

1. Basic Issues:

Choosing between px, pt, em, % and keywords

Typography for the Web IAP 2010

Andrew Whitacre
awhit@mit.edu

Everything included here is
available online at:
[www.fungibleconvictions.com/
web-typography/](http://www.fungibleconvictions.com/web-typography/)

You have a basic knowledge of CSS—cascading style sheets, the code that tells text on a website how it should look. An example CSS statement for text in a paragraph could look like this:

```
P {  
  FONT-FAMILY: "TIMES NEW ROMAN";  
  FONT-SIZE: 20px;  
  LINE-HEIGHT: 200%;  
  COLOR: #333;  
  TEXT-ALIGN: CENTER;  
}
```

Or in plain English, for all paragraphs, set...

- the typeface to Times New Roman
- the type size to 20 pixels
- the line spacing to double (so, 200% is the same as double-spacing)
- the text color to a medium gray
- and center the text

Choosing your measurement for text

But why, in that statement, would you choose pixels as the measurement for your font-size when “em”, “pt”, a percentage, or even a keyword such as “xx-small” could also be used? The answer has to do with context.

Pixels are absolute measurements, so that in a cascading stylesheet, pixels don’t “cascade”. If you set a parent element (p, for example) to be 12px and a child element (p > p.smalltext, for example) to be 10px, they will be 12px and 10px, respectively, on the nose.

But ems, percentages, and keywords are relative measurements. If `p` happens to be set in absolute terms like `12px` but `p.smalltext` is set to `.8em`, `p > p.smalltext` will be .8 of the size of `p` (9.6 px). The same goes for percentages.

Why would you use relative measurements like `em` and `%`? Because it keeps all of the text in relative balance no matter how it gets resized by the user.

Choosing your measurement for block elements

Text is one thing. But these measurement kinds play a crucial design and usability role with block-level elements—`div`s mostly.

It would make lots of sense, and is a best practice on the typical blog design, to have the main-text `div` have a width that changes as the user resizes his or her screen. (We'll talk about why this is a best practice ahead.) You can't do that with `px`, but you can with `em`s and `%`.

But for the sidebar, it does make sense to keep it a specific width—something you can only do with `px`.

To touch briefly on points (`pt`), they should be used *exclusively* for your print stylesheet. It's what tells the printer what size you want the text to be when a webpage is printed in hardcopy, and you don't want the user to muck with how text is displayed in print.

So to sum up, when choosing `pt`, `px`, `%`, etc., each element of your webpage has its own needs, and you want to choose the right thing for that specific element.

References

CSS Tricks: “`px – em – % – pt – keyword`”
<http://css-tricks.com/css-font-size/>

2. Basic Issues: Page/paragraph level, header level, and word level

Now we'll look at some of the basic typographical issues for the three main CSS "levels": pages and paragraphs, headers, and individual words.

Page/paragraph level

As mentioned in section 1, page- and paragraph-level typography mainly has to do with width considerations. You'll want to obsess about how wide your lines of text run: too narrow and you may push a lot of text below the fold, resulting in a lot of scrolling; too wide, and your readers' eyes get tired.

In print typography, it's called "choosing a comfortable measure". Traditionally, you want to keep lines limited to no more than 75 characters, with 66 characters being considered optimum. But on the web, it can and should be up to the user.

Therefore, take advantage of adjustable widths. You can set your text box width to be 50%, meaning no matter how wide the user sets their screen, the box will always fill 50% of it, with measures adjusting accordingly. Or you can set your text box to, say, 33em, meaning the box will always accommodate approximately 66 characters, with the box itself changing in size as the user makes adjustments.

Next, for the same reasons, obsess over your line-height. (This is akin to leading in print typography.) Each line needs space between the lines above and below, but too much or too little space strains the eyes, especially for long texts. It should suit the typeface, text level (body, heading, etc.), and measure. For example...

```
P {  
  FONT-SIZE: 12px;  
  LINE-HEIGHT: 1.25;  
}
```

...results in type that is 12px with a breathable, comfortable line-height, 1.25 times the font-size, and stays 1.25 times the size no matter how the user resizes the type.

