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:

```
01. margin-top:1em;
02. margin-right:0;
03. margin-bottom:2em;
04. margin-left:0.5em;
```

But this is much more efficient:

```
01. 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:

```
01. border-width:1px;
02. border-style:solid;
03. border-color:#000;
```

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

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

I always specify border values in that order:

```
01. 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:

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

is shorthand for

```
01. border-top-width:1px;
02. border-right-width:2px;
03. border-bottom-width:3px;
```

```
04. 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:

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

And so will these:

```
01. border:1px solid #000;
02. 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`:

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

can be condensed to

```
01. 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:

```
01. 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:

```
01. p {
02. background:#f00 url(image.gif) no-repeat;
03. }
04. div p {
05. background:#0f0;
06. }
```

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:

```
01. font-style:italic;
02. font-variant:small-caps;
```

```
03.  font-weight:bold;
04.  font-size:1em;
05.  line-height:140%;
06.  font-family:"Lucida Grande",sans-serif;
```

Can be combined into

```
01.  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:

```
01.  list-style:none;
```

instead of

```
01.  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 elements. 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.

Translations

This article has been translated into the following languages:

- **Chinese:** 高效CSS属性缩写
- **Italian:** CSS efficienti con l'uso delle shorthand (Translation by Vincenzo Mania)
- **Polish:** Efektywny CSS ze skróconymi właściwościami (Translation by Piotr Janeczek)

Possibly related posts

- Remember non-vendor-prefixed CSS 3 properties (and put them last)
- New CSS properties in Safari
- Efficient creation of sensible and usable forms
- Useful tips for writing efficient CSS

Posted on February 21, 2005 in CSS

Comments

1. February 21, 2005 by Charles Martin

   One of the shorthands that I think is probably important (and you used it throughout the article, but made no reference to it) is that of the RGB identifiers. Since many sites must use web-safe colors, it makes more sense to use #FA9 instead of #F0A090. A very minor saving of file size, but a shorthand nonetheless.

2. February 21, 2005 by Roger Johansson *(Author comment)*

   Charles: Yeah I somehow missed that, probably because it comes so naturally to me after using it for years. I'll update the article to include it.

3. February 21, 2005 by Thomas

   This is an excellent article to remind us developers the power of CSS and the importance of keeping file sizes to a minimum. I always try to use the shorthand method when coding my CSS and found this article quite helpful in reminding me of this technique. Another very well written article. Thank You.

4. February 21, 2005 by Charles Martin

Oh, and forgive me for being rude, but it was a great article with excellent points about the order of values used in shorthand properties.

5. February 21, 2005 by icaaq

   One thing i often see is that people set font-attributes to every element. yuck.....

   mv icaaq

6. February 21, 2005 by Josh Bryant

   Its funny, I never learned CSS without shorthand values for some reason. Every once in awhile when I am helping someone with CSS and see the long-hand method they have done, I just sit there and try to figure out why they did it that way, then of course realizing that they just didn't know shorthand.

   I have always wanted borders to be simpler however, I've long ben frustrated typing out

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

   Thanks for your suggestion on just using the border: width; property to control it!

   And yes, I too find the RGB shorthand regular as well, definitely something else to add to this article.

   Thanks Roger.

   P.S. I'm all about optimizing CSS, and lately I have been throwing around the idea of stripping all the whitespace out of CSS. What's your take on that? And do you know of an app on OS X that will do it?

7. February 21, 2005 by Greg Hinch

   Great little article, I knew about most of these but the font one always threw me. It seems to be particulary strict about getting the right order, or it simply doesn't work.

   Also thought I should point out in the above comment by Charles, #FA9 would be the same thing as #FFAA99, not #F0A090.

8. February 21, 2005 by Roger Johansson *(Author comment)*

   Charles: Don't worry, I didn't think you were being rude :-)

   Greg: Nice catch. I got so busy adding something about the colour shorthand that I didn't

look close enough at Charles' example to notice that mistake.

9. February 21, 2005 by Nick Rigby

   Good article.

   One thing you might of mentioned is this:

   No need to specify the unit, if something has a value of 0 (zero) i.e.

   margin-right: 0px;

   If it's 0, then no unit is required. Should be:

   margin-right: 0;

   I see a lot of people doing this, and although it's not a major sin, or the reason for bloated style sheets, it can make your CSS look untidy.

   Nick.

10. February 21, 2005 by Colin Stein

    Whenever I'm trying to clean up a site I usually run it through the CSS Optimizer. It's not perfect and you'll still have to check things over but it's a good way to make code that uses 12 rules to specify borders a little more manageable. I treat it like Tidy for CSS.

