Efficient CSS with shorthand properties

I get a lot of questions about CSS from people who aren’t crazy enough to have spent the thousands of hours working with CSS that I have. Sometimes I’m asked to take a look at something they’re working on to see if I can figure out why it doesn’t work as expected. When I look at their CSS I often find that it’s both bloated and unorganised.

One of the reasons for using CSS to layout websites is to reduce the amount of HTML sent to site visitors. To avoid just moving the bloat from HTML to CSS, you should try to keep the size of your CSS files down as well, and I thought I’d explain my favourite CSS efficiency trick: shorthand properties. Most people know about and use *some* shorthand, but many don’t make full use of these space saving properties.

Some background

Shorthand properties can be used to set several properties at once, in a single declaration, instead of using a separate declaration for each individual property. As you’ll see, this can save **a lot** of space in your CSS file.

Quite a few shorthand properties are available – for details I suggest the W3C CSS specifications of the background, border,border-color, border-style, border sides (border-top, border-right, border-bottom, border-left), border-width, font, list-style, margin,outline, and padding properties.

Colours

The most common way of specifying a colour in CSS is to use hexadecimal notation: an octothorpe (#) followed by six digits. You can also use keywords and RBG notation, but I always use hexadecimal. One great shortcut that many don’t know about is that when a colour consists of three pairs of hexadecimal digits, you can omit one digit from each pair:

`#000000` becomes `#000`, `#336699` becomes `#369`.

Box dimensions

The properties that affect box dimensions share the same syntax: the shorthand property followed by one to four space separated values:

- `property:value1;`
- `property:value1 value2;`
- `property:value1 value2 value3;`
- `property:value1 value2 value3 value4;`

Which sides of the box the values affect depends on how many values you specify. Here’s how it works:

- One value: all sides
- Two values: top and bottom, right and left
- Three values: top, right and left, bottom
- Four values: top, right, bottom, left

Thinking of the face of a clock is an easy way of remembering which side each value affects. Start at 12 o'clock (top), then 3 (right), 6 (bottom), and 9 (left). You can also think of the**TRouBLe** you'll be in if you don't remember the correct order – I first saw this in Eric Meyer's excellent book Eric Meyer on CSS.

Margin and padding

Using shorthand for these properties can save a lot of space. For example, to specify different margins for all sides of a box, you could use this:

```
margin-top:1em;
margin-right:0;
margin-bottom:2em;
margin-left:0.5em;
```

But this is much more efficient:

```
margin:1em 0 2em 0.5em;
```

The same syntax is used for the `padding` property.

Borders

Borders are slightly more complicated since they can also have a style and a colour. To give an element a one pixel solid black border on all sides, you could use the following CSS:

```
border-width:1px;
border-style:solid;
border-color:#000;
```

A more compact way would be to use the `border` shorthand:

```
border:1px solid #000;
```

I always specify border values in that order:

```
border:width style color;
```

Most browsers don't care about the order, and according to the specification they shouldn't, but I don't see a reason for not using the same order as the W3C does in the specification. There's always the chance of a browser being very strict about the order of shorthand values.

The same syntax can be used with the `border-top`, `border-right`,`border-bottom`, and `border-left` shorthand properties to define the border of any single side of a box.

You don't have to specify all three values. Any omitted values are set to their initial values. The initial values are `medium` for `width`,`none` for `style`, and the value of the element's `color` property for`color`.

How wide a medium border is depends on the user agent.

Note that since the initial value for `style` is `none` you do need to specify a `style` if you want the border to be visible.

The `border-width`, `border-style`, and `border-color` properties used in the first border example above are themselves shorthand properties. Their longhand alternatives are very rarely used, but they do exist:

```
border-width:1px 2px 3px 4px;
```

is shorthand for

```
border-top-width:1px;
border-right-width:2px;
border-bottom-width:3px;
border-left-width:4px;
```

The `border-style` and `border-color` shorthands use the same syntax as `border-width`: the box dimensions syntax described above.

Using the various border shorthands can also save some typing when you want to give an element's border different properties on different sides. These declarations will make an element's right and bottom borders solid, black, and one pixel wide:

```
border-right:1px solid #000;
border-bottom:1px solid #000;
```

And so will these:

```
border:1px solid #000;
border-width:0 1px 1px 0;
```

First the borders on all sides are styled identically, and then the different widths are specified.

