics found in the "concept" column of the relationships table).
  - **reference** Modifies the look of the heading that is displayed above any reference links (i.e., any topics found in the "reference" column of the relationships table).
  - **task** Modifies the look of the heading that is displayed above task links (i.e., topics found in the "task" column of the relationships table).
- 

### MadCap|relationshipsItem

Modifies the look of link items created from a relationships table.

---

### MadCap|relationshipsProxy

Modifies the look of the "container" holding the content from a generated relationships table.

---

### MadCap|searchBarProxy

Modifies the look of search bar.

---

### MadCap|searchResultsProxy

Modifies the look of a custom search results page.

---

### **MadCap|section**

This style displays in the interface due to Flare's schema. However, it doesn't have a function, so you can ignore it.

---

### **MadCap|shortcut**

Modifies the look (e.g., font, color) of a shortcut control link. When you edit the style for a shortcut control, the style changes for all shortcut controls in any topics in your project.

---

### **MadCap|slide**

Modifies the look of the containers holding individual slides.

---

### **MadCap|slideshow**

Modifies the look of the container holding the entire slideshow. Keep in mind that if you have competing styles set, those set in the MadCap|slide style take precedence. For example, if you set the background on MadCap|slideshow to blue and the background on MadCap|slide to yellow, the background will be yellow. But if you then change the background for MadCap|slide to (default), the background will show as blue.

---

### **MadCap|slideshowBullet**

Modifies the look of the series of dots (or "bullets") used to navigate to specific slides in a slideshow. Keep in mind that if you choose to include thumbnail images, the MadCap|slideThumbnail style will be used instead.

---

### **MadCap|slideshowCaption**

Modifies the look of the caption at the bottom of the slide in a slideshow.

---

### **MadCap|slideThumbnail**

Modifies the look of the thumbnail image area at the bottom of the slide.

---

## MadCap|snippetBlock

Modifies the look of block snippets that have been inserted in the XML Editor. If you insert a snippet on a blank line in a topic, it is inserted as a block snippet (as opposed to a text snippet) and takes up all of the room so that no other content can be added.

Snippet blocks are best used for snippets that contain multiple paragraphs, lists, images, or specific formatting, or other large elements.

Using a snippet block (as opposed to a text snippet) style does not affect the output. If you want to make changes to a snippet that affect the output, you can open the snippet and modify it in the XML Editor, just as you would edit a regular topic.

---

## MadCap|snippetText

Modifies the look of text snippets that have been inserted in the XML Editor. If you insert a snippet on a line where other content exists, it is inserted as a text snippet, as opposed to a block snippet.

Text snippets are best used for shorter snippets, such as a few sentences or words. You might add text snippets in the middle of a paragraph, in a table, or within other snippets.

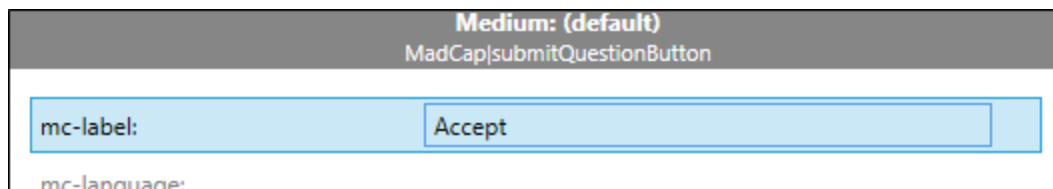
Using a text snippet (as opposed to a block snippet) style does not affect the output. If you want to make changes to a snippet that affect the output, you can open the snippet and modify it in the XML Editor, just as you would edit a regular topic.

---

## MadCap|submitButton

Modifies the look of the submit button that can be added to eLearning question sections for HTML5 output.

In the stylesheet, select the **Unclassified > mc-label** property to change the label for the button.



## MadCap|testResultsProxy

Modifies the look of the container holding the Test Results when customizing the eLearning test results pages.

---



## MadCap|tocProxy

Modifies the look of the "container" holding a generated TOC for print-based output.

---

## MadCap|toggler

Modifies the hotspot portion of a toggler.