11. February 21, 2005 by Roger Johansson *(Author comment)*

    Josh: I've been thinking a bit about stripping whitespace from deployed CSS files, either by doing it manually (well, with the help of an application) or on the server before sending the file. I'd definitely keep non-stripped copies for development work though ;-)

    BBEdit will do it for you - Markup > Utilities > Optimize.

12. February 21, 2005 by Charles Martin

    Thanks, Greg. I used to use that other method (#F0A090) until I found out recently that it wasn't correct. Still a bad habit... which is why I switched to the shorthand. ;)

13. February 21, 2005 by mateo

    A better idea than removing all the whitespace would be to enable gzipping on your webserver, which will compress all that whitespace for you...

14. February 21, 2005 by Charles Martin

    Nick, As you note, it doesn't hurt to have the units identified on a value of zero. However, it can be a bad habit to forget to put the units when putting in a non-zero value. Thus, having the units specified is probably just a sign of a good habit. :)

15. February 21, 2005 by Rimantas

    Amazing, how long does myth about "web-safe" palette live.

    > *It is also worth noting that the three-digit hexadecimal notation only contains colours from the "web-safe" palette.*

    This is not true. To make it true you should limit heximal digits to these: 0,3,6,9,C and F because "web safe" colors ar those having R,G and B values of these triplets: 00,33,66,99,CC and FF (0,51,102,204,255 in decimal format).

    FA9 does not belong to safe palette because of "A". #FC9 (or #FFCC99) — does.

16. February 21, 2005 by Rimantas

    Oops, I've left 153 (for 0x99) from decimals list. Sorry.

17. February 21, 2005 by Charles Martin

    Doh... I didn't realize that either.. I have gotten lazy. Of course, what I had mentioned before would result in 4096 colors, it is far over the 216 colors that is the web-safe palette. *sigh*... how often must I open my mouth and show my ignorance? Much more, I fear. Much more.

18. February 21, 2005 by Roger Johansson *(Author comment)*

    Rimantas, Charles: Yeah I got sloppy and typed too fast without stopping to think. That's what I get for doing edits on the fly. I removed the part about web-safe colours.

    And no, web-safe colours aren't something I spend a lot of time thinking about anymore. It's been several years since I worked on a project where that was even mentioned.

19. February 21, 2005 by Eddie Sowden

    One thing that I dont think many people know about shortening css is that some things like colours can be omited and inherited.
    For example with the border if you want it the same colour as the text you only need:
    border: 1px solid;

Other than that nice article..

20. February 22, 2005 by Tom

    Another handy little post, when working I tend to avoid using shorthands then when its time to upload I get Topstyle to shrink everything down.

21. February 22, 2005 by Tommy

    If you find it difficult to remember which sides are affected when using two or three values with properties like margin, here's a tip:

    Imagine the four TRouBLe positions being filled from the left. Any omitted value is the same as the corresponding 'other' side.

    With two values, top and right are specified. Bottom will be equal to top and left will be equal to right.

    With three values, top, right and bottom are specified. Left will be equal to right.

22. February 22, 2005 by Douglas Clifton

    Consider **gesso** at the top of your master stylesheet:

```
* {
  margin: 0;
  padding: 0;
}
img { border: 0; }
```

    from then on, margins and padding are additive only. Plus, you start with a clean slate in all browsers.

    Break-up your stylesheets into modular units and import only those you need for a particular page.

    I agree with using Roger's points regarding shorthand properties, except when you find yourself repeating them. Consider background images as bullets and hover events:

```
selector {
  background-image: url(/img/bullet.gif);
  background-repeat: no-repeat;
```

```
  background-position: 0 100%;
}

selector:hover {
  background-image: url(/img/hover-bullet.gif);
}
```

My stylesheets are large, I have many of them. I do many things that other people don't bother with. For instance, adding focus events along with hover, so your links highlight in the same way when someone tabs through your pages. White space and comments in code are important in development, and maintenance. That's what mod_gzip is for.

Don't get me started. Oops! Too late. ;-)
~d

23. February 22, 2005 by Nick Rigby

    Charles. Not specifying a unit identifier for a non-zero value is more than a bad habit, it's invalid :-). Specifying a unit identifier for a zero value is optional and, in my opinion, uneccessary and wrong.

24. February 22, 2005 by Adam Taylor

    Happily, Mozzer Firefox also understands Outline nowadays, although I must admit I haven't experimented with different styles other than solid.

