3.17 Font Properties

In this section, we describe how to display text with different font characteristics. You've probably heard of the term "font," but if not, *font* refers to the characteristics of text characters—height, width, thickness, slantedness,[7] body curvatures, and endpoint decorations. That should make more sense later on when we present CSS font property details and show examples. Specifically, you'll learn about the `font-style`, `font-variant`, `font-weight`, `font-size`, `font-family`, and `font` shorthand properties.

`font-style` Property

The `font-style` property specifies whether the text is to be displayed normally or slanted. Here are the valid values for the `font-style` property:

`font-style` Values	Description
`normal`	Upright characters (not slanted).
`oblique`	Use the same font as the current font, but slant the characters.
`italic`	Use a cursive font (which tends to be slanted and is supposed to look like handwriting).

[7] Apparently "slantedness" is not a word, but it should be. If it becomes a word, you heard it here first.

These descriptions indicate a slight difference between the `oblique` and `italic` properties, with `italic` tending to be more decorative. Most web developers use the value `italic`. Because italics are so common, you should memorize the following technique for generating italics:

```
.italics {font-style: italic;}
```

As always, choose a name for the class selector that's descriptive. Here, we chose the name `italics` because it's descriptive and easy.

Upright (normal) characters are the default, so why would you ever want to specify normal for the `font-style` property? Suppose you have a whole paragraph that's italicized and you want one word in the paragraph not italicized. To make that word normal (not italicized), you can use `font-style: normal`.

The W3C provides default values for all CSS properties, and to force the default value to be used for a particular property, you can specify `initial` for that property. So, given the situation described with an italicized paragraph, to make one word not italicized, you can apply the following rule to that word:

```
.not-italicized {font-style: initial;}
```

`font-variant` Property

The `font-variant` property specifies how lowercase letters are displayed. Here are the valid values for the `font-variant` property:

`font-variant` Values	Description
`normal`	Display lowercase letters normally.
`small-caps`	Display lowercase letters with smaller-font uppercase letters.

Here's an example that uses a `small-caps` CSS rule:

```
.title {font-variant: small-caps;}
...
<div class="title">The Great Gatsby</div>
```

And here's the resulting displayed text:

THE GREAT GATSBY

`font-weight` Property

The `font-weight` property specifies the boldness of the text characters. Here are the valid values for the `font-weight` property:

`font-weight` Values	Description
`normal, bold`	It's up to the browser to determine a font weight that can be described as normal or bold.
`bolder, lighter`	Using a value of `bolder` causes its targeted text to have thicker characters than the text that surrounds it. Using a value of `lighter` causes its targeted text to have thinner characters than the text that surrounds it.
`100, 200, 300, 400, 500, 600, 700, 800, 900`	`100` generates the thinnest characters and `900` the thickest characters. `400` is the same as `normal`, and `700` is the same as `bold`.

These descriptions for `bolder` and `lighter` are probably all you need to know, but you may want to dig a little deeper. With `bolder` and `lighter`, the targeted text inherits a default `font-weight` value from its surrounding text, and then the targeted text's weight gets adjusted up or down relative to that inherited weight. For example, if you specify a `bold` font weight for a paragraph, you can make a particular word within the paragraph even bolder by specifying `bolder` for that word's font weight.

Because boldfacing is such a common need, you should memorize the following technique for making something bold:

```
.bold {font-weight: bold;}
```

`font-size` Property

The `font-size` property specifies the size of the text characters. There are quite a few values allowed for the `font-size` property. Here are the most appropriate ones:
Here's an example class selector rule that uses the `font-size` property with an `xx-large` value:

`font-size` Values	Description
`xx-small, x-small, small, medium, large, x-large, xx-large`	It's up to the browser to determine a font size that can be reasonably described as `xx-small, x-small, small`, etc.
`smaller, larger`	Using a value of `smaller` causes its targeted text to have smaller characters than the text that surrounds it. Using a value of `larger` causes its targeted text to have larger characters than the text that surrounds it.
`number of em units`	One em unit is the height of the element's normal font size.

```
.huge-font {font-size: xx-large;}
```

So far, most of the CSS property values you've seen have consisted of a text description, such as xx-large. As an alternative, some properties have values that are comprised of two parts—a number and a unit. For example, for the font-size property, you can use a value with a number next to em, where one em unit is the height of a typical character. The em unit's name comes from the letter M. Originally, one em unit equaled the height of the letter M. However, there are fonts for languages that don't use the English alphabet, so it was deemed inappropriate for em to rely on the letter M. Thus, em is no longer tied to the letter M, and testing shows that a single em unit is a bit taller than the height of M.

Here are class selector rules that use the font-size property with em values:

```
.disclaimer {font-size: .5em;}
.advertisement {font-size: 3em;}
```

The first rule is for disclaimer text, which is supposed to be annoyingly small to avoid scrutiny, and its .5em value displays text that is half the size of normal text. The second rule is for advertisement text, which is supposed to be annoyingly large to draw attention, and its 3em value displays text that is three times the size of normal text.