---

## MadCap|topicToolbarProxy

Modifies the look of the "container" holding a generated topic toolbar. For HTML5 outputs, the Topic Toolbar proxy will use whatever settings are specified in a Topic Toolbar skin component (if you have added one to your project), overriding any buttons you may have selected directly in the proxy. If you have not associated a Topic Toolbar skin component with the proxy, Flare will just use the first one it finds in your project. However, for outputs using Standard and Mobile skins, the settings in the proxy take precedence over anything you may have set on the Toolbar tab in the Skin Editor.

---

## MadCap|variable

Modifies the look of variables in the XML Editor and in generated output files.

---

## MadCap|xref

Modifies the look and format in cross-references. This is the main style used for basic cross-references that you create.

In addition to creating your own custom classes of the main MadCap|xref style, you can also edit the following classes to control the look of page numbers in various places for print-based output.

- **ConceptPageNumber** Modifies the look of page numbers in a generated list of concepts.
- **IndexPageNumber** Modifies the look of the page numbers in a generated index.
- **ListOfPageNumber** Modifies the look of page numbers in a generated list of elements.
- **RelLinkPageNumber** Modifies the look of page numbers in a generated list of relationship links.
- **TOCPageNumber** Modifies the look of page numbers in a generated table of contents.

## APPENDIX B

---

# MadCap-Specific Properties

In addition to the many standard properties from W3C, you might notice several unique-looking ones that always start with "mc" (e.g., mc-footnote-format, mc-hyphenate).

These special properties have been added to the Flare user interface in order to support some of the unique features available only in MadCap Software products.

Following is a list of MadCap-specific tag properties. For more details and steps for each, see the online Help.

**mc-auto-number-class**

Determines the span class for the generated autonumber.

---

**mc-auto-number-format**

Opens the autonumber format dialog, which lets you enter an autonumber format or select an existing one.

---

**mc-auto-number-offset**

Determines the margin.

---

**mc-auto-number-position**

Determines where the autonumber will be positioned in relation to the paragraph.

---

**mc-breadcrumbs-count**

Determines the number of breadcrumbs to show (i.e., how many levels of file and folder names you wish to display from the TOC). For example, the default setting is 3, so 3 breadcrumbs would appear in the following manner: You are here: First Topic > Second Topic > Third Topic.

---

**mc-breadcrumbs-divider**

Determines the text or characters that appear between breadcrumbs.

---

**mc-breadcrumbs-prefix**

Determines the text or characters that appear before the breadcrumbs.

---

**mc-caption-continuation**

You can add continuation text to the end of table captions that span multiple pages in printed output (other than the caption appearing on the first page where a table occurs). Typically this text would be something like "(continued)."

---

**mc-caption-repeat**

You can use styles to repeat captions on tables that span multiple pages in print-based output.

---

### **mc-closed-image**

Lets you select an image to be shown next to the following types of links when they are in a "closed" state: drop-down text, expanding text, glossary term links, togglers.

---

### **mc-closed-image-alt-text**

Lets you set alternate text on the following types of links when they are in a "closed" state: drop-down text, expanding text, glossary term links, togglers.

---

### **mc-code-border**

Lets you control the look of the vertical border to the right of the numbers (if they are included) in a code snippet. Typically, this would be associated with the MadCap|codeSnippetBody style, or a class of it.

---

### **mc-code-lang**

Lets you select the coding language to be used by default with the associated style. Typically, this would be associated with the MadCap|codeSnippetBody style, or a class of it.

---

### **mc-column-count**

Lets you select the number of columns to be used for indexes in Microsoft Word output. This is the default setting for the column count, but it can be overridden on a specific Index proxy. For other print-based outputs, you can set the number of columns for an index in a page layout instead.

---

### **mc-community-features**

Specifies whether or not community features are enabled for a topic. This setting is applied to the html style in the stylesheet.

---

### **mc-concepts-list-headings**

If a concept proxy has been inserted into the topic, this property lets you choose whether the style to which it is applied will display headings in the list of concepts.

---

### **mc-conditions**

Normally you would apply a condition to a piece of content or a file. This property lets you also set conditions on styles, which you can then apply to content. This is simply another alternative and might be more efficient for some authors. You might even find that you use both methods in your projects.

---

### **mc-disable-glossary-terms**

Lets you choose whether glossary terms are converted to links automatically when they are found in topics. By default, the setting is turned on for h1 through h6 styles and all hyperlink tags. Setting the property to "True" disables the conversion of glossary terms to links. Setting the property to "False" enables the conversion of glossary terms to links.