    As a 'decorative' addition, it's a nice micro-reward for non-IE-users.

25. February 22, 2005 by Gordon

    Excellent article. I think I knew about 80% of this. One question, do you write your CSS on a "one line per element" basis, or go with the line break before curly bracket model?

    And does it matter?

    Ohh and got spot with the font order info - gets me every time that one!

26. February 22, 2005 by Charles Martin

    "Specifying a unit identifier for a zero value is optional and, in my opinion, uneccessary and wrong." Unnecessary? Yes. Wrong? No. Unless the specifications forbid it, it is not wrong. Unless the browsers misinterpret it, it is not wrong. Unnecessary, but not wrong.

27. February 22, 2005 by Nick Rigby

    Charles, this is only my opinion.

    I say it is wrong, because it makes no logical sense. If a value is zero, then it doesn't matter if it's zero pixels or zero elephants. It's zero.

    If you want to keep your style sheets lean and mean, then why include completely redundant information?

28. February 22, 2005 by Charles Martin

    I'm not insisting that units be specified. I merely pointed out that "accidentally" putting them in there is just a sign of a good habit. Believe it or not, you can leave off the unit identifier and some browsers will assume what that unit is. In many cases, they assume correct and the developer "gets away with it". I would rather see two extra characters appear after a zero than see them left off a non-zero value and try to figure out why the heck two browsers are interpreting things differently. More importantly, the specifications for both CSS2.1 and CSS3 state that the unit identifier is *optional*. Thus, it cannot be wrong according to the specifications.

29. February 22, 2005 by Nick Rigby

    Charles,

    I appreciate what you are saying. I personally think it's more a case of people not knowing that it is optional to specify the unit with zero values. That's why I mentioned it in my original post. If you think about it, it doesn't make any sense to include it, and I see it more as a bad habit. It is the responsibility of the designer to include the appropriate unit when specifying non-zero values.

    To be fair, if you are comfortable with your approach, and specify the unit for zero values, then you should stick with it. As you say, it does not violate the spec. All I'm saying is that in the context of this article, it was an appropriate thing to mention, and a valid action for keeping your style sheets lean.

30. February 22, 2005 by Charles Martin

    Well, while we're on the subject of "saving bytes", the use of a semicolon after the last property specification within the {} braces is not really necessary either; however, it seems to be a good habit to always include it so that you don't have to remember to add it in there when you append another property setting.

All of this is getting down to the nit-picky stuff that, in some cases, is more a personal choice of coding-style versus actually saving as much time/data as possible or following standards.

31. February 22, 2005 by Colin

    Quick note: the first commenter noted that you could use the shorthand for RGB values, for example by only specifying "#FA9" instead of "#F0A090".

    Actually, "#FA9" is short for "#FFAA99", not "#F0A090".

32. February 22, 2005 by Justin Perkins

    Very interesting read, a must for those learning CSS. It's always easier to learn new principles with a clean slate than with a clouded past of long-winded rules and bloated CSS :)

    While on the subject of shorthand, I like to cut the element type out of the selectors when I can.

    So instead of:

    ```
    div#header h1.replace span{/*do something*/}
    ```

    One can do:

    ```
    #header .replace span{/*do something*/}
    ```

    Sure it's trivial, but it *does* add up.

    Of course if you're trying to override an existing selector that has more weight, you'll need the extra points from the element (div) for overriding.

33. February 22, 2005 by John Nunemaker

    I love shorthand and am amazed at the number of people who don't take advantage of it.

    Also, I have gotten into the habbit of single line declarations with good tabbing. I have found this to reduce the number of lines greatly and I feel as though it is still maintainable. Example: single line declarations

34. February 22, 2005 by Gordon

John, thanks for posting about what I was trying to mention “single line declarations” - I’ll remember that.

35. February 23, 2005 by Malarkey

    This is damn good stuff Roger. (Not the most original comment, I know, but sincere.)

36. February 23, 2005 by Jim Amos

    CSS Shorthand is one of those things we probably take for granted as seasoned professionals, but this is a valuable resource for those encountering style code for the first time. Good write up.

37. February 23, 2005 by Bryan

    Nice post man. Good stuff.

    One thing I noticed, and I could be wrong on this, but Opera sometimes barfs when looking at the no-repeat position in the code.

    I believe it either needs to be in the place you have it, or have it listed last.

    In my experience, I had an issue with opera not showing a background properly because the no-repeat wasn’t in the right spot.

    Just a note.

38. February 23, 2005 by Roger Johansson *(Author comment)*

    Thanks for the comments and encouraging feedback, people.

    Some of you have mentioned additional ways of saving bytes here and there. I wanted to keep this article focused on the CSS shorthand properties, so I left those out.

    I’m working on a little write-up of the tricks mentioned and a few other tips related to CSS. Coming soon to a website near you. Well, this site.

39. February 24, 2005 by Kim Siever

    > *you would **se** a huge difference in file size*

    :)

40. February 24, 2005 by Roger Johansson *(Author comment)*

Oops. A typo. Thanks for alerting me, Kim.

41. February 24, 2005 by Nicolas Chachereau

    About the web-safe palette matter: the 4096 colours that we can access through the "#fa9" form are sometimes called the "web-**smart**" palette… maybe that confused you.

42. February 24, 2005 by Roger Johansson *(Author comment)*

    Gordon: Looking through the comments I see that I neglected to answer your question about single-line rules. Most of the time I don't write my rules that way. I find it makes it difficult to scan through a CSS file. However, the example that John posted does look kind of tidy so who knows — I may reconsider.

    Nicolas: I think I was just in too much of a rush ;-)

43. February 24, 2005 by Gordon

    No problem Roger. It's one of those things which appears to be personal preference rather than having a "right or wrong" way to do. Although I'm sure I recall reading something about the single line approach being more efficient for the browser's parser?? Not sure.

44. February 25, 2005 by Creford

    I think it's convenienter to use the unit "px"(or omit it. The default is "px") instead of "em".

45. February 25, 2005 by Roger Johansson *(Author comment)*

    Creford: There is no "default" unit for length values in CSS. For non-zero values, you **must** specify a unit: Syntax and basic data types, 4.3.2 Lengths.

46. February 25, 2005 by Alexandru Mihai Bîrsan

    First of all, great article Roger! I don't get to browse alot this year because of my exams, but your website is a must see every once in a while.

    I myself use many of the guidelines in this article. I knew all of them, but I guess lazyness takes over when I actually write CSS files.

    I disagree with the whitespace removal from the file because of two reasons: I find it hard to keep two versions of the same file (one with whitespace, one without); second, maybe

somebody wants to look through my CSS file online (I do it alot), why not let him do that? Moreover, maybe I want to look throught my CSS file!? Also, comments are a good dev-habit. Oh, and… Single line declarations harm me.

I myself find unit specifcation to zero values unnecessary and wrong. And I also made a personal favorite quote out of Nick Rigby's post:

> *If a value is zero, then it doesn't matter if it's zero pixels or zero elephants. It's zero.*

Mmm… Great article, again!

47. February 26, 2005 by Creford

    Hi Roger, thank you for pointing it out! I've made a mistake.

48. March 1, 2005 by Joel Bernstein

    I suppose I have to be the anal retentive jerk and point out that there are several quantities (sound intensity and temperature are two that leap to mind) where zero doesn't mean "nothing", and a unit should be used. Since, without a unit, we have no idea if we're talking about decimeters or decibels, I say put a unit on everything to be safe.

    Does any of this matter when the only units used in a CSS file represent lengths? Probably not, but then again it was you who brought up elephants.

49. March 2, 2005 by Paul Dell

    Very good article, I use shorthand css and have been doing for some time. From a personal point of view I find it organises the css better and it is so much easier on the eyes :)

    I don't agree with one comment about zero and giving it a value, to me it makes no sense but I suppose it depends how we work. To me zero is zero :)

50. March 3, 2005 by Mordechai Peller

    So instead of:

    *div#header h1.replace span{/*do something*/}*

    One can do:

*#header .replace span{/*do something*/}*

The two are not the same. For example:

header h2.replace span{/*do something else*/}

Even div#header and #header can sometimes be different as one style sheet often is used for multiple pages.

It's also worth noting that

id1 #id2 * {}

isn't always interchangeable with

id2 * {}

as the former is more specific than the latter.

51. March 16, 2005 by Jewel

    Very helpful article for us newbies. Thank you very much. I am off to tidy up my stylesheets now :-)

52. March 26, 2005 by Aaron Schmidt

    I love the shorthand for color in CSS so much that I will purposely "round" the hex values for my colors to the closest shorthand value.

    For example, take the color **#85A0C9**:

    R: 85 -> round up to 88

    G: A0 -> round down to 99

    B: C9 -> round up to CC

    And we get the color **#8899CC** or **#89C**.