Here, for each font-size value, note that there is no blank space separating the number from em. Likewise, for all CSS values that consist of a number and a unit, you should not have a blank space. Having no blank space is a requirement of the CSS standard.

Absolute Units

There are quite a few other techniques for specifying font size. For example, you can specify a number along with an in, cm, mm, pt (point), or pc (pica) unit. The W3C refers to those units rather disdainfully as "so-called absolute units." Although they're still used every now and then, and you should understand them, absolute units have fallen out of favor for everyday needs. This is because it's good to allow users to adjust the size of things, particularly people with impaired eyesight. Also, with regular monitors, testing shows that the "absolute units" are not absolute; their sizes vary with different resolutions, different monitors, and zooming in and out.

The W3C says absolute units are required to have absolute lengths only when the target device has a high resolution; that usually means just printers, but it can also mean high-resolution monitors. For a scenario where a fixed size is essential and absolute units are appropriate, picture your wedding preparations. Suppose you're tasked with printing your wedding invitations on pre-cut pink cherub-adorned cards. To avoid agitating a stressed-out fiancé(e), you'll want to use absolute units to make sure the words fits perfectly on the cards.

font-family Property

The prior font properties allow the web developer to choose values for specific font characteristics (font-weight for characters' thickness, font-size for characters' height, etc.). The next font property, `font-family`, is more holistic in nature. The `font-family` property allows the web developer to choose the set of characters that the browser uses when displaying the element's text. As you'd expect, the characters in a particular character set are similar in appearance—same basic height, width, body curvature, and so on.

Here's an example class selector rule that uses the `font-family` property:

```
.ascii-art {font-family: Courier, Prestige, monospace;}
```

Note that with the `font-family` property, you should normally have a comma-separated list of fonts, not just one font. In applying the preceding rule to elements that use "asci-art" for their `class` attribute, the browser works its way through the `font-family` list from left to right, and uses the first font value it finds installed on the browser's computer and skips the other fonts. So if `Courier` and `Prestige` are both installed on a computer, the browser uses the Courier font because it appears further left in the list.

There are lots of fonts, and different browsers support different ones. Users are able to install fonts in addition to the ones provided by their browsers. For a small sample of popular font names, see https://www.w3.org/TR/css-fonts-3/#generic-font-families. On that website, you can see that most font names use title case (the first letter of each word is capitalized), but a few use all lowercase letters. Font names are case sensitive on some operating systems, so you should take care to use the proper case. The font names that use all lowercase letters are special, and they are known as generic fonts.

A *generic font* is a name that represents a group of fonts that are similar in appearance. For example, `monospace` is a generic font, and it represents all the fonts where each character's width is uniform. Whenever you use a `font-family` CSS rule, you should include a generic font at the end of the rule's list of font names. In the following CSS rule (copied from earlier for your convenience), note the `monospace` font at the end of the list of font names:

```
.ascii-art {font-family: Courier, Prestige, monospace;}
```

The generic font name provides a *fallback mechanism*. If a user's browser doesn't support any of the fonts at the left of the generic font name, the browser uses the generic font to display a font that the browser does support. Generic font names are not actual fonts with specific appearance characteristics; they are just placeholders that tell the browser to look for an actual font in a particular family of fonts. So in the `ascii-art` CSS rule, if a browser does not support the `Courier` or `Prestige` fonts, the browser digs up a monospace font that it does support and uses it. To ensure that a web page's font looks pretty much the same on different browsers, for each `font-family` CSS rule, you should always use fonts that are similar, which means that they are associated with the same generic font. Referring once again to the preceding `ascii-art` CSS rule, `Courier` and `Prestige` both display uniform-width characters, and they are both associated with the same `monospace` generic font.

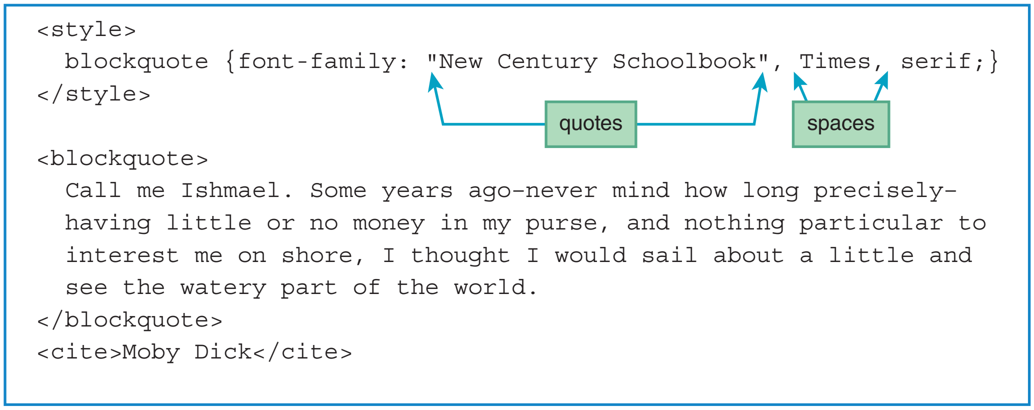
The `monospace` font is one of five generic fonts that all browsers recognize and support. Here's a list of the five generic fonts with descriptions and examples:

Generic Font Names	Description	Example Font
`monospace`	All characters have the same width.	`Courier New looks like this.`
`serif`	Characters have decorative embellishments on their endpoints.	Times New Roman looks like this.
`sans-serif`	Characters do not have decorative embellishments on their endpoints.	Arial looks like this.
`cursive`	Supposed to mimic cursive handwriting, such that the characters are partially or completely connected.	*Monotype Corsiva looks like this.*
`fantasy`	Supposed to be decorative and playful.	**Impact looks like this.**

As mentioned earlier, generic font names use lowercase, and you can verify that for the five generic fonts shown. On the other hand, specific font names use title case. You can verify that by examining the five example fonts shown in the right column (e.g., `Courier New`).

When specifying a multiple-word font name, surround the name with quotes. For example, in **FIGURE 3.16**, note the quotes around `New Century Schoolbook`. In addition, note the coding convention of inserting a blank space after each comma in a `font-family` list of font names.

Suppose a web page includes the code in Figure 3.16. If a user's computer has the `New Century Schoolbook` font and also the `Times` font installed on it, which font will the browser use to display the paragraph? It will use `New Century Schoolbook`, because it's the first font in the list and it's installed on the user's computer.



```
<style>
  blockquote {font-family: "New Century Schoolbook", Times, serif;}
</style>

<blockquote>
  Call me Ishmael. Some years ago-never mind how long precisely-
  having little or no money in my purse, and nothing particular to
  interest me on shore, I thought I would sail about a little and
  see the watery part of the world.
</blockquote>
<cite>Moby Dick</cite>
```

FIGURE 3.16 **An example font-family CSS rule**

`font` Shorthand Property

Fairly often as a web programmer, you'll want to apply more than one of the prior font-related properties to an element. You could specify each of those font properties separately, but there's an easier way. The `font` property can be used to specify all these more granular font properties—`font-style`, `font-variant`, `font-weight`, `font-size`, `line-height`, and `font-family`. Previously, we mentioned all these font properties except for `line-height`. We'll cover `line-height` shortly, but first we'll discuss some overarching details about the `font` property.

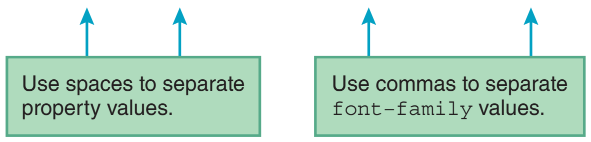
Here's the syntax for a `font` property-value pair:

`font:` [*font-style-value*] [*font-variant-value*] [*font-weight-value*]
font-size-value [/*line-height-value*] *font-family-value*

As usual, the italics are the book's way of telling you that the italicized thing is a description of what goes there, so for a `font` property-value pair in a CSS rule, you would replace *font-style-value* with one of the `font-style` values, such as `italic`. The square brackets are the book's way of telling you that the bracketed thing is optional, so the `font-style`, `font-variant`, `font-weight`, and `line-height` values are all optional. On the other hand, the `font-size` and `font-family` values have no square brackets, so you must include them whenever you use the `font` property. For the property values you decide to include, they must appear in the order previously shown. If a `line-height` value is included, you must position it at the right of the `font-size` value, with a / separating the two values.

Here's an example type selector rule that uses a `font` property:

```
blockquote {
  font: italic large "Arial Black", Helvetica, sans-serif;
}
```


Use spaces to separate property values.
Use commas to separate font-family values.

In the preceding rule, how many types of font properties are specified and what are they? Glancing at the `font` property's value, you can see there are five items in the list, so you might think there are five types of font properties. Upon closer inspection, you should notice the commas between the last three items. Those commas are delimiters for a sublist, where the sublist is the value for the `font-family` property. At the left of the font-family property's list, you can see two other property values—`italic` and `large`—separated by spaces. Those values are for the `font-style` and `font-size` properties.

By the way, the W3C refers to the `font` property as a *shorthand property* because it's a time-saving construct for handling multiple font characteristics. Later, we'll introduce additional shorthand properties for other groups of related CSS properties. When given a choice between using a shorthand property and using a set of more granular properties, some web programmers prefer using the shorthand property because of its compactness, whereas others prefer using the more granular properties because of their clarity. In this book, we use both techniques and let the situation dictate our preference.