

Introduction to CSS3 and HTML5: Lesson 5

Alan Simpson

Chapter 1

Introduction

Virtually all websites contain text (words), and we can present that text in many different ways. There are two main ways we handle text in websites and other electronic documents. We use HTML tags to define what each chunk of text *is* (for example, a heading, paragraph, list item, or whatever). In addition to the HTML, we use CSS to define how each chunk of text *looks* in terms of font, color, size, and other stylistic features.

In today's lesson, we're going to look at the main tags and CSS properties for organizing and styling text. First we'll focus on the tags and HTML. Then we'll look at the CSS side of things where you define fonts, colors, and other aspects of the text. We have a lot of ground to cover, so I'm going to hold off on the hands-on part for the duration of the lesson. Please use the lesson just to understand the options available to you first. The manual skills for applying these things are the same as you've been using all along—there won't be anything new there. Plus, you'll get plenty more hands-on in the assignment and future lessons.

Chapter 2

Tagging Text With HTML

Most Web pages contain some text (words). The text is usually organized into different types of elements, such as headings, paragraphs, lists, and so forth, just like you see in printed books and magazines. We use various tags in HTML to organize text into different types of elements, as I've summarized below. You probably learned about most of these in the *Creating Web Pages* course, so I won't cover all that ground again here. However, please notice that some element types are categorized as *Block* elements, while others are *Inline* elements. You'll see the difference in a moment.

Common Text Elements		
HTML5 Tag	Type	Description
<abbr>	Inline	Abbreviation
<address>	Block	Contact information (physical address)
	Inline	Boldface
	Inline	Deleted text (strikethrough)
	Inline	Italics and spoken with emphasis

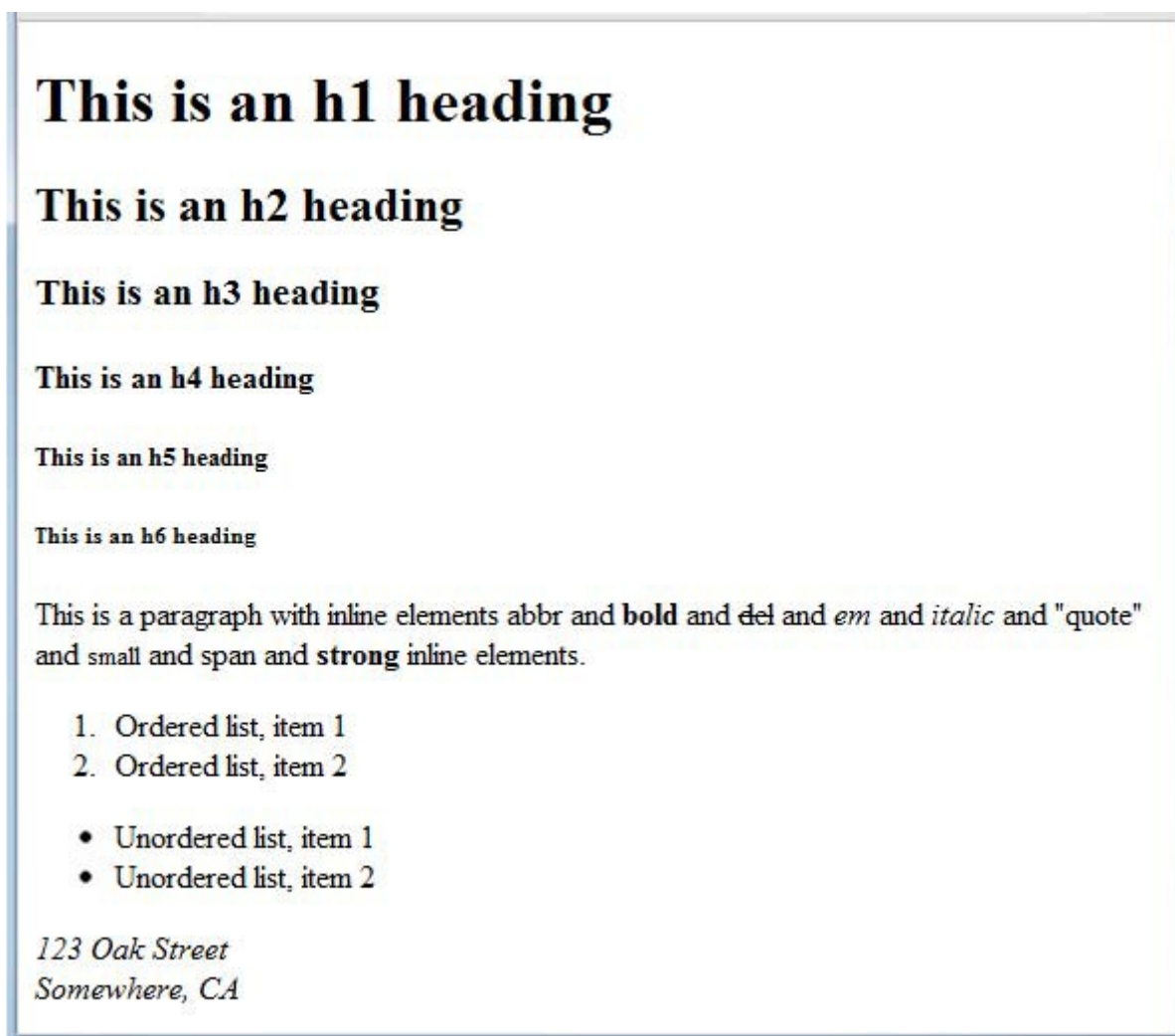
Common Text Elements		
HTML5 Tag	Type	Description
<h1> to <h6>	Block	Headings
<i>	Inline	Italics
	Block	List Item
	Block	Ordered (numbered) list
<p>	Block	Paragraph
<q>	Inline	Short quotation
	Inline	Generic tag for applying CSS styling
	Inline	Boldface and spoken strongly
	Block	Unordered list

Below is a page that contains examples of all those tags. It's a generic page—not something you'd likely use in real life. So you don't need to create it or copy it right now.

```
<!DOCTYPE html>
<html>
<head>
<title>Some Text Elements</title>
</head>
<body>
<h1>This is an h1 heading</h1>
<h2>This is an h2 heading</h2>
<h3>This is an h3 heading</h3>
<h4>This is an h4 heading</h4>
<h5>This is an h5 heading</h5>
<h6>This is an h6 heading</h6>
<p>This is a paragraph with inline elements <abbr>abbr</abbr> and <b>bold</b>
and <del>del</del> and <em>em</em> and <i>italic</i> and <q>quote</q> and
<small>small</small> and <span>span</span> and <strong>strong</strong> inline
elements.</p>
<ol>
<li>Ordered list, item 1</li>
<li>Ordered list, item 2</li>
</ol>
```

```
<ul>
<li>Unordered list, item 1</li>
<li>Unordered list, item 2</li>
</ul>
<address>123 Oak Street<br>Somewhere, CA</address>
</body>
</html>
```

Here's how that page would look in a Web browser. Since we haven't done any styling with CSS, you're seeing the *default styling* for each element type (the styling you get when you don't use CSS to define your own styles).



Common text elements in browser

The sample document above helps to illustrate the key difference between *block elements* and *inline elements*. A block element is one that always forces the text to start on a new line, like the headings, paragraph, lists and list items, and address in the example above.



Note

I didn't include `div` in the list of tags because it's not strictly a text-related element. You can put anything in a `div`, including text, tables, and images. But a `div` is a block element, and so its content always starts on a new line.

An inline element doesn't start on a new line. Instead, inline elements stay *in line* with other words on the same line. In other words, inline elements don't disrupt the natural left-to-right, top-to-bottom flow of text. The inline elements in the example above are `abbr`, `bold`, `del`, `em`, `italic`, `quote`, `small`, `span`, and `strong`. The `abbr` and `span` elements don't seem to have any default styling at all. I'll show you the purpose of the `abbr` element in a moment. A `span` is just a generic container for text to which you apply CSS styling. You'll see examples of that later in the course. But first, there's another important tag for text that we need to discuss.

Line Breaks

Most of you have probably noticed that line breaks in code don't translate to line breaks in the Web browser. For example, suppose you type a paragraph like this in your code:

```
<p>
Mary had a little lamb
Its fleece was white as snow
And everywhere that Mary went
The lamb was sure to go
</p>
```

The Web browser pays no attention to line breaks in the code. So when viewed through a Web browser, that paragraph will be formatted to fit within the width of the browser window or containing `div`, like this:

Mary had a little lamb Its fleece was white as snow And everywhere that Mary went The lamb was sure to go

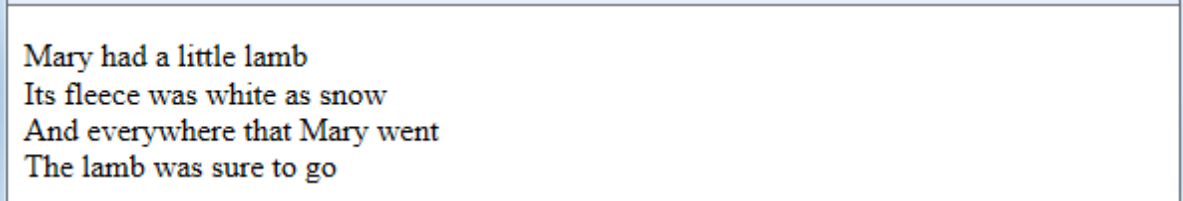
Standard text flow

If you want to end a line in the browser but not start a whole new paragraph (which would skip a line), you have to put a `
` (line break) tag in your code where you want the line to end. For example, in the code below, I put a `
` tag at the end of each of the first three lines.

```
<p>
Mary had a little lamb<br>
```

```
Its fleece was white as snow<br>
And everywhere that Mary went<br>
The lamb was sure to go
</p>
```

Here's how that code will render in the Web browser.



Mary had a little lamb
Its fleece was white as snow
And everywhere that Mary went
The lamb was sure to go

Line breaks in browser

Notice that each line ends right where the `
` tag is in the code. And the text is single-spaced, no blank lines at all. That's the purpose of the `
` tag. It lets you break a line and style using single-spacing within the text.

In books, online references, and other code, you may see the `br` tag written as `
` rather than `
`. The extra `/` at the end is the *XHTML syntax*. XHTML is the version of HTML that dominated Web development for many years starting at about the turn of the century, and it's still widely used today. HTML5 allows you to use either the `
` syntax from traditional, earlier HTML, or `
` from XHTML. So if you already have experience writing HTML code and you're in the habit of writing `
` or `
` already, you don't really have to worry about it. HTML5 will take it either way.

Abbreviations

When you're using abbreviations and acronyms, it's a good idea to put the text between `<abbr>...</abbr>` tags. The default styling is the same as the rest of the text. So in the browser window, the text won't look any different. However, it's still a good idea to use those tags because they allow a screen reader for the blind to understand that the letters inside the tags aren't a normal word, and so the program that's reading the text aloud won't try to pronounce the abbreviation as a regular word.

Another advantage to using `abbr` is that you can use the `title=` attribute right in the tag to define or expand the abbreviation. The `title=` attribute is a *global attribute* in HTML, meaning you can use it in virtually any tag that renders a visible element on the page. The text in the attribute is hidden, initially. But it appears in a screen tip near the mouse pointer if the user touches the mouse pointer on the word. Screen readers for the blind can also access the text in the `title=` attribute and read it aloud to a blind user.

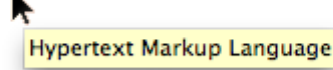
Below is an example where I've used the `abbr` tag to enclose and title the CSS and HTML abbreviations in a paragraph.

```
<p>Web developers use <abbr title="Cascading Style Sheets">CSS</abbr> and <abbr
title="Hypertext Markup Language">HTML</abbr> to create author Web pages.</p>
```

In the browser, the title text is hidden unless the user touches the mouse pointer to the abbreviation. When the tip of the mouse pointer is on the abbreviation, the title text appears in a screen tip near the

mouse pointer. The exact appearance of the screen tip depends on the browser (and you can't style the screen tip box, you can only define the text that appears in that box). Here's an example of pointing to the HTML abbreviation using Safari on a Mac.

CSS and HTML are wonderful tools for creativity in the digital age.



Abbreviation screen tip

Here's an example you can try out in your browser. Move your mouse over the words CSS and HTML.

CSS and HTML are wonderful tools for creativity in the digital age.

So those are some key tags for marking up text with HTML. Let's head over to Chapter 3 now and talk about the other side of the coin—using CSS to style text with fonts, colors, and more.

Chapter 3

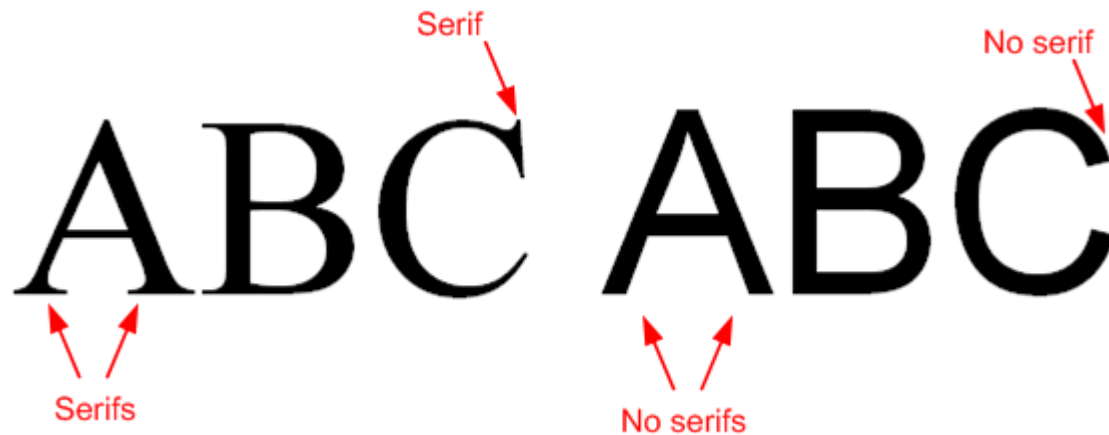
Styling Text With CSS

If you were to create a page using only the most basic HTML tags and no styling whatsoever, you'd get all of the default styling. That's basically a black serif font like Times or Times New Roman at 12 points for the body text. Headings and other elements use the same font, but at different sizes or with boldface or italic applied. Nothing fancy though. HTML just provides enough default styling to make the text readable. If you want something fancier than that, you need CSS. You also need an understanding of fonts, so let's start with that.

About Fonts

As most of you probably know, fonts are styles of text. There are hundreds of them, and there are a few ways to categorize them. For example, some fonts are *serif* fonts while others are *sans-serif* fonts. Serif fonts have tiny curlicues at some corners, like on the letters at left below. Sans-serif fonts have cleaner corners, as at right below.

Show Text Equivalent

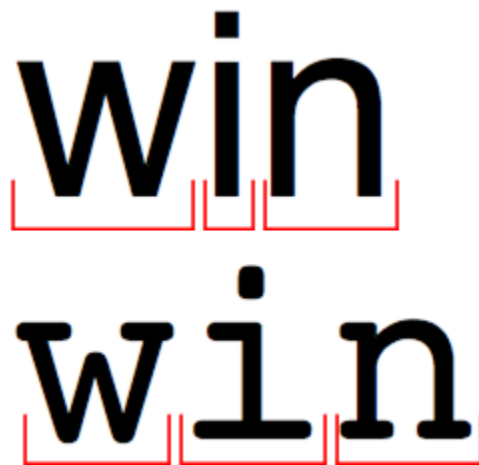


Serif font (left) and sans-serif font (right)

Newspapers and other printed materials often use serif fonts for the main body text (smaller letters) because the serifs make it easier to read across the line. Sans-serif fonts are generally recovered for headlines, titles, and other larger text in print material.

Another distinction in font types centers around the width of each letter. In *proportional fonts*, each letter is only as wide as it needs to be, as in the top example below. In a *monospace font* (also called a *fixed-width font*), each letter is about the same width, as in the bottom example below.

Show Text Equivalent



Proportional (top) and monospace (bottom) fonts

The serif and sans-serif fonts are the real workhorses of Web publishing. Tutorial websites about computer programming occasionally use monospace fonts to show computer code. But you won't see monospace used in headings and body text much because, frankly, it usually looks ugly there. Beyond those three basic font types are the *cursive fonts* (which emulate handwriting) and *fantasy fonts* (which are more elaborate decorative fonts that have to be used sparingly for brief titles and such). These can be tricky to use on the Web, for reason's we'll discuss a little later. Let's start with the basic fonts.

Choosing Fonts

CSS offers several font properties for defining fonts. The main thing that most people think of when they think of fonts is the font family, which is basically the name of the font. The CSS property you use to specify a font family is *font-family*, with the following syntax:

```
font-family:[familyname],... ,genericname
```

As in all syntax charts, the italics above are placeholders for values you provide. The square brackets indicate optional values you can omit. You never type the square brackets in your code, even if you provide a value. The square brackets are just in the syntax to distinguish between required values (no brackets) and optional values. The ... indicates that you can provide multiple values to the type to the left of the dots.

The *familyname* can be any valid font name on the user's computer. Unfortunately, not all computers are exactly the same. So you don't have any way of knowing exactly which fonts are available on any given user's computer. There are, however, certain fonts built into most versions of the Windows and Mac OS operating systems. Those are often referred to as *Web-safe fonts* because most people have them, so they work on most computers.

Since you can't be exactly sure which fonts a given person may have, the font-family property lets you devise a kind of wish list of fonts, where you specify the font you want to use as well as alternative backups. At the very end of the list, you specify one of the following generic font names.

serif

sans-serif

monospace

cursive

fantasy



Note

Font names aren't case-sensitive. People often type them as proper nouns where the first letter of each word is capitalized. However, it's really not necessary to do that because it doesn't matter if they're uppercase or lowercase.

You always want to put the generic name last so it's used as a last resort, when none of your preferred fonts is available. For example, consider the following:

```
font-family:Arial, Helvetica, Sans-Serif
```


What that bit of code is telling the browser is "Use the Arial font if it's available on this computer. If that's not available, then use Helvetica. And if Helvetica isn't available, then use whatever Sans-Serif font is available to this Web browser." Here are some common font-family descriptors with wish lists that cover all the most common Web-safe fonts.

Show Text Equivalent

Font List	Sample
font-family:Arial, Helvetica, Sans-Serif	Arial
font-family:Arial Black, Gadget, Sans-Serif	Arial Black
font-family:Brush Script MT, Cursive	<i>Brush Script MT</i>
font-family:Comic Sans MS, Cursive	Comic Sans MS
font-family:Courier New, Courier, Monospace	Courier New
font-family:Georgia, Serif	Georgia
font-family:Impact, Charcoal, Fantasy	Impact
font-family:Lucida Console, Monaco, Sans-Serif	Lucida Console
font-family:Lucida Sans Unicode, Lucida Grande, Sans-Serif	Lucida Sans Unicode
font-family:Palatino Linotype, Palatino, serif	Palatino Linotype
font-family:Tahoma, Geneva, sans-serif	Tahoma
font-family:Times New Roman, Times, serif	Times New Roman
font-family:Trebuchet MS, sans-serif	Trebuchet MS
font-family:Verdana, Geneva, sans-serif	Verdana

Web-safe fonts

Along with choosing a font family, which is basically the lettering style, you'll often want to specify a size.

Specifying a Font Size

Those of you with a publishing or similar background may be familiar with font sizes being expressed in *points*, where 1 point is equal to about 1/72 of an inch. In CSS, you can use *pt* to specify font sizes in points. You can also use pixels (px), percent (%), which is relative to the default font size, or em, which is also relative to the default font size. The ratio is 16px = 12pt = 1em = 100%. The default font size is whatever font size is specified for the element that contains the text. Unless you specify otherwise, the default font size is 12 points in most browsers.

Here are some sample font sizes using different units of measure. Note, however, that there's no advantage or disadvantage to using one sizing method over another. So feel free to choose whichever seems easiest to you.

Sample font sizes						
Points	8pt	10pt	12pt	18pt	24pt	36pt
Pixels	11px	13px	16px	24px	32px	48px
Ems	0.7em	0.8em	1em	1.5em	2em	3em
Percentages	70%	80%	100%	150%	200%	300%

You can use the CSS `font-size` property to specify a font size. The syntax is:

```
font-size: value
```

Where *value* is a number followed by a unit of measure (for example, *pt* for points, *px* for pixels or *%* or *em* for a relative unit of measure). Never put a space between the number and the unit of measure. For example, all of the following examples illustrate valid CSS code for setting the font size.

```
font-size: 12pt
font-size: 16px
font-size: 1em
font-size: 100%
```

CSS for Boldface, Italics, and Small Caps

You can set boldface, italics, and small caps with CSS. As you know, you can set boldface and italics with tags. In HTML the `` and `<i>` tags provide boldface and italics. The `...` tags also provide boldface, but has the added bonus of making a screen reader for the blind that's reading the text aloud add some strength to the spoken word. Similarly, the `...` tags in HTML display text in italics and make a reading voice add emphasis to the word. The ``, `<i>`, ``, and `` tags are often used to mark up a word or two of text with a flow of otherwise normally formatted text.

To use bold or italic strictly as a visual style for headings, titles, or other specialized text, you can use CSS. The CSS property for boldface is *font-weight* with the syntax:

```
font-weight: value
```

You must replace *value* with *bold* to apply bold, or *normal* to remove bold (such as the default boldface that's automatically applied to headings). Here are two examples.

Sample font weights	
font-weight:normal	Sample text
font-weight:bold	Sample text

The syntax for italics is similar to that for boldface.

```
font-style: value
```

where *value* can be either the word *italic* to italicize text or *normal* to remove italics from text that's already been italicized through some other styling.

Sample font styles	
font-style:normal	Sample text
font-style:italic	<i>Sample text</i>

Small caps is a style where lowercase letters are shown in uppercase, but at a smaller size than the uppercase letters. The style is sometimes used for fancier titles, but should never be applied to regular body text, because it makes that smaller text difficult to read. CSS offers the *font-variant* as a means of applying (or removing) the small caps style. The syntax is:

```
font-variant: value
```

Replace *value* with *small-caps* (no spaces) for small caps or *normal* for normal text. Here are examples.

Font variants	
font-variant:normal	Sample text
font-variant:small-caps	SAMPLE TEXT

Setting the Line Height

By default, text in paragraphs and other elements is single-spaced, which provides a comfortable amount of space between each line in a paragraph. But that's just the default styling. You can use the CSS `line-height` property to change the line height. As with the font size, you can specify the line height using pixels, points, em, or percent. You can also use a number with no unit of measure or the word *normal*. If you use a number without a unit of measure, like this:

```
line-height:2
```

that means you want double-spacing. If you just want the normal line spacing, you can use the word *normal* as a value, like this:

```
line-height:normal
```

Most people use either the whole number method or a percent to specify the line height. For example, you can use 1.5 or 150% for space-and-a-half line spacing. Use 2 or 200% for double spacing. Here are some examples of line spacing using percentages.

Sample line heights	
This text has the line height set to 80% This text has the line height set to 80% This text has the line height set to 80%	
This text has the line height set to 100% This text has the line height set to 100% This text has the line height set to 100%	
This text has the line height set to 120% This text has the line height set to 120% This text has the line height set to 120%	This text has the line height set to normal. This text has the line height set to normal. This text has the line height set to normal.
This text has the line height set to 150% This text has the line height set to 150% This text has the line height set to 150%	
This text has the line height set to 200% This text has the line height set to 200% This text has the line height set to 200%	

You can set any of the font styles we've discussed on any element that contains text. It's common for people to set the font styles in the `body{}` style rule because there they become the *default font* for the page. In other words, all the headings, paragraphs, and lists will be in the font family you specify. The sizes of headings will be relative to the size of the font you specify.

However, the "default font" doesn't mean "the font you're stuck with." You're never stuck with anything when it comes to styling your Web pages! The default font you specify for the body element is just the

font you get when you don't specify something else for some other element. Like all CSS properties, you can use the font properties in any style rule or inline style. If you want to set a default font for all the pages in your site, use the `body{}` style rule that you already have. For example, if you want to use a sans-serif font (rather than a serif font) as your base font, and you want to size it at 10 points (rather than the default 12 points), and you want a little extra space between lines in paragraphs, you could add these three descriptors to your `body{}` style rule.

```
body{
  font-family:Arial, Helvetica, Sans-Serif;
  font-size:10pt;
  line-height:1.5;
  ...
}
```

In the example above, the ... just means "and any other descriptors you want here," such as background styling. You never type ... in your code.

Using the font: Shorthand Property

As an alternative to spelling out every aspect of font styling on a separate line with a separate property, CSS offers a shorthand property called *font*: that lets you pack a lot of information about font styling onto a single line using this syntax:

```
font: [style] [variant] [weight] size[/height] family
```

As always, italics are placeholders for values you provide, and square brackets indicate optional values. The values you can provide match those of the non-shortcut properties you just learned about:

- *style*: This can be italic or normal. If you leave it out, it defaults to normal.
- *variant*: This can be small-caps or normal. If you leave it out, it defaults to normal.
- *weight*: This can be bold or normal. If you leave it out, it defaults to normal.
- *size*: This should be a font size expressed as a number followed by a unit of measure like *pt* (points), *px* (pixels), *%* (percent), or *em* (relative size).
- */line-height*: This is a number followed by a unit of measure as above. If omitted, it defaults to the normal spacing for the font (usually about 120%). Only type the slash if you provide a line height. If you don't specify a line height, don't type the slash.
- *family*: This is a wish list of font names and a generic font name. Same as you'd type using *font-family*:

Earlier you saw how to specify a font size, line height, and family using three lines of code. Here's the shorthand way to accomplish the same goal and exactly the same styling, using only one line of code:

```
body{
```

```
font: 10pt/1.5 Arial, Helvetica, Sans-Serif
}
```

The above works because if you omit the first three values (*style variant weight*), they default to normal. So the line above is the same as this:

```
body{
    font: normal normal normal 10pt/1.5 Arial, Helvetica, Sans-Serif
}
```

Of course, you don't have to make them all normal. But if you don't want normal, you have to specify the value you want. For example, if you wanted all of your h1 headings to be 24 points, italic, small caps, and bold with an Arial Black font, you could add a style rule like this one to your style sheet:

```
h1{
    font: italic small-caps bold 24pt Arial Black, Gadget, Sans-Serif
}
```

Because that style rule has h1 as its selector, that font styling will be applied to h1 elements only. Any other elements will still obey the more general font styling defined in the body{} style rule. You'll have a chance to try that out in the assignment for this lesson. For now, let's forge ahead with learning more ways you can style your text.

What About Color?

You may be wondering if there's a font-color property to define the color of text. That's a *no* for the simple reason that there's no need for a font-color property. The color: property in CSS defines the foreground color, which naturally applies to all text. So you wouldn't specify a color anywhere in the font shorthand. You would have to do that on a separate line using the color: property, like this (if you wanted the text to be blue):

```
font: italic small-caps bold 24pt Arial Black, Gadget, Sans-Serif;
color:blue
```

Let's mosey over to Chapter 4 now and talk about still more ways you can style your text.

Chapter 4

More Text Styling

We've discussed many ways you can style text in the previous two chapters. But in addition to all of those, there are some CSS text properties you may find useful in your work. So we'll take a quick look at those in this chapter. Then we'll look at ways you apply all of this to your sites.

Underlines, Overlines, and Strikethrough

If you need to control underlines, overlines, or strikethrough with CSS, the `text-decoration` property is your ally. The syntax is:

```
text-decoration: value
```

Replace *value* with the word *underline*, *underline*, *line-through*, or *none* to get the effect you want, as in the examples below.

Text decoration examples	
text-decoration:underline	Sample text
text-decoration:underline	Sample text
text-decoration:line-through	Sample text
text-decoration:none	Sample text

Using Text Shadows

In an earlier lesson, you learned how to give your divs a raised appearance using the `box-shadow` property to put a drop shadow behind a div. CSS3 also offers a `text-shadow` property that lets you add a drop shadow to text. Before I give you the syntax, let me just issue a couple of warnings:

- Since `text-shadow` is new in CSS3, it only works in the newer browser versions. People using older browsers will still see the text, but they won't see the shadow.
- You can only apply text shadows to short headings and such. Don't apply shadows to all the text on a page, especially regular body text in paragraphs and lists. It makes that smaller text too hard to read.

The syntax for using `text-shadow` is similar to that of `box-shadow`, only there's no *spread* value. The syntax is

```
text-shadow: horizontalOffset verticalOffset [blur] [color],...
```

As always, the italics are placeholders for values you must provide, square brackets indicate optional values that you can omit, and... is there because you can repeat the values with different settings to apply multiple shadows to the text. You'll see examples shortly. First, let's discuss the values:

Replace *horizontalOffset* with a number and unit of measure (like `px`) indicating the horizontal location of the shadow. A negative number puts it on the left side of the text, while a positive number puts it on the right side.

Replace *verticalOffset* with a number and unit of measure (like `px`) indicating the vertical location of the shadow. A negative number puts it on the top, and a positive number puts it on the bottom.

The *blur* value is optional and can be omitted for a value of zero. No blur means a sharp edge to the shadow. You can define a blur as a number followed by a unit of measure. The larger the blur value, the blurrier the edge.

Replace *color* with any valid color name or hex code. If omitted, the shadow will be the same color as the text.

Here are some examples. In the last two examples, I used the CSS color property to give the text a gray or silver color first; then I used two shadows to get an embossed or sunken effect.

<code>text-shadow:-2px -2px silver</code>	Sample Text
<code>text-shadow:2px 2px silver</code>	Sample Text
<code>text-shadow:-4px -4px 2px gray</code>	Sample Text
<code>text-shadow:4px 4px 2px silver</code>	Sample Text
<code>text-shadow:0 0 10px lime</code>	Sample Text
<code>text-shadow:0 -4px 9px red</code>	Sample Text
<code>color:#ccc; text-shadow: -1px -1px white, 1px 1px #333</code>	Sample Text
<code>color:#ddd; text-shadow: 1px 1px white, -1px -1px #444</code>	Sample Text

Text shadow examples

Tip: To design a text shadow interactively, try the page at https://alansimpson.me/html_css/tools/textshadowmaker.html

Text alignment has to do with how text aligns inside its containing element. The containing element might be the page body. Or it could be a block element like a heading, paragraph, or div. To control text alignment, you use the CSS `text-align` property in the opening tag of the block element that contains the text. In other words, you don't apply `text-align` to inline elements like `abbr`, `b`, `strong`, or `span` because those elements flow naturally with the rest of the text and applying an alignment would disrupt that natural flow. You always apply `text-align` to block elements like `body`, `div`, `h1` through `h6`, or paragraphs, to align the text within the element.

The syntax for `text-align` is:

```
text-align: value
```

Replace *value* with *left*, *center*, *right*, or *justify*. You can apply the left, center, and right value to any amount of text, even single lines, as you see below.

Text alignment with single lines of text	
<code>text-align:left</code>	Sample Text
<code>text-align:center</code>	Sample Text
<code>text-align:right</code>	Sample Text

The `justify` format only applies to chunks of text that are long enough to wrap to multiple lines within their containing element. In those elements, the `justify` value inserts spaces within each line to give each side of the text a smooth edge. The left, center, and right values, which can be applied to that longer text, creates a ragged edge on one or both sides. In practice, you should use left or justify for paragraphs. Use center or right for shorter single-line chunks of text only.

Text alignment with multiple lines of text	
<code>text-align:left</code>	Sample text that's long enough to wrap within this cell, so you can see how different kinds of alignments look with the longer multi-line blocks of text.
<code>text-align:justify</code>	Sample text that's long enough to wrap within this cell, so you can see how different kinds of alignments look with the longer multi-line blocks of text.
<code>text-align:center</code>	Sample text that's long enough to wrap within this cell, so you can see how different kinds of alignments look with the longer multi-line blocks of text.
<code>text-align:right</code>	Sample text that's long enough to wrap within this cell, so you can see how different kinds of alignments look with the longer multi-line blocks of text.

First Line Indent

If your pages will contain full paragraphs of text, you may want to indent the first line of each paragraph. You can use the text-indent property with this syntax for that:

```
text-indent: value
```

Replace *value* with a number and unit of measure. The most common unit of measure is *px*, and the most common width is around 48px. Since this kind of indenting is used almost exclusively with paragraphs, it's most commonly used in style rules for paragraphs, which are defined by `<p>...</p>` tags. So the style rule to indent the first line of each paragraph would look like this in a style sheet:

```
p{  
  text-indent: 48px  
}
```

Let's head over to Chapter 5 now and wrap things up.

Chapter 5

Conclusion

Whew, we covered a lot of tags and a lot of CSS for styling text in this lesson! But it's important to be aware of your options, and that's what much of this lesson has been about. Making sure you're aware of the fact that when it comes to styling text, you have many options. You'll see the many different ways you can apply the tags and styling in upcoming lessons. And you'll get some hands-on practice in the assignment for this lesson. For now, to summarize the key points:

- You use HTML tags in your page to define what each element of text *is* (heading, paragraph, abbreviation, list, and so forth).
- Block elements are elements that always start on a new line. Inline elements stay in the same line of text.
- Line breaks in code don't translate to line breaks in the Web browser. Use the `
` tag to insert line breaks.
- Use CSS to style your text in terms of fonts, size, alignment, drop shadows, and more.

In the next lesson, we'll focus on advanced skills for creating CSS style rules. These will help pave the way for getting into ever-fancier designs and pages that are easy to maintain and change. See you there!

Supplementary Material

Please see this page for [updated Supplementary Material links](#).

FAQs

Q: I think I've lost track of what my code should look like by now. Can I see that somewhere?

A: Sure. And in fact, feel free to copy and paste the page code right into your HTML5 Template.htm page, and the style sheet into your style sheet.css file, if you want to get right up to where you should be by the time you finish Lesson 5 and Assignment 5. Here is the HTML5 Template.htm page code.

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <link href="stylesheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <div id="wrapper">
    <h1>Sample h1 Heading</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis
aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non proident, in culpa qui officia
deserunt mollit anim id est laborum</p>
    <h2>Sample h2 Heading</h2>
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
<!-- End wrapper -->
```

```
</body>
</html>
```

And here's how we left the style sheet at the end of Assignment 5.

```
/* stylesheet.css */
/* Style rule for the body element */
body{
    background-color:yellow;
    background-image:url(pix/bkgleather2.jpg);
    /* Default font styling */
    font: 10pt Arial, Helvetica, Sans-Serif;
    color:#5d3f25;
}
/* Style rule for the wrapper div */
#wrapper{
    width:900px;
    margin:0 auto;
    background-color:white;
    border:solid 2px #4c2e16;
    border-radius:20px;
    box-shadow:8px 8px 8px black;
}
/* Style for h1 headings */
h1{
    font:italic small-caps bold 24px Arial Black, Gadget, Sans-Serif;
    text-shadow:-2px -2px 2px #a48362;
    text-align:center;
}
/* Style for h2 headings */
h2{
    font:italic small-caps bold 18px Arial Black, Gadget, Sans-Serif;
    text-shadow:-1px -1px 1px #a48362;
}
/* Style for paragraphs */
p{
    line-height:1.5;
}
```

Q: Having the text so close to the edges of the wrapper div is really bothering me. Can't you just tell me the code to add some space there?

A: Sure, I can do that. Try adding `padding:16px;` to the `#wrapper` style rule in your style sheet like this:

```
/* Style rule for the wrapper div */
#wrapper{
    width:900px;
    margin:0 auto;
    background-color:white;
    border:solid 2px #4c2e16;
    border-radius:20px;
    box-shadow:8px 8px 8px black;
    padding:16px;
}
```

The 16px is just an example. You can make your padding any size you want. But do keep in mind that if you're really going to be good at this, you need to understand *why* that works, and in what other contexts you might use it. We'll tackle the bigger picture explanation in a later lesson.

Q: It looks like both the `` and the `` tags make boldface. Both `` and `<i>` make italics. What gives?

A: The `...` tags show text as boldface, like `...`. But they do something else that the `...` tags don't. The `...` tags make the voice that's reading the text speak the boldface words with some extra strength. Similarly, the `...` tags show the text as italics, but they make a screen reader for the blind say the words with added emphasis. The added strength and emphasis can make it a little easier for a blind person to understand the organization of the page. For that reason, the W3C is recommending that everyone use `...` rather than `...` for boldface, and `...` rather than `<i>...</i>` for italics. The `` and `<i>` tags are mainly staying in the language for backward compatibility with earlier versions of HTML.

Assignment

You can start on the text styling for a site at any time—even before you've decided what text you're going to put into it. All you have to do is put in some random *filler text*, and mark it up with the tags you're mostly likely to use. For longer passages of text, you can use lorem ipsum text commonly used in publishing and graphic design. The text has no meaning. But the word lengths, spaces, and punctuation are similar to real text. So it serves as a good placeholder for designing fonts when you haven't yet decided what words you're going to put into a page.

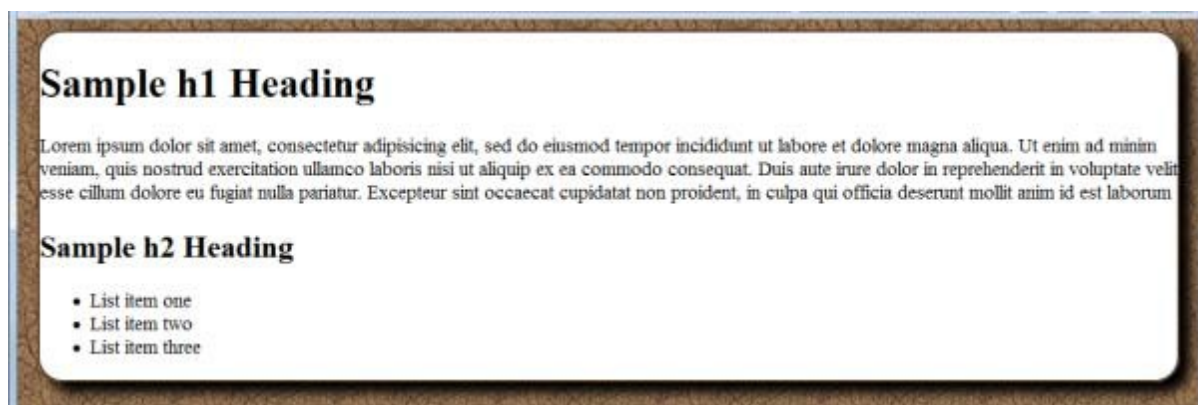
You may not even know offhand exactly what kinds of elements you want to use. But that's no big deal either. Just start off with the most common ones like h1, h2, paragraphs, and lists. That will get you enough to start designing. To give it a whirl, follow these steps to add some practice text to your HTML5 Template.htm page now:

1. Open your *Intro HTML5* folder.
2. Open *HTML5 Template.htm* for editing (right-click or CTRL + click its icon, and choose **Open With** and then the name of your editor).
3. Delete the temporary text that reads *Page content goes here*.
4. Type or copy and paste in the tags and filler text below so they're all inside the wrapper (between the `<div id="wrapper">` and `</div><!-- End wrapper -->`).