    Of course, #89C is not exactly the same as #85A0C9 but in most cases (especially with text color) you'll probably never notice the difference and you save yourself a few bytes in the CSS.

53. May 3, 2005 by woman's shoes

Very good article,i like CSS !

54. June 3, 2005 by David Latapie

    Nice article. I'm quite proficient with CSS optimisation so I did not learned anything about optimisation but thanks to you, I discovered outline and could compare with a previous hack with invisible borders - French

    You may be interested in my own article about CSS optimisation. It is in French but you can figure out just by reading the part (preferably with Opera, as I use pre-wrap) :

55. August 12, 2005 by Ove Klykken

    Good to see an in-depth explanation. I made an article on the same topic years ago because an explanation like yours did not exist yet. I'm happy to say I liked yours better than mine :)

56. August 15, 2005 by Steven

    So instead of:

    div#header h1.replace span{/do something/}

    One can do: #header .replace span{/do something/}

    One of the reasons behind removing the element in front of ID and class definitions was because Netscape 4 (correct me if I'm wrong) didn't understand it. Using #header instead of div#header, allowed Netscape 4 to display some of the CSS properties.

    As mention later by Mordechai (#50), being more specific adds to the specificity of an element (what a bad pun eh?). This is an issue when you have two elements within the same set of tags, I can't give an example off the top of my head, but there are some times when you would need like to specify the element tag along with the class.

57. September 18, 2005 by Brian K. Shoemake

    Re: zero values. It's true that you can save space by excluding zero values, but I find that when I am proofreading my own code, I sometimes hesitate at a missing value wondering if I may have left something out. I simply insert the zero from the start for my own sanity later on. =:]

58. December 26, 2005 by +rt+

Wonderful summary; I appreciate your explanations beyond just “do this.” The whys help us understand beyond straight-forward rules.

Thank you for your work and making it available to others!

59. January 19, 2006 by Sapphire

    You could also unite identical sections into one with seperating commas:

    Instead of:
    a.title:link {IDENTICAL CSS}

    a.title:active {IDENTICAL CSS}

    a.title:visited {IDENTICAL CSS}

    Do:
    a.title:link, a.title:active, a.title:visited
    {IDENTICAL CSS HERE}

    Not only convenient but cuts down the size of your CSS file dramatically.

60. January 27, 2006 by thrstn

    If you are doing that why not ..

    a.title a { IDENTICAL CSS }

    a.title a:hover { DIFFERENT CSS }

    cuts down even more.

61. March 19, 2006 by James A. Sablan Sr.

    This is incorrect method of shorthand
    http://www.456bereastreet.com/archive/200502/efficient*css*with*shorthand*properties/

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

    This is the correct method the correct method for shorthand example

    .font { font: 1em/140% bold, italic, small-caps, “Lucida Grande”, sans-serif; }

62. March 20, 2006 by Roger Johansson *(Author comment)*

James: Sorry, the example in the article is the correct shorthand for the font property when you want to specify font-style, font-variant, and font-weight along with font-size, line-height and font-family.

Your example is valid, but will make the browser try to find an installed font called “bold”, and if it can’t find that it will look for one named “italic”, then “small-caps”, then “Lucida Grande”, and finally “sans-serif”.

63. April 24, 2006 by A M Sarno

    Good article. Nothing I didn’t already know per se, but a good refresher. As for the comments; some good but much hair splitting and semantic polemics. Example, CSS relative positioning allows for negative pixels, but negative Elephants don’t exist either. It’s a personal choice to be made, leveraging saving a byte in bandwidth versus preemptively preventing future omissions. Keep in mind, shorthand is opposed to longhand. In other words, it is to remove semantic superfluosity via logically equivalent syntax. Logical entities can be inconclusive, but semantically the words Zero and Nothing mean two utterly different things. Synonymous but not equivalent. All in all, my comments amounts to Zero (giggles). If your true mission in life is to save bandwidth, don’t publish to the web. If it is to be efficient, as this artical is, then I applaud you.

    Also, for the person that wants to work without whitespace, I suggest using a semantic coloring text editor such as UltraEdit, allowing semantic coloring of multiple interpretive languages as well as customizable semantics. It’s served me well for a couple years now. Work hard play harder everyone.

64. May 21, 2006 by Andrew

    Thanks for thois info!

65. May 26, 2006 by chas

    is there an app that converts longhand to shorthand?

    none of the supposed optimizers seem to do this, which seems like the obvious (but i suppose more difficult to impliment) optimization.