Recommendation: One popular solution is to use a combination of percentages and ems to avoid the problems associated with both. This method was first introduced by Owen Briggs as a conclusion to his deep exploration of browser font-size differences. The method works by making the text slightly smaller with a percentage at the body level. Then use ems on the individual elements that you'd like to be larger than the surrounding text. Here is an example using his suggested values:

```
body {font-size: 76% }  /* results in 12 pixel text when the base size is 16
pixels */
p {font-size: 1em; }
h1 {font-size: 1.5em; }
```

The advantage is that the percentage value gives you more fine-tuned control, and the em sizing doesn't compound the way percentages do. The disadvantage is that if the base size is less than 16 pixels, everything may appear too small. However, because the sizes are specified in ems, resizing text in the browser is an option for users.

See all 264 of Owen Briggs' screenshots, as well as solutions for dealing with inconsistent font sizing, at *thenoodleincident.com/tutorials/box_lesson/font/index.html*.

Other Font Settings

Compared to the hassles of font-face and font-size, the other font-related properties are a walk in the park (albeit, a short walk). This section introduces style properties for adjusting font weight, style, and "small caps" display.

Font Weight

The font-weight property specifies the weight, or boldness, of the type.

font-weight

Values: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit

Initial value: normal

Applies to: All elements

Inherited: Yes

Font weight can be specified either as a descriptive term (normal, bold, bolder, lighter) or as one of the nine numeric values listed above. The default font weight is normal, which corresponds to 400 on the numeric scale. Typical bold text corresponds to 700 on the numeric scale. There may not be a font face within a family that corresponds to each of the nine levels of boldness (some may come in only normal and bold weights). Figure 18-6 shows the effect of each of the values on the

popular Verdana web font face in the Firefox browser (note that bold kicks in at 600, not 700).

It is evident that the numeric `font-weight` values are not useful when multiple weights are not available for the font. There's no harm in using them, but don't expect them to change the weights of an existing font. It merely looks for font weights that are already available.

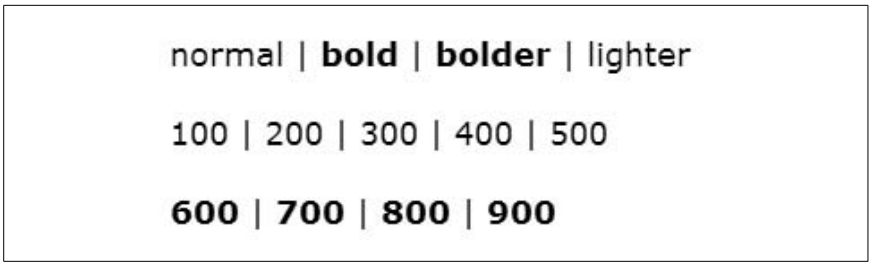

Figure 18-6. The effect of font-weight values

Unfortunately, the current browsers are inconsistent in support of the `font-weight` property, mainly due to the lack of available fonts that fit the criteria. The values that are intended to make text lighter than normal weight are particularly unsuccessful. Of the possible values, only `bold` and `bolder` will render reliably as bold text. Most developers stick to those values and ignore the rest.

Font Style

`font-style` controls the *posture* of the font, that is, whether the font is italic, oblique, or normal.

font-style

Values: `normal | italic | oblique | inherit`

Initial value: `normal`

Applies to: All elements

Inherited: Yes

Italic and oblique are both slanted versions of the font. The difference is that the italic version is usually a separate typeface design with more curved letter forms, while oblique text takes the normal font design and displays it on a slant using mathematical calculations, as shown in Figure 18-7 (top). At small text sizes on low resolution monitors, italic and oblique text may look exactly the same (Figure 18-7, bottom).

```
<p style="font-style: oblique">This is a sample of oblique Times as rendered
in a browser.</p>
<p style="font-style: italic">This is a sample of italic Times as rendered
in a browser. </p>
```