Having a stable line-height also gives you the chance to square up lines of body text with lines of sidebar text, no matter the font-size.

Header level

Similar in thinking to using a good measure, establish a good rhythm with your headers.

Book designers will tell you, “Don’t compose without a scale,” meaning everything—*everything*—has to be in rational proportion, especially headers. For example, if the default body text happens to be 16px, you should set your headings and paragraph text to sizes in proportion to 16px:

```
BODY { FONT-SIZE:100%; }
H1 { FONT-SIZE:2.25EM; /* 16x2.25=36 */ }
H2 { FONT-SIZE:1.5EM; /* 16x1.5=24 */ }
H3 { FONT-SIZE:1.125EM; /* 16x1.125=18 */ }
H4 { FONT-SIZE:0.875EM; /* 16x0.875=14 */ }
P { FONT-SIZE:0.75EM; /* 16x0.75=12 */ }
```

Word level

At the word level, there’s actually very little agreement on best practices.

Historically, web designers argued that sans serif typefaces were all you should use for main text. And that was because screen resolutions couldn’t quite render serifs cleanly enough—serif type was blurry or pixelated and generally looked worse than cleaner sans serif typefaces.

But with higher resolution screens, this isn’t as much an issue. It’s up to you and your needs.

However, there are some lesser-known CSS properties that you’ll want to take advantage of, especially for your headers. Pre-installed typefaces like Times New Roman, Arial, and Verdana are versatile but not perfect and can often benefit from tweaks at larger or very small sizes.

For example...

```
P {  
  WORD-SPACING: 0.25EM;  
}
```

```
H1 {  
  WORD-SPACING: -0.125EM;  
}
```

...will result in words in a paragraph that are proportionally spaced out slightly more than default, while it will pull words in top-level headings slightly closer together.

The same can also be done with the letter-spacing CSS property, but should only be applied to large headings or with special combinations of letters such as an upper-case T preceding a lower-case o.

And lastly there is the text-transform property, which should be used to define text that you always want to be uppercase, always lowercase, or always have their first letters capitalized.

References

WebTypography.net:

http://webtypography.net/Rhythm_and_Proportion/Horizontal_Motion/2.1.2/

http://webtypography.net/Rhythm_and_Proportion/Vertical_Motion/2.2.1/

http://webtypography.net/Harmony_and_Counterpoint/Size/3.1.1/

http://webtypography.net/Rhythm_and_Proportion/Horizontal_Motion/2.1.1/

http://webtypography.net/Rhythm_and_Proportion/Horizontal_Motion/2.1.8/

3. Exercise #1: Create perfect CSS statements

You and the person sitting next to you have a friend, Donna, who has asked you two to help design a blog for her existing website. Donna feels a little in over her head, and all she knows is that the font needs to be Verdana—the same as the rest of her site—and that she’s worried no one will read her entire posts because she tends to write very long posts...especially her father, who says he tends to skim other blogs because he thinks the type is always too small.

Your task is to reassure Donna by filling out these CSS statements so that they will beautifully render the typography for her blog’s main text block. (Assume it is just the text block, with no sidebar or other `div`s.) How would you explain to Donna your decisions? Without knowing anything else about Donna’s website, what might you look for to make sure your blog typography fits with the site as a whole?*

```
/* THE MAIN TEXT DIV */
#MAIN-TEXT {
}

/* THE BLOG POST'S HEADLINE */
H2 {
}

/* THE TEXT OF THE BLOG POST ITSELF */
P {
  FONT-FAMILY: VERDANA, SANS-SERIF;
}
```

*If you’re not comfortable enough with writing CSS from scratch, simply describe your thinking about what you would do with the main text block, heading, and paragraph text.

4. Choosing the right typeface for the job

This is a depressingly short section. Why? Because despite the web's having been around for two decades, we as website designers are still (practically speaking) limited to seven “web-safe” typefaces supported by the majority of web browsers:

- Arial
- Comic Sans
- Courier New
- Georgia
- Times New Roman
- Trebuchet
- Verdana

You'd better have an ironic reason to ever use Comic Sans, which takes us down to six options.

It goes without saying that your typeface should reflect the style of your website. Trebuchet and Verdana might be good for more modern sites, while Georgia is good for text meant to look more conservative.

