


CSS text styling part 1

Contents [\[hide\]](#)

- 1 Introduction
- 2 Print to web?
- 3 font styles
 - 3.1 Choosing a font with font-family
 - 3.1.1 Web-safe fonts
 - 3.1.2 Font stacks
 - 3.1.3 Overriding body fonts
 - 3.2 Image replacement
 - 3.3 Web fonts
 - 3.4 Choose your units! sizing with font-size
 - 3.4.1 Setting a base font size
 - 3.4.2 Setting heading and body sizes
 - 3.5 Changing the details with font-weight, font-style and font-variant
 - 3.5.1 font-weight
 - 3.5.2 font-style
 - 3.5.3 font-variant
- 4 Summary

Introduction

The web is composed of a lot of different things, technical, social and structural, but at its heart is something we all understand — human language. The beauty of the web is that it allows human-readable text to be presented so that it transcends barriers such as disability and geography. Of course, as humans we have been working out ways to format text so that it is legible to others for many centuries, but in printed form.

Web design has a lot of parallels with print design, but it also has a lot of differences and a lot more limitations. In this article we will explore how to style text on the web using CSS. The onrunning example will be a short biography page about [Frank Zappa](#) .

Note: our earlier article, [typography on the web], lays a good foundation for this chapter by explaining a number of typographic concepts. Make sure you read it before going on to this one.

Print to web?

The main shock that print designers get when coming to the web is the lack of control. In print design, the designer has ultimate control over how their creations will look, and are safe in the knowledge that the space the work appears in will not change dimensions, number of colours available, or any other such property.

On the web however, you can't guarantee that your beautifully crafted design will look identical to you and your users. They might be using a different operating system, or a different device altogether, or they might be visually impaired, meaning that they might be using your site with a lot of magnification, or a custom stylesheet that changes your fonts and colours, or a screen reader, which reads the text out to them!

So, you need to accept this uncertainty, and move on. We'll look at ways of making your designs more flexible so they look better in widely varying viewing contexts later on in the course, but for now, we'll leave it at that.

font styles

CSS has a number of properties beginning with `font-`, which allow you to control many features of the text characters (or glyphs) themselves. Let's look at these now.

Choosing a font with font-family

`font-family` allows you to specify which font, or fonts, a selection of elements will use. Take the Zappa example linked to above and try adding the following CSS line to the existing `body` rule, then save and reload:

```
font-family: Arial;
```

This will apply Arial — a sans-serif font (no serifs, the little decorative features at the end of the strokes of the text) — to the whole document, rather than the default serif font (fonts that have got serifs), provided this font is available on user's systems.

It probably will be, because Arial is a ubiquitous font, which you can pretty much guarantee will be available on all systems.

Web-safe fonts

There are about 11 fonts that are installed across pretty much all systems, termed **web safe fonts** because they are safe for use in your pages. The full list is as follows:

- **Sans-serif:** fonts without serifs: Verdana, Arial, Trebuchet MS
- **Serif:** fonts with serifs: Times new roman, Georgia
- **Monospaced:** fonts in which every glyph takes up the same space, like in computer code: Andale mono, Courier new
- **Cursive:** fonts that have a decorative, often handwritten-looking style: Comic Sans
- **Fantasy:** fonts that have a bold, often ornamental or quirky style, which are meant to be used for headings, not body copy: Impact

Font stacks

If you want, you can also apply a number of fonts to a single element selection, known as a **font stack**. Try changing the line you added above to the following:

```
font-family: 'helvetica neue', arial, verdana, sans-serif;
```

You'll probably not see much difference from what was there before, so let's explain it. When you specify a font stack like this, the browser goes through them from left to right, until it finds a font that is installed on the system, and therefore can be used.

So:

- The browser searches for Helvetica neue on the user's system, and uses this if it finds it. This is a nicer looking font than the more common ones, so I want to use it if possible.

- If Helvetica neue isn't found, the browser searches for Arial on the user's system, and uses it if it finds it. This is still quite a nice sans-serif font.
- If Arial isn't found, we'll use Verdana, which is pretty ugly as fonts go, but better than nothing.
- As a last resort, if none of the fonts in the font stack are found we fall back to sans-serif, which basically instructs the browser to use whatever the system's default sans-serif font is. You don't know exactly what will be used in this eventuality, which is an annoying loss of control, but at least this is better than ending up with the browser default — Times new roman — which is a serif font!

Note that fonts with more than one word in their name need surrounding in quotes.

Overriding body fonts

So far we've set fonts on the `<body>` element, which sets a page-wide default that affects everything; now we'll look to override some of our `<body>`'s child elements to set things up nicely.

For a start, it is often a good idea to use a different font for at least some of your headings, to make them stand out and give your site a bit more character. Add the following to your styles:

```
h1, h2 {  
    font-family: impact;  
}
```

Save and reload and you'll see that the page now has a very different look.

As for other fonts we could affect, how about the emphasised text? I've used `` for all references involving dates to make them stand out a bit. As it is, the browser gives emphasis a default italic look, but this doesn't stand out as much as it could, so let's add the following to our styles:

```
em {  
    font-family: georgia;  
}
```

Your fonts (especially your headings) are one of the biggest contributors to the overall personality of your site, so choose them carefully to fit. This may sound difficult to begin with, but you'll soon get the hang of it. As a rule, Times new roman is good for corporate sites, whereas a serif like Arial might be better for a less formal site. Cursive fonts need to be used sparingly — for example comic sans is hated by many designers, but it is suitable sometimes, for example if you want to give a chalk on blackboard effect on a children's site.

Headings used to be difficult to deal with. Given that the only web safe fantasy font seems to be Impact (not bad, but imagine if all sites were stuck with it!) and you don't want to be stuck with a default system font such as the dreadful Papyrus, designers used to resort to using various image techniques for their headings. As you should know by now, using images for text on a web site is really bad because it can't be read by screen readers or search engines. In the next section we'll look at ways around this problem.

See [zappa2.html](#) for the additions so far.

Image replacement

Image replacement is a very common technique for having better looking headings and company logos, but not making the text inaccessible or compromising on search engine optimization. The way it works is roughly thus.

Create an image of how you want your heading to look. You can [use my Frank Zappa image](#). Now, try adding the following rule to your CSS, and save and reload, making sure my image is in the same directory as the zappa.html file:

```
h1 {  
  background-image: url(zappa.png);  
  height: 55px;  
  text-indent: -9000px;  
}
```

There are lots of variations on image replacement, but this is basically how most of them work: first of all, we include our image as a background image on the element we want to replace, setting the dimensions of the element to make sure the image can be fully seen (I've made my image 600 x 55 px - it's the same width as my content column.) Then we use a very large negative text indent to push the actual heading text right off the page so you can't see it, just leaving the background image in view. Don't worry about the unfamiliar CSS properties: you'll learn more about these later in the course.

This works pretty well — you get the heading look you want displayed on the page, across most browsers (not many browsers these days can't display background images), and the actual text is still there to provide screen readers and search engines with what they need. There are some disadvantages however:

- This technique is very inflexible. You have to create an image for each bit of text you want to replace, and then change them every time you want to change size, colour, wording, or anything else. And imagine trying to use image replacement on bits of text inside paragraphs as well as headings? Too fiddly.
- Text scales nicely as you zoom in, images don't. If you try zooming in the example too much, the heading will start to look grainy and horrible.
- If you try to use too much text replacement, the extra HTTP requests could slow things down a fair amount. Some more sophisticated image replacement techniques can slow down the page even more, as they use Flash ([siFR](#)) or SVG ([cufon](#)).

See [zappa3.html](#) for the additions at this point.

Web fonts

CSS3 introduces Web fonts, a feature that allows us to specify our own custom font files to download along with our web pages. This is great, as it complete gets around the problem of fonts not being available on user's machines. To specify a web font for download on a page, you reference the font in a special `@font-face` block that goes at the top of the page, and looks something like this:

```
@font-face {
```

```
font-family: 'My font';
src: url('myfont.ttf') format('truetype');
}
```

There are more available options, but let's keep it simple for now. You basically specify what you want your font family to be called, and then point to the font file you want to download along with your web page. This should always go at the top of your CSS file, so you can use it afterwards.

You then include the font in your page in exactly the same way as you would other fonts:

```
font-family: 'My font';
```

Simple huh? Well, not quite. the reality is that not all browsers support the same font formats, so the syntax to make this work across browsers is more complicated. Luckily, you'll not have to write it yourself, as there are a number of services available that will do the hard work for you. We'll look at a free service called [Font Squirrel](#), which not only has lot of great fonts available for you to use, but also generates all the CSS and font files you'll need.

Let's use this now.

- For a start, you need a font file. Any format just about will do to start with. The font I used for the Frank Zappa heading is called Romantiques, and I got it from <http://www.fontspace.com/category/circus>. Download the zip file, then unzip it.
- Now go to [fontsquirrel.com](#) and choose the [@font-face generator](#).
- Click "add fonts", select the Romantique font file you want to use, and check the disclaimer checkbox agