CSS Classes and IDs

The Need for Classes and IDs

Recall our prior example of using CSS to style three paragraph elements differently:

```
<p>Normal text</p>
<p style="font-weight:bold;">Text in Bold</p>
<p style="font-style:italic;">Text in Italics</p>
<p style="font-weight:bold;font-style:italic;">Text in Bold and Italics</p>
```

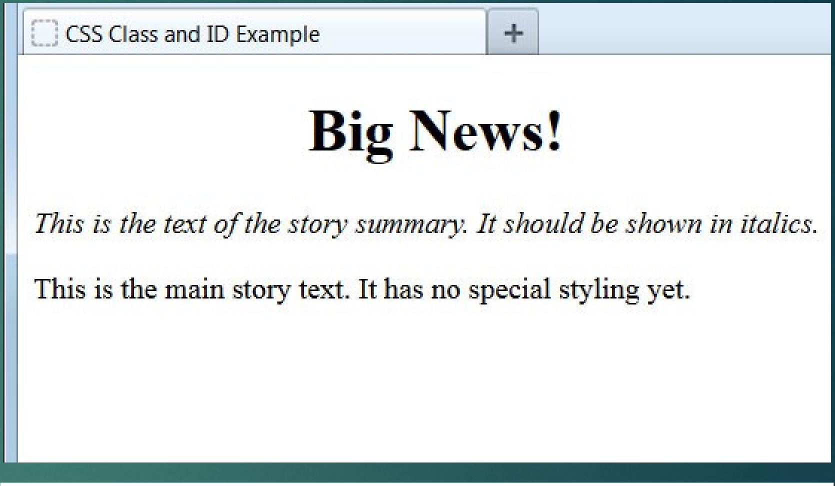
We had to use the inline style for these paragraphs since we had no other option with the tools we had learned to that point. If we had attempted to style the `<p>` element in an internal or external style sheet, the formatting rules would have applied to all the `<p>` elements, not specific ones.

With CSS classes and IDs, we have much more power in styling specific elements on the page. The following slides will demonstrate how this works.

Syntax of Classes and IDs

CSS style declarations for classes and IDs may be placed in internal and/or external style sheets. Here we're using an

```
<style type="text/css">
  #headline {
    text-align: center;
  }
  .summary {
    font-style: italic;
  }
</style>
...
<h1 id="headline">Big News!</h1>
<p class="summary">This is the text
   of the story summary. It should
   be shown in italics.</p>
<p>This is the main story text. It
   has no special styling yet.</p>
```



Big News!

This is the text of the story summary. It should be shown in italics.

This is the main story text. It has no special styling yet.

Styling declarations for IDs begin with the pound sign and for classes with a period. The **id** and **class** attribute, respectively, are added to page elements.

The style, class, and id attributes are known as **global attributes**, meaning they can be assigned to nearly any XHTML content element.

Classes and IDs Compared

At first glance, classes and IDs appear to function identically, but there are important differences between the two:

- IDs can be applied to one and only one element in a web document.
- Classes can be applied to an unlimited number of elements, even different types of elements on the same page.
- If conflicting styles from an ID and class are applied to the same element, the ID will outrank the class in the cascade.

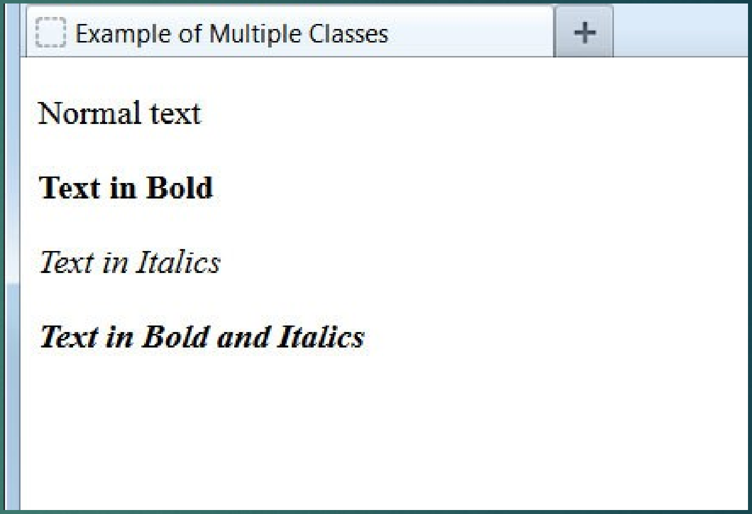
Newcomers to CSS often get confused over classes and IDs and the differences between them, so we'll look at several examples.

You might recall that we earlier used the **id attribute** to create a bookmark link. IDs (but not classes) are also used to manage page elements via JavaScript, a powerful scripting language that's beyond the scope of this introductory course.