Backgrounds

Another very useful shorthand property is `background`. Instead of using `background-color`, `background-image`, `background-repeat`,`background-attachment`, and `background-position` to specify an element's background, you can use just `background`:

```
background-color:#f00;
background-image:url(background.gif);
background-repeat:no-repeat;
background-attachment:fixed;
background-position:0 0;
```

can be condensed to

```
background:#f00 url(background.gif) no-repeat fixed 0 0;
```

Like with the border shorthands the order of the values isn't**supposed** to matter, but I've seen reports of early versions of Safari having problems when the values aren't listed in the order used in the W3C specification, which is this:

```
background:color image repeat attachment position;
```

Remember that when you give two values for `position`, they have to appear together. When using length or percentage values, put the horizontal value first.

As with the `border` and border sides properties, you don't have to specify all values. If a value is omitted, its initial value is used. The initial values for the individual background properties are as follows:

- `color`: `transparent`
- `image`: `none`
- `repeat`: `repeat`
- `attachment`: `scroll`
- `position`: `0% 0%`

This means that it's pointless to use the `background` shorthand without giving a value for either `color` or `image` – doing so would make the background transparent.

I almost always use the `background` shorthand to specify background colours for elements, since `background:#f00;` is the same as `background-color:#f00;`.

Remember that this will remove any background image specified by a previous rule. Consider these rules:

```
p {
background:#f00 url(image.gif) no-repeat;
}
div p {
```

```
background:#0f0;
}
```

All paragraphs not in a `div` element will have a background image and be red where the image doesn't cover the background. Any paragraph that is in a `div` will have a green background, and no background image.

Fonts

As with the `background` property, `font` can be used to combine several individual properties:

```
font-style:italic;
font-variant:small-caps;
font-weight:bold;
font-size:1em;
line-height:140%;
font-family:"Lucida Grande",sans-serif;
```

Can be combined into

```
font:italic small-caps bold 1em/140% "Lucida Grande",sans-serif;
```

Again, when it comes to the order of the values, I see no reason not to use the order given by the W3C. Better safe than sorry.

When using the `font` shorthand you can omit any values **except**`font-size` and `font-family` – you always need to give values for those, and in that order. The initial values for the individual `font`properties are these:

- `font-style`: `normal`
- `font-variant`: `normal`
- `font-weight`: `normal`
- `font-size`: `medium`
- `line-height`: `normal`
- `font-family`: depends on the user agent

Lists

The shorthand property for ordered and unordered lists is `list-style`. I personally only use it to set the `list-style-type` property to `none`, which removes any bullets or numbering from the list:

```
list-style:none;
```

instead of

```
list-style-type:none;
```

You can also use it to set the `list-style-position` and `list-style-image` properties, so to specify that unordered lists should render their list item markers inside each list item, use an image for the list item markers, and use squares if that image is not available, the following two rules would do the same thing:

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

is shorthand for

```
list-style-type:square;
list-style-position:inside;
list-style-image:url(image.gif);
```

Outlines

The `outline` property is very rarely used, mainly because of its current poor browser support – as far as I know only Safari, OmniWeb and Opera currently support it. Anyway, using the individual properties you can define an outline like this:

```
outline-color:#f00;
outline-style:solid;
outline-width:2px;
```

or like this:

```
outline:#f00 solid 2px;
```

Outlines have some interesting characteristics that make them useful: unlike borders, they do not take up any space and are always drawn on top of a box. This means that hiding or showing outlines doesn't cause reflow, and they don't influence the position or size of the element they are applied to or that of any other boxes. Outlines may also be non-rectangular.

Reduced file size and easier maintenance

Those are the shorthand properties available in CSS 2. If you were to take the CSS file of a fairly large site and make one version that uses no shorthand properties and another version that uses shorthand efficiently, you would see a huge difference in file size. That's one reason for using shorthand. Another is that doing so makes your CSS files easier to maintain – at least that's my experience.

Got any other tips related to CSS shorthand? Let us know.