BACKGROUND

Backgrounds can be tricky. Nevertheless, effective when condensed correctly. The syntax for declaring the background shorthand values are as follows:

BACKGROUND PROPERTIES

```
element {

  background-color: color || #hex || (rgb / % || 0-255);

  background-image:url(URI);

  background-repeat: repeat || repeat-x || repeat-y || no-repeat;

  background-position: X Y || (top||bottom||center) (left||right||center);

  background-attachment: scroll || fixed;

}
```

Believe it or not, all these properties can be combined into one single **background** property as follows:

THE BACKGROUND SHORTHAND PROPERTY

```
element {

  background:

    #fff

    url(image.png)

    no-repeat

    20px 100px

    fixed;

}
```

THE UNKNOWN

Often times developers find themselves wondering "What if I leave out this value or that one? How will that effect the design?". Good questions. By default, the background property will assume the following when you**do not** declare each value of the properties.

DEFAULT BACKGROUND PROPERTY VALUES

```
element {

  background-color: transparent;

  background-image: none;

  background-repeat: repeat;

  background-position: top left;
  background-attachment: scroll;

}
```

Lesson learned: *be careful on what you don't declare. By chosing to not declare a value on a shorthand property, you are explicitly declaring the above default settings.* For example, let's look at the following example.

BACKGROUND SHORTHAND EXAMPLE (UNEXPLICIT)

```
element {

  background:red url(image.png);

}
```

This would be the same as declaring the following values:

BACKGROUND SHORTHAND EXAMPLE (EXPLICIT)

```
element {

  background:red url(image.png) repeat top left scroll;

}
```

FONT

Font is perhaps the trickiest. However, it follows the same rules as the background shorthand property. *All that you do not declare will have unexplicit values*. Here is the font shorthand specification:

FONT PROPERTIES

```
element {

  font-style: normal || italic || oblique;

  font-variant:normal || small-caps;

  font-weight: normal || bold || bolder || || lighter || (100-900);

  font-size: (number+unit) || (xx-small - xx-large);

  line-height: normal || (number+unit);

  font-family:name,"more names";

}
```

The default values for the font shorthand property are as follows:

DEFAULT FONT PROPERTY VALUES

```
element {

  font-style: normal;

  font-variant:normal;
  font-weight: normal;

  font-size: inherit;

  line-height: normal;

  font-family:inherit;

}
```

And of course without any further ado. The font shorthand property syntax:

THE FONT SHORTHAND PROPERTY

```
element {

  font:

    normal

    normal

    normal

    inhert/

    normal

    inherit;

}
```

Here is where it gets tricky. The fact that `font-style`, `font-variant`, and `font-weight` all come "normal" out of the box, you may need to pay a little more close attention when you're styling elements that come with default browser styles like <h1> - <h6> or <strong> and <em>. For example, styling the strong element:

STRONG ELEMENT STYLED WITH FONT

```
strong {

  font:12px verdana;

}
```

By writing the above into your style sheet, you will be unexplicitly removing the **font-weight:bold** default browser style that is applied to strong elements. Last but not least (for -font- that is), a real world example:

FONT SHORTHAND PROPERTY EXAMPLE (UNEXPLICIT)

```
p {

  font:bold 1em/1.2em georgia,"times new roman",serif;

}
```

This would be the same as declaring the following properties:

THE FONT SHORTHAND PROPERTY (EXPLICIT)

```
p {

  font-style:normal;

  font-variant:normal;

  font-weight:bold;

  font-size:1em;

  line-height:1.2em;

  font-family:georgia,"times new roman",serif;

}
```

BORDER

Let's not waste time discussing the warnings. The same rules apply from here on out. This is all you need to know

BORDER PROPERTIES

```
element {

  border-width: number+unit;

  border-style: (numerous);

  border-color: color || #hex || (rgb / % || 0-255);

}
```

becomes this:

THE BORDER SHORTHAND PROPERTIE

```
element {

  border:

    4px

    groove

    linen

}
```

Don't ask me how that would look. The fact that "linen" is in there, things could get scary. Nevermind the matter, here is where 'border' gets funny.

BORDER EXAMPLES

```
p {

  border:solid blue;

}

/* will create a '3px' solid blue border...
```

```
   who knows where 3px came from?? */


p {

  border:5px solid;

}

/* will create 5px solid 'black' border...

    default must be black?? */


p {

  border:dashed;

}

/* will create a '3px' dashed 'black' border...

    3px black lines unite! */




p { border:10px red; }

p { border:10px; }

p { border:red; }

/* these just don't even work */
```

One thing to specially take note about declaring a border without a color, the default will be 'black' unless otherwise noted through an explicit or inherited 'color' property. See the following examples:

BORDER COLOR EXAMPLES

```
p {

  border:dotted;

  color:red;

}

/* will create a 3px dotted red border */

/* ----------------------------- */

body {
  color:blue;
}
body p {
  border:5px solid;
}
/* will create a 5px solid blue border */
/* ----------------------------- */
```

Get it? Got it. Good! (isn't that a song?) Anyway. On with this

MARGIN AND PADDING

These are by far the easiest. Just think about the hands of a clock starting at noon, and follow the hour. For the sake of brevity, we'll be working with margin (since it's a shorter word). So for all cases of margin, the same rules apply to padding.

MARGIN PROPERTIES.

```
element {
  margin-top: number+unit;
  margin-right: number+unit;
  margin-bottom: number+unit;
  margin-left: number+unit;
}
```

... combined into the margin superpowers:

THE MARGIN SHORTHAND PROPERTY

```
/* top right bottom left */
element {
  margin: auto auto auto auto;
}
```

Of course, you may declare your margin with one, two, three, or four values. Here is how each scenario will be played out:

MARGIN FUN

```
/* adds a 10px margin to all four sides */
element {
  margin:10px;
}
```

```
/* adds a 20px margin to top and bottom

    and a 5px margin to left and right */

element {

  margin:20px 5px;

}



/* adds a 50px margin to top

    and a 10px margin to left and right

    and a 300px margin to bottom */

element {

  margin:50px 10px 300px;

}
```

Understood? Let's keep going. This is fun isn't it! (whatever, you like it).

OUTLINE

Quite frankly, this property has dropped off the existence of the design radar. Mainly because of lack of browsers supporting this CSS 2.1 standard (yep, it's an actual property), but nonetheless, it too has a shorthand property. This property follows the exact same (or same exact - they mean the same thing) specification as the 'border' shorthand property. But, for purposes of this being a Guide, it must be here. So:

OUTLINE PROPERTIES

```
element {

  outline-width: number+unit;

  outline-style: (numerous);

  outline-color: color || #hex || (rgb / % || 0-255);

}
```

Outline written as shorthand:

OUTLINE SHORTHAND PROPERTY

```
element {

  outline:3px dotted gray;

}
```

For purposes of trying to keep things from repeating, please see the border shorthand section on this document to understand the odds, ends, and quirks of the outline property.

LIST-STYLE

This is it. The last one. It's rarely used frequently. Hence rarely. That is why I kept it until the end (sorry, the best was first in my own opinion). Here is the list-style properties:

LIST-STYLE PROPERTIES

```
element {

  list-style-type: (numerous);

  list-style-position:inside || outside;

  list-style-image:url(image.png);

}
```

Here is the defaults:

LIST-STYLE PROPERTY DEFAULTS

```
element {

  list-style-type:disc;

  list-style-position:outside;

  list-style-image:none;

}
```

And for the sake of final brevity. Here is a simple example:

LIST-STYLE SHORTHAND PROPERTY EXAMPLE

```
ul li {

  list-style:square inside url(image.png);

}

/* in this particular case if image.png is not available

    then a square will be provided as secondary */
```