```
5. <h1>Sample h1 Heading</h1>
6.
7. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
  quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
  consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
  cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
  proident, in culpa qui officia deserunt mollit anim id est laborum</p>
8.
9. <h2>Sample h2 Heading</h2>
10. <ul>
11.   <li>List item one</li>
12.   <li>List item two</li>
13.   <li>List item three</li>
    </ul>
```

14. Close and save the page.

To see how the text looks without any styling, go ahead and double-click the icon for *HTML5 Template.htm* to open it in a Web browser now. The text will be black and in a Times font, like this (though your background and other styles may be different if you're working on your own design right now).



Filler text in the page

Next we'll do a little styling. In this course, we'll be doing most of our styling in the external style sheet, `stylesheet.css`, so that you can apply the styling to multiple pages in the site. As we've discussed, that makes it easier to maintain a consistent look-and-feel across all the pages in the site. It also makes it easier to make stylistic changes to multiple pages in the site in the future, because you need only make the change in the one `stylesheet.css` file, not each individual page in the site. The filename of our external style sheet is `stylesheet.css`, so let's open that in an editor now and add some styling:

In your Intro HTML5 folder, right-click or CTRL + Click **stylesheet.css** and choose **Open With** and then the name of your editor.

Put the cursor just before the `}` for the body style rule, and type (or copy and paste in) the following comment and font styling.

```
/* Default font styling */
font: 10pt Arial, Helvetica, Sans-Serif;
color: #5d3f25;
```

If you did it correctly, you'll have added those three lines to whatever you already had in your `body{}` style rule, so that whole style rule now looks something like this:

```
/* Style rule for the body element */
body{
    background-color:yellow;
    background-image:url(pix/bkgleather2.jpg);
    /* Default font styling */
    font: 10pt Arial, Helvetica, Sans-Serif;
    color:#5d3f25;
}
```

The font and color you set in the body style rule define a kind of default font that applies to everything. But we're not done yet. Let's go ahead and add some fancier styling for `h1` and `h2` heads. And we'll increase the line spacing for paragraphs too. Continue with the steps below:

1. Move the cursor so it's below the `#wrapper` style rule.
2. Type or copy and paste in the following style rules.

```
3. /* Style for h1 headings */
4. h1{
5.     font:italic small-caps bold 24px Arial Black, Gadget, Sans-Serif;
6.     text-shadow:-2px -2px 2px #a48362;
7.     text-align:center;
8. }
9. /* Style for h2 headings */
```

```

10. h2{
11.     font:italic small-caps bold 18px Arial Black, Gadget, Sans-Serif;
12.     text-shadow:-1px -1px 1px #a48362;
13. }
14. /* Style for paragraphs */
15. p{
16.     line-height:1.5;
    }

```

17. Close and save the style sheet.

Now go ahead and open the page (HTML5 Template.htm) again by double-clicking its icon. Assuming you use a browser that supports text-shadow, you should see the new font styling applied. (If you don't see it at first, click the Reload or Refresh icon in your browser.) If you see everything but the text shadow, you may be using an older browser that doesn't support that CSS3 property. For example, Internet Explorer 9 does not support text-shadow. But assuming you followed the steps correctly and you're using a modern Web browser, your filler text should now look something like this.



Filler text with some styling

Of course, you're free to style your text however you want. The example I've provided is just a working example.

You'll notice the text is very close to the left side of the wrapper div. Too close, in fact. That's because we haven't put any padding in that div yet—or even discussed padding! But we will soon. So for now, just hold that thought. Better to spend your time right now staying focused on your text, since that's the topic of this lesson.

When you get everything done, congratulations! If you got a little lost somewhere along the line, and you aren't quite sure what code you should have in your HTML 5 Template.htm and stylesheet.css files right now, take a look at the Lesson 5 FAQs. I put it all there so you can see it all in one place and copy it into your own files, if need be, to get your own code in sync with the lesson code.