---

### **mc-dita-type**

Lets you select the tag used for exporting the style in DITA. By default, this property is either not set or set to the style to which this property is applied. However, using this property, you can override the default settings. For example, you can set a p style to a table style for DITA export.

---

### **mc-exclude-action**

Displays the kind of "exclude" action tied to the condition tag. In most cases, the exclude action is "remove." However, in some instances, this action might be "unbind." For example, you might have the unbind action for a condition tag if you have applied the tag to a hyperlink and want the link to be removed from the text in some outputs, but you still want the text to be shown in those outputs.

When you set a condition tag on a style, you can optionally set an exclude action on the tag, as well.

---

### **mc-expandable**

Determines whether the glossary entries to which the designated style is applied are expandable.

---

### **mc-feader-type**

This property is used with the MadCap|pageFooter and MadCap|pageHeader styles when building Microsoft Word output. This property specifies which types of pages a page footer or header is displayed on (all, even pages, first page, odd pages). You can always override the page type used at the spot where a particular page footer or header proxy is inserted. This setting simply lets you control the default setting.

---

### **mc-float**

Determines where the text to which the style is applied will be displayed in relation to the paragraph.

---

### **mc-footnote-comment-format**

Determines the footnote format for the numbers in the comment. The default is the setting used in mc-footnote-format.

---

### **mc-footnote-comment-style**

Determines the span class for the footnote numbers inside the footnote comment. The default is the setting used in mc-footnote-style.

---

### **mc-footnote-format**

Determines the footnote format for numbers in the text. For example, you can change the format to uppercase alpha (e.g., A, B, C) or lowercase Roman (e.g., i, ii, iii). You can also replace the number with a symbol, such as an asterisk.

---

### **mc-footnote-number**

Determines where you can restart the footnote numbering at a certain location in the output (e.g., at the end of the next chapter or section).

---

### **mc-footnote-position**

Determines the location where the footnote comments are placed in the output (e.g., end of page, document, chapter, section, book).

---

### **mc-footnote-style**

Determines the span class for the footnote number.

---

### **mc-format**

Determines the style format for cross-references.

---

### **mc-heading-format**

Determines the style format for index headings.

---

### **mc-heading-level**

Determines which topic headings will be displayed at which level in the generated TOC. The higher the number, the lower in the hierarchy the heading will be displayed in the print TOC. If you select 0, the heading will not be included in the print TOC.

---

### **mc-help-control-display**

Lets you specify whether the default setting for Help control links should be "list" or "popup". Therefore, this property is used for the following styles: MadCap|conceptLink, MadCap|keywordLink, and MadCap|relatedTopics.

This can also be specified at the spot where you insert the Help control, but if you want all Help controls to automatically use the same display, you can edit this style property instead.

---

### **mc-hidden**

Determines whether a style class is hidden in the interface (such as from the Styles window pane or the ribbon). This does not hide the content itself.

You might want to use this if you have created a style but do not want it to be editable, or if you do not want it to be selectable in the interface (e.g., a rarely used company-mandated style that should not be edited).

Hiding a style is somewhat different from disabling a style. Although neither hidden nor disabled styles are visible in the interface, disabled styles can only be reenabled from the Disable Styles dialog, because they are also removed from the stylesheet. Hidden styles are still visible in the stylesheet, and are not counted among "disabled" styles in the Disable Styles dialog. Disabling a style is preferable to hiding a style if you want to "clean up" your stylesheet and only show the styles you regularly use.

---

### **mc-hide-bottom-ruling**

Lets you hide the bottom border on tables when crossing page breaks.

By default, if you have a table that crosses multiple pages in print-based output, the bottom border is shown before the table continues on the next page. However, you also have the option to hide the bottom border when the table continues on another page.

---

### **mc-hyphenate**

Determines the hyphenation rules on the selected style.

---

### **mc-hyphenate-maximum-adjacent-line-count**

Determines the maximum number of lines next to each other that are allowed to end with a hyphenated word.

---

### **mc-hyphenate-shortest-prefix**

Determines the minimum number of characters that must remain on the initial line when a word is hyphenated.

---

### **mc-hyphenate-shortest-suffix**

Determines the minimum number of characters that must be carried over to the second line when a word is hyphenated.

---

### **mc-hyphenate-shortest-word**

Determines the minimum number of characters that a word must have in order to be hyphenated.

---

### **mc-image**

Determines the image used for the selected style. These are used most often with help controls (i.e., concept links or related topic links) and navigation links (i.e., drop-down text, expanding text, or togglers).

---

### **mc-image-alt-text**

Lets you set alternate text on images for related topic links.

---



### **mc-image-position**

Determines how the image to which the style is applied is positioned, (i.e., right, left, center).

---

### **mc-image-spacing**

Determines the spacing around the images used for drop-down text, expanding text, shortcuts, and other dynamic effects.

---

### **mc-index-headings**

This property is used with the MadCap|indexProxy style when building print-based output that includes a generated index (i.e., an Index proxy). This property specifies whether the generated index should include headings above the alphabetical groups of entries. The default setting is "True," meaning that index headings will be included. However, you can always override this setting at the spot where a particular Index proxy is inserted. If you want all index proxies to not include headings by default, you can edit this style property, setting it to "False."

---

### **mc-italic-correction**

You can correct the spacing in a line when italic formatting is involved. Often, when you italicize a word in the middle of a sentence, the last letter of the italicized word appears to have less space behind it. This is typically due to the fact that the blank space immediately after the word is italicized and the word that comes after is not. Using italic correction, you can increase this space between the italicized word and the non-italicized word.

---

### **mc-label**

Determines the text labels for links. For example, the default setting for the MadCap|relationshipsHeading style is "Related Information", which is the text that appears in the output when you insert a concept link. However, you can use this property to change that text.

---

### **mc-language**

Determines the language used when spell checking. You can enter *inherit*, *none*, or a language code (e.g., *en-us*).

A benefit to assigning a language setting to a certain style is if you want to use this property to prevent Flare from spell checking a certain style (e.g., styles used for code snippets).

---

### **mc-leader-align**

Determines how the page numbers are aligned at the end of the leader. The default setting is right alignment.

---

### **mc-leader-format**

Determines the material that is shown between a topic name and the page number where it can be found. The default setting is a series of dots.

---

### **mc-leader-indent**

Determines the amount of space from the end of the list entry to the start of the leader.

---

### **mc-leader-offset**

Determines the amount of space from the end of the leader to the page number following it.

---

### **mc-multiline-indent**

Determines the amount of indentation for all lines of text following the initial indentation.

---

### **mc-open-image**

Lets you select an image to be shown next to the following types of links when they are in an "open" state: drop-down text, expanding text, glossary term links, togglers.

---

### **mc-open-image-alt-text**

Lets you set alternate text for related topic links.

---

### **mc-output-support**

Lets you choose which kinds of outputs (e.g. all, all online, all print, or a specific output type) support a particular style. You typically will not need or want to change the value in this field, but there might be times when you do. For example, instead of using a skin for an index in online output, you could use an index proxy, which is primarily intended for print-based output. Therefore, you could change the mc-output-support property for the MadCap|indexProxy style from all-print to all.

---

### **mc-page-number-format**

If you want the page numbers for this template page to take on a particular format (e.g., numbers without dashes, numbers with dashes, uppercase alpha, lowercase alpha, uppercase Roman, lowercase Roman), select it from this field.

---

### **mc-page-number-start-value**

If you want the first printed page associated with this template page to start at a particular page number, enter that number in this field.

---

### **mc-pagenum-display**

Determines how page numbers are displayed in TOC entries for print output.

---

### **mc-page-type**

Select the pages to which these settings will be applied (All Pages, Odd Pages, Even Pages, First Page). The default is to apply the settings to all pages using the template page. If you want all of the pages in your output to share the same settings (page size, margins, etc.), you only need one Body proxy in the template page. However, if you want different pages to have different settings, you can add a Body proxy page for each configuration. Then, edit the settings for a particular Body proxy for specific pages (e.g., odd pages), and use this field to specify which pages the settings pertain to.

---

### **mc-popup-height**

Determines the initial height of the popup window.

---

### **mc-popup-width**

Determines the initial width of the popup window.

---

### **mc-printer-page-custom-height**

If you selected "(custom)" from the mc-page-size field, enter a specific height for the page. Enter a value in the lower-left area and choose from several different units of measurement (points, pixels, centimeters, etc.) in the lower-right area.

---

### **mc-printer-page-custom-width**

If you selected "(custom)" from the mc-page-size field, enter a specific width for the page. Enter a value in the lower-left area and choose from several different units of measurement (points, pixels, centimeters, etc.) in the lower-right area.

---

### **mc-printer-page-footer-margin**

Specifies the margin between the footer and the page bottom.

---

### **mc-printer-page-header-margin**

Specifies the margin between the header and the page top.

---

### **mc-printer-page-margin-bottom**

Specifies the bottom margin for the printer page.

---

### **mc-printer-page-margin-left**

Specifies the left margin for the printer page.

---

### **mc-printer-page-margin-right**

Specifies the right margin for the printer page.

---

### **mc-printer-page-margin-top**

Specifies the top margin for the printer page.

---

### **mc-printer-page-orientation**

The orientation of the printer page.

---

### **mc-printer-page-size**

Select a standard size for the page (e.g., Letter, Legal) or select **(custom)** to specify a non-standard width and height.

---

### **mc-redacted**

This property lets you set redaction on a specific style. Redaction occurs when content is permanently eliminated from a printed or electronic document. In place of that content, end users will see black rectangles that indicate where the original content was found. The property can be set in the following ways.

- **inherit** The style inherits the redaction setting from the type of tag in which it is placed.
  - **none** The style is not redacted.
  - **redacted** The style is redacted.
- 

### **mc-reference-initial- separator**

Determines the text or character that immediately precedes page references in the index. The default is blank, but you might change it to say " pages " or "—".

---

### **mc-reference-separator**

Determines the text or character between page references in the index. The default is a comma (e.g., 1, 2, 3).

If your page references span a range of pages (e.g., 8–10) the dash is not replaced.

---

### **mc-required-question**

Specify to restrict navigation for a learner. If set to "true," then a question is required to answer before advancing to the next question in an eLearning course. If set to "false," then the navigation is not restricted (i.e., it is optional).

---

### **mc-short-line**

Specify the length at which a line is considered "short," and therefore the short line settings come into play. If you do not select a length, a line is considered short if it has 10 or fewer characters.

---

---

**mc-short-line-loosen-end-length**

Specify the length at which a line is considered "long enough." For example, let's say you have specified 8 characters as the length of a short line, and you have specified 15 characters as the length of a long line. In that case, Flare will not allow the final line in a paragraph to have only 8 characters, and when you type content so that the text wraps around to a new line, the paragraph is automatically adjusted so that the new line always starts at a minimum of 15 characters.

---

**mc-short-line-loosen-maximum**

Specify the maximum number of pixels to loosen during short line elimination.

---

**mc-short-line-method**

Determines the method by which short line elimination will function.

---

**mc-short-line-step**

Specify the number of pixels of kerning should be used incrementally on the entire paragraph until the desired effect is achieved. For example, if you enter .5, the kerning will occur in increments of .5 pixels.

---

**mc-short-line-tighten-maximum**

Specify the maximum number of pixels to tighten during short line elimination.

---

**mc-template-page**

Lets you use multiple template pages for different online topics. For example, you might want most topics in your project to use the same template page (to display the same text at the bottom of each topic). However, maybe you want a select few other topics to use a different template page.

It is recommended that you use the Topic Properties to select the template page, which automatically adds mc-template-page into the code of the topic. However, you can do this manually with the mc-template-page property if you want.

---

### **mc-term-display**

Determines how the style will display glossary term links. You can select from expanding text, a hyperlink, or a popup.

When you first create a glossary term and definition, you can specify the style to be used for glossary term links. The popup display is associated by default with the primary MadCap|glossaryTerm style, as well as with the MadCap|glossaryTermPopup style. The expanding display is associated by default with the MadCap|glossaryTermExpanding style. And the hyperlink display is associated by default with the MadCap|glossaryTermHyperlink style. The most likely case in which you would change the value for the mc-term-display property is if you want most of your glossary term links to open in a display other than popup; in that case, you would usually select the main MadCap|glossaryTerm style and change the mc-term-display to either the expanding or hyperlink value.

---

### **mc-thumbnail**

Determines the manner in which the user can switch from the thumbnail size to the image in its full size.

---

### **mc-thumbnail-max-height**

Sets the maximum height for thumbnail images.

---

### **mc-thumbnail-max-width**

Sets the maximum width for thumbnail images.

---

### **mc-toc-depth**

Changes the number of levels of topic links that are shown in the mini-TOC. The default setting is 3.

---

### **mc-topic-toolbar-items**

Determines what combination of Toolbar Item style classes are applied to the given style. You can add all of them, a combination of them, or choose to inherit them. These style classes are defined in the Styles tab of the Skin Editor.

These buttons are typically enabled or disabled using settings in the skin. However, you can manually enter them in this field.

---

### **mc-use-custom-sort-order**

Determines whether to enable custom sorting of topics (instead of alphabetical sorting) for topics listed in related topics links.

This setting is typically enabled or disabled in the Insert Related Topics Control dialog. However, you can manually edit it here if desired.



## APPENDIX C

---

# PDFs

The following PDFs are available for download from the online Help.

## I Tutorials

*Getting Started Tutorial*

*Autonumbers Tutorial*

*Back-to-Top Button Tutorial*

*Context-Sensitive Help Tutorial*

*Custom Toolbar Tutorial*

*eLearning Tutorial—Basic*

*eLearning Tutorial—Advanced*

*Image Tooltips Tutorial*

*Lists Tutorial*

*Meta Tags Tutorial*

*Micro Content Tutorial—Basic*

*Micro Content Tutorial—Advanced*

*Responsive Output Tutorial*

*Single-Sourcing Tutorial*

*Snippet Conditions Tutorial*

*Styles Tutorials*

*Tables Tutorial*

*Word Import Tutorial*

# | Cheat Sheets

*Context-Sensitive Help Cheat Sheet*

*Folders and Files Cheat Sheet*

*Learning & Development Cheat Sheet*

*Lists Cheat Sheet*

*Micro Content Cheat Sheet*

*Print-Based Output Cheat Sheet*

*Search Cheat Sheet*

*Shortcuts Cheat Sheet*

*Structure Bars Cheat Sheet*

*Styles Cheat Sheet*

# **I** User Guides

*Accessibility Guide*

*Analysis and Reports Guide*

*Architecture Guide*

*Autonumbers Guide*

*Branding Guide*

*Condition Tags Guide*

*Context-Sensitive Help Guide*

*Eclipse Help Guide*

*eLearning Guide*

*Getting Started Guide*

*Global Project Linking Guide*

*HTML5 Guide*

*Images Guide*

*Import Guide*

*Indexing Guide*

*Key Features Guide*

*Lists Guide*

*MadCap Central Integration  
Guide*

*Meta Tags Guide*

*Micro Content Guide*

*Navigation Links Guide*

*Plug-In API Guide*

*Print-Based Output Guide*

*Project Creation Guide*

*QR Codes Guide*

*Reviews & Contributions With  
Contributor Guide*

*Scripting Guide*

*Search Guide*

*SharePoint Guide*

*Skins Guide*

*Snippets Guide*

*Source Control Guide: Git*

*Source Control Guide:  
Perforce Helix Core*

*Source Control Guide:  
Subversion*

*Source Control Guide: Team  
Foundation Server*

*Styles Guide*

*Tables Guide*

*Tables of Contents Guide*

*Targets Guide*

*Template Pages Guide*

*Templates Guide*

*Topics Guide*

*Touring the Workspace Guide*

*Transition From FrameMaker  
Guide*

*Translation and Localization  
Guide*

*Variables Guide*

*Videos Guide*

*What's New Guide*

# INDEX

---

## A

- Absolute measurements 214
- Attributes 24
- Autocomplete
  - Internal Text Editor 120
  - stylesheets 120

## B

- Blockquote tag 187
- Borders
  - color 281

## C

- Cascading stylesheets 9, 11-12, 14, 17, 22, 86, 104, 106, 111, 151, 153, 157, 159, 161, 272, 279
  - embedded 14
  - external 15
  - inline 14
- Classes 33
  - generic 38
  - pseudo 53, 55, 60
- Color
  - border 281

## Comments

- styles 176

## Concepts

- styles 199

## Conditions

- styles 126

## Content

- editing and formatting 71

## CSS file 9, 11, 15, 157

## CSS variables 125, 218, 221, 225, 228, 232

## D

## Declaration blocks 25

## Declarations 25

## DITA

- styles 194

## DIV tag 187

## Divisions

- div 187

## F

## Fieldset tag 187

## Fixed measurements 214

## Form tag 187

## Formatting

- inline 71

- local 71
- topic content 71

## G

- Generic classes 38
- Groups (tags) 187
  - blockquote 187
  - div 187
  - fieldset 187
  - form 187

## H

- HTML
  - style class 194

## I

- Identifiers 48
- Importing
  - style sheets 161
  - styles 168
- Inheritance 64, 118
- Inline formatting *See Local formatting*
- Internal Text Editor
  - autocomplete 120
  - stylesheets 120

## K

- Keyword links
  - styles 199

## L

- Layouts *See Page layouts*
- Linking
  - concepts 199

- keyword links 199
- related topics links 199
- style sheets 162

- Local formatting 71

- Local stylesheets 153

## M

- MadCap Pulse
  - styles 196
- Map IDs *See Identifiers*
- Media queries 298-301, 307-313, 321, 323, 335
  - mobile 310
  - tablet 310
- Mediums 298-301, 307-313, 320, 323, 335
  - page layouts 307
  - table styles 307
  - targets 335

## N

- Next style 186

## P

- Page layouts
  - mediums 307
  - styles 307
- Pinning
  - styles 204
- Primary stylesheets 151
- Print-based output
  - table styles 275
  - topic styles 201
- Projects
  - primary stylesheets 151
- Properties
  - styles 350

- tables 71
- Proportional measurements 212
- Pseudo classes 53, 55, 60
- Pulse See *MadCap Pulse*

## Q

- Quotations
  - blockquote 187

## R

- Related topics links
  - styles 199
- Relative measurements 212

## S

- Search
  - styles 196
- See Also links See *Concepts*
- Selectors
  - advanced 28
  - classes 33
  - creating 22, 106
  - identifiers 48
  - renaming 170
  - spans 32
- Snippets
  - stylesheets 153
- Spans 32
- Style Inspector 125, 235-236, 248, 250, 253, 255, 259-260, 265, 267
- Styles 6, 9, 11-14, 17, 20, 70, 101, 103, 212, 319, 337, 350
  - advanced selectors 28
  - applying to content 136
  - attributes 24
  - blockquote 187

- classes 20, 33, 170
- comments 176
- concept links 199
- conditions 126
- creating 22, 106
- CSS variables 125, 218, 221, 225, 228, 232
- declaration blocks 25
- declarations 25
- deleting 211
- disabling 207
- DITA 194
- div 187
- editing 111, 120, 279
- fieldset 187
- form 187
- identifiers 48
- importing 161, 168
- inheritance 64
- inherited 118
- Internal Text Editor 120
- keyword links 199
- local stylesheets 153
- MadCap-specific 99
- media queries 298-301, 307-313, 321, 323, 335
- mediums 298-301, 307-313, 320, 323, 335
- next 186
- opening style sheets 159
- page layouts 307
- pinning 204
- primary stylesheets 151
- properties 20, 26, 99, 215, 217
- pseudo classes 53, 55, 60
- Pulse 196
- related topics links 199
- renaming 170
- renaming classes 170
- search 196
- selectors 22, 28, 33, 106
- spans 32

- stylesheets 6, 9, 11-14, 17, 22, 70-71, 86, 99, 101, 103-104, 106, 111, 151, 153, 157, 159, 161-162, 176, 186, 194, 212, 270, 272, 275, 279, 294, 319, 337, 350
- tables 279, 294
- tag groups 187
- tags 20
- topics 136, 194, 201
- types 74
- values 20, 27, 215, 217

Stylesheet Editor 74

Stylesheets

- creating 104
- editing 111
- local 153
- media queries 298-301, 307-313, 321, 323, 335
- mediums 298-301, 307-313, 320, 323, 335
- precedence 86
- primary 151
- project 151
- regular 86, 111, 157
- table 270
- target 151

Syntax coloring

- stylesheets 124

## T

Table styles 71, 270

- mediums 307

Tables

- editing 71
- properties 71
- styles 279, 294
- stylesheets 71, 270, 272, 275

Tags 337, 350

Targets

- mediums 335

- primary stylesheets 151

Template pages

- stylesheets 153

Templates

- factory 104, 272

Text Editor *See Internal Text Editor*

Topics

- styles 136, 194, 201
- stylesheets 151, 153, 194

## U

Unit of measurement 212

UOM *See Unit of measurement*

## V

Variables

- CSS 125, 218, 221, 225, 228, 232

## W

W3C 9, 11

World Wide Web Consortium *See W3C*