The key thing, though, is to collect some data on your users' screen resolutions and colors. Serif fonts can be a strain on users' eyes whose monitors can't smoothly render small details. So if your website serves a large part of the general population—or, conversely, a large number of mobile users—consider sticking exclusively to sans serif fonts like Arial and Verdana for your body text.

Ironically, even as the average user's monitor increases in resolution and number of colors it supports, the trend is also for users to have larger screens that display type even smaller, another argument in favor of sans serif typefaces (as well as larger default text sizes).

Keep in mind the concept of “graceful degradation”—meaning a website works even when a user doesn't have everything needed

for it to work as ideally intended (for example, when Javascript is turned off). Graceful degradation for fonts means understanding that different browsers handle different fonts differently, and thus you need some amount of control over that handling. The very basic example is the font-family statement:

So instead of:

```
h1 {  
  font: helvetica;  
}
```

Use this instead:

```
h1 {  
  font-family: helvetica, arial, sans-serif;  
}
```

That way, if the user doesn't have Helvetica, the browser will render the text as an alternative sans serif font, Arial. And if the user then also doesn't have Arial, it will use any available sans serif on the user's system. The text degrades, but gracefully.

So in short:

- Stick to the five typefaces that are common enough not to cause problems
- Keep Arial and Verdana in your head as your default for body text
- And write your font-family statements so they degrade gracefully

Later, we'll touch upon reasons and methods for using typefaces that aren't one of those five—largely for headings.

References

<https://www.google.com/analytics/reporting/resolutions>

http://www.w3schools.com/browsers/browsers_display.asp

<http://www.theinternetdigest.net/archive/websafefonts.html>

5. Exercise #2: Choose the right font, according to your readers

Donna loves what you did for her blog (see exercise #1). However, even though you went ahead with her request to use Verdana for the body text and even though her father says he can read it fine, Donna comes back to you with an issue:

She says that one of her posts got linked to by a few major websites, and she's been getting tons of emails from random readers essentially saying, "I love your content. I want to keep reading it, but, at least on my screen, it's really hard to read."

With your partner, use everything you now know to discuss how you might:

1. Respond to these readers to probe their concerns
2. Use new CSS to address their concerns—without affecting Donna's father's reading experience

6. Design with accessibility in mind

Accessibility means being able to say, “Yes, everyone can access my content.” It not only has to do with users with visual disabilities but also with users whose needs we touched on above: those with small or huge screens, those who prefer a certain text size, etc.

Text size

Legibility: Your typography at the very least should be legible. That means body text that is generously-sized, line-height that strikes a balance between the length of the line (horizontal) and the length of the text (vertical), and a font-choice that has a bias towards a sans serif.

Accommodating resizing: Some users will want to resize your text, usually to make it larger. It’s as simple as hitting Control+ (or Command+ on a Mac). If you haven’t built your text blocks to handle resizing—if your text overruns, overlaps, or does nothing at all—you’re not doing right by your user.

So as we discussed above:

1. Set min and max widths for your divs, and try to set a comfortable measure, around 66 characters per line
2. Set your text styles to relative measures like ems or %
3. In your headers, pay attention to lesser-used CSS elements such as word-spacing in order to space things comfortably

Accommodating hard-copy printing: But don’t forget to create a separate print style sheet, one that, in contrast to your master style sheet, strictly sets the sizes for your text. When you create print.css, every text element should be sized in points (pt).

Color

You want to keep two things in mind when choosing a color for your typeface.

First, always have sufficient contrast. Black type on a white

background is as contrasty as you can get, but while there's no magic formula for measuring good contrast among the thousands of colors available, you can always print out your website (print the screen image, that is) on any black and white printer and see if it's comfortable to read. W3.org suggests it this way:

To test whether color contrast is sufficient to be read by people with color deficiencies or by those with low resolution monitors, print pages on a black and white printer (with backgrounds and colors appearing in grayscale). Also try taking the printout and copying it for two or three generations to see how it degrades. This will show you where you need to add redundant cues (example: hyperlinks are usually underlined on Web pages), or whether the cues are too small or indistinct to hold up well.

Second, remember that all colors on a computer screen are being radiated. In print, black text absorbs light while the water paper reflects it, but on a screen, every color is shooting photons at the user. And these colors are being radiated differently depending on the monitor. So a red background can not only be intense on the eyes but it might show up as orange or pink to some users depending on how their monitor is calibrated. Therefore, remember that your dark gray text on a light blue background ultimately might not have enough contrast when it comes out of a monitor rendering them as medium gray text on a medium blue background.

So, again, concentrate on the user's needs—you can even go as far as to make a second stylesheet that the user can opt for, for example if you sense a 50/50 split among your users' preferences for light text on a dark background vs. the opposite.

Colorblindness: Around 8% of the male population suffers from some level of color-blindness. (Through a genetic quirk, colorblindness is significantly lower among women). So when you choose colors for your type and background, consider using a tool like Adobe's Color Scheme Designer, which has a feature to help you see what a colorblind user sees. While there's no magic formula for what colors work best, you always want to know what your site would look like to someone who isn't seeing it as intended.

Fluid width

We've talked this one to death, but to reiterate one last time: different users have different screen sizes and different preferences for width. For your main blocks of text, always set a max and a min width, either in ems or %.

SIFr

One option I haven't spoken about is SIFr (and similar options like FLIR and Typeface.js). They're beyond our scope, but here's the background for SIFr and when you might use it.

SIFr stands for Scalable Inman Flash Replacement (Shaun Inman being the guy who came up with it), and it does what it says: it replaces each letter in a block of text with a Flash equivalent, using any typeface of the designer's choosing. So if I have a heading that reads "Typography for the Web" but want it to be displayed with the typeface *Officina*, SIFr will use Flash to overlay the letters T, y, p, o and so on, in *Officina*, over top of those same letters in a traditional web-friendly typeface.

The benefits are twofold: you can use *any* typeface, and screenreaders (for blind users) and search engines can still read the content, by reading the original text behind the Flash version.

The drawbacks are twofold as well: it can be difficult to implement, and because you're using Flash or a script, it's really only appropriate for headings—it would slow down the loading of body text to a point where the site is unpleasant to use.

Licensing

Lastly, I want to mention the licensing of typefaces. When I asked a group of typographers what they want people to know about type on the web, every one of them said—yelled, really—"Everyone needs to know that typefaces aren't free!"

We're accustomed to thinking type is free only because it comes pre-installed on our computers. And we at MIT are culturally accustomed to sharing everything. But Microsoft and Apple have in fact paid a lot of money to have those half a dozen basic types on our computers.

Typefaces can be downloaded just like music files—legally and illegally. So if you ever stray away from those basic typefaces and want to take advantage of something like SIFr, be sure you have some money budgeted to buy a license or be willing to risk running afoul of the law.

References

<http://www.w3.org/TR/WCAG10-CSS-TECHS/>

<http://colorschemedesigner.com/>

http://en.wikipedia.org/wiki/Color_blindness

http://veerle.duoh.com/blog/comments/my_view_on_light_text_on_dark_background_vs_readability/

<http://www.mikeindustries.com/blog/sifr>

<http://www.fonts.com/>

7. Tools you can use

(See <http://fungibleconvictions.com/web-typography/> for URLs)

Huge lists of resources

- 50 Useful Design Tools For Beautiful Web Typography
- 100 Free High-Quality XHTML/CSS Templates

Pure typographical tools

- Matrix of fonts bundled with Mac and Windows operating systems
- Typechart: “Lets you flip through, preview and compare web typography while retrieving the CSS”
- PXtoEM
- CSSType
- Type matrix and code generator
- Typetester
- CSS Type Set

More general CSS tools

- Gridr Builderrr: Helps create versatile CSS for columns
- Grid Designer 2
- PSD2CSS: Converts a Photoshop layout into CSS—requires strict attention to its guidelines to work
- Layout Generators

Firefox plugins

- abcTajpu: Easy reference for special characters
- Web Developer: My all-time favorite tool, allows you live-tweak your CSS on your own site

Inspiration

- 12 Examples of Paragraph Typography
- Google: CSS Inspiration