Example: Multiple Classes, One Element Type

Let's look at our earlier example again, but this time let's create three CSS classes to style the three paragraphs differently.

```
<style type="text/css">
  .style1 {
    font-weight: bold;
  }
  .style2 {
    font-style: italic;
  }
  .style3 {
    font-weight: bold;
    font-style: italic;
  }
</style>
...
<p>Normal text</p>
<p class="style1">Text in Bold</p>
<p class="style2">Text in Italics</p>
<p class="style3">Text in Bold and
   Italics</p>
```



Normal text

Text in Bold

Text in Italics

Text in Bold and Italics

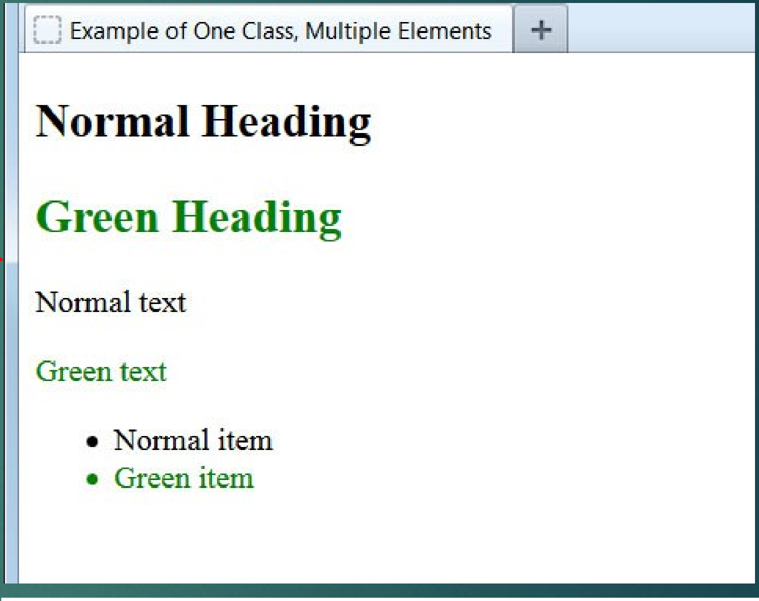
By using classes, we're no longer reliant on inline styles to format each instance of an element differently.

Example: One Class, Multiple Element Types

A single CSS class may also be applied to different element types on the same page:

```
<head>
     <style type="text/css">
       .style1 {
         color: green;
       }
     </style>
</head>

<body>
     <h2>Normal Heading</h2>
     <h2 class="style1">Green Heading</h2>
     <p>Normal text</p>
     <p class="style1">Green text</p>
     <ul>
       <li>Normal item</li>
       <li class="style1">Green item</li>
     </ul>
</body>
```



Normal Heading

Green Heading

Normal text

Green text

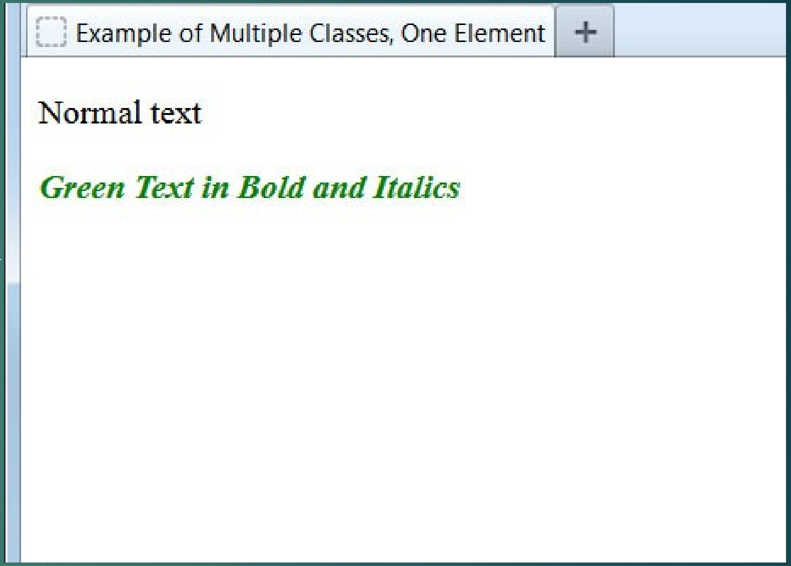
- Normal item
- Green item

Here we're beginning to see the power of CSS classes. By changing the class style declaration in the style sheet, all page elements assigned to that class will be modified.