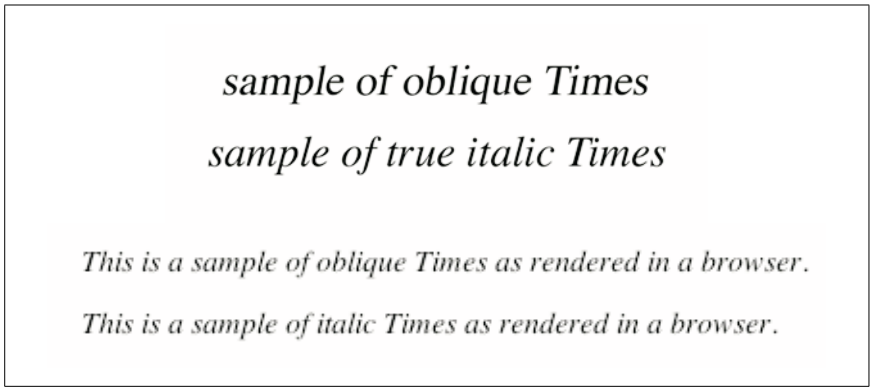

Figure 18-7. Comparison of oblique and italic type set with the font-style property

Font Variant

The sole purpose of the `font-variant` property is to specify that text should appear as small caps. Small caps fonts use smaller uppercase letters in place of lowercase letters. More values may be supported for this property in future style sheet versions.

font-variant

Values: `normal | small-caps | inherit`

Initial value: `normal`

Applies to: All elements

Inherited: Yes

If a true small caps font face is not available, the browser may simulate small caps by displaying all caps at a reduced size. Figure 18-8 shows such a simulation using this style rule.

```
<span style="font-variant: small-caps">lorem ipsum dolor sit amet,</span>
consectetuer adipiscing elit. Pellentesque pharetra, urna in laoreet
tincidunt, nunc quam eleifend libero, a tincidunt purus augue eu felis.
Phasellus quis ante. Sed mi.
```

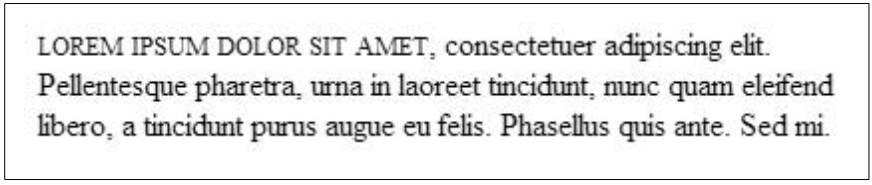

Figure 18-8. Using font-variant for small caps

Unlike a true small caps typeface design, the proportions of the capital and small cap letters do not blend well because the line weight of the small caps has been reduced. One use of small caps typefaces in the print world is to reduce the size of acronyms so they do not stand out like sore thumbs in the flow of text. Unfortunately, the `font-variant` property only transforms lowercase letters, so it cannot be used for this purpose.

There are two additional font-related properties in CSS 2 that were dropped in CSS 2.1 due to lack of support. The `font-stretch` property was for making a font's characters wider or more narrow using these keyword values: `normal`, `wider`, `narrower`, `ultra-condensed`, `extra-condensed`, `condensed`, `semi-condensed`, `semi-expanded`, `expanded`, `extra-expanded`, `ultra-expanded`, and `inherit`. The other dropped property is `font-size-adjust`, which was intended to compensate for the varying x-heights of fonts at the same size settings.

Putting It All Together with the font Property

Specifying multiple font properties for each text element could get repetitive and lengthy, so the authors of CSS provided the shorthand `font` property that compiles all the font-related properties into one rule. Technically, `font` is more than just a shorthand property, because it is the only property that allows authors to specify fonts from the operating system of the user agent.

font

Values: `[[<'font-style'> || <'font-variant'> || <'font-weight'>]? <'font-size'> [/<'line-height'>]? <'font-family'> ] | caption | icon | menu | message-box | small-caption | status-bar | inherit`

Initial value: Uses individual property default values

Applies to: All elements

Inherited: Yes

When using the `font` property as shorthand for a number of font properties, the order in which the property values appear is important. All of these font rules show correct usage of the `font` property.

```
h1 { font: 1.75em sans-serif; } /* minimum value list for font */
h1 { font: 1.75em/2 sans-serif; }
h1 { font: bold 1.75em sans-serif; }
h1 { font: oblique bold small-caps 1.75em Verdana, Arial, sans-serif; }
```

The rule may include values for all of the properties or a subset, but it *must* include `font-size` and `font-family`, in that order, as the last two properties in the list. Omitting one or putting them in the wrong order causes the entire rule to be invalid. These examples are invalid:

```
h1 { font: sans-serif; } /* font-size omitted */
h1 { font: 1.75em/2 sans-serif oblique; } /* size and family come first */
```