Example: Multiple Classes, One Element

Multiple classes may also be applied to a specific element:

```
<style type="text/css">
  .style1 {
    font-weight: bold;
  }
  .style2 {
    font-style: italic;
  }
  .style3 {
    color: green;
  }
</style>
...
<p>Normal text</p>
<p class="style1 style2 style3">Green
   Text in Bold and Italics</p>
```



Normal text

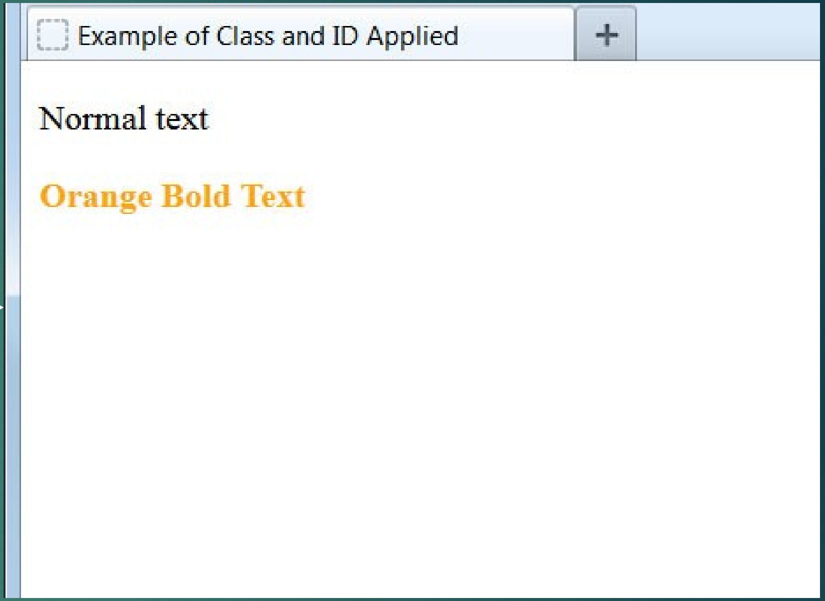
Green Text in Bold and Italics

We place a space between the different class names. There's no limit to the number of classes that can be applied to an element.

Example: Class and ID, Same Element

Both an ID and a class may be applied to a specific element:

```
<style type="text/css">
  #style1 {
    font-weight: bold;
  }
  .style2 {
    color: orange;
  }
</style>
...
<p>Normal text</p>
<p id="style1" class="style2">Orange
   Bold Text</p>
```



Normal text

Orange Bold Text

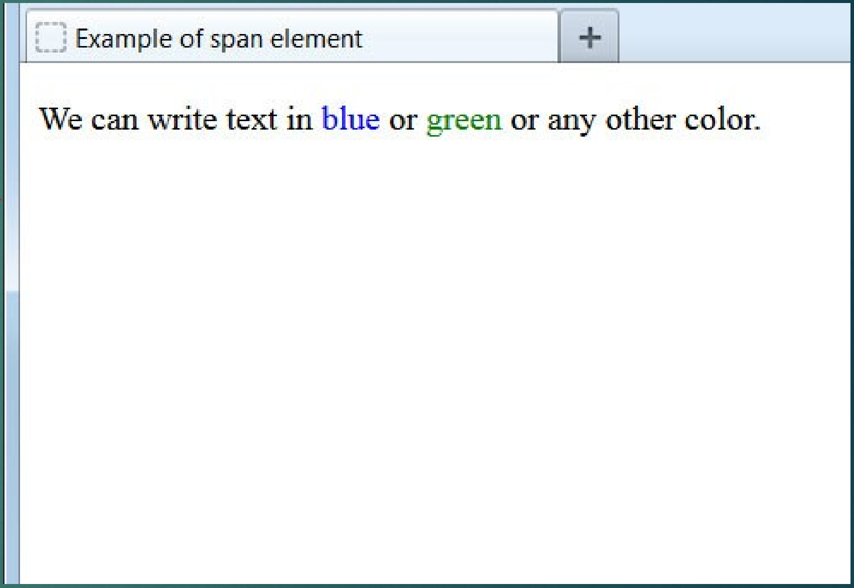
Remember that if there's a conflict between ID and class, the style from the ID will be applied, as ID takes priority in the CSS cascade.

An ID is restricted for use with a single element in a web document. Using the same ID more than once on a page will result in an error. We'll use IDs sparingly in this course, preferring classes for their greater flexibility.

The <span> Element

Sometimes we don't want a style to apply to an entire text element, but just a specific subset of text. For these instances, we use the handy <span> element:

```
<style type="text/css">
  .style1 {
    color: blue;
  }
  .style2 {
    color: green;
  }
</style>
...
<p>We can write text in <span
   class="style1">blue</span> or
   <span class="style2">green</span>
   or any other color.</p>
```



We can write text in blue or green or any other color.

Note that the <span> element does nothing on its own. Only once we associate it with a CSS class will the text inside the <span> be affected.

Naming Classes and IDs

We should maintain the goal of keeping our content and presentation as separate as possible. Using class and ID names that describe how the element appears departs from this objective. We should try to use class names that describe the *meaning* of the content, not its appearance:

Problematic names	Better names
.green	.slogan
.underline	.booktitle
.center	.caption
.bigletters	.headline

There's another hazard of using class names that describe the style. If we later change the class's appearance, we could create a confusing situation, for example a .green class that is styled to appear in blue!

Class and ID names must be at least two characters and can contain lowercase letters, numbers, dashes, and underscores. Always start the name with a letter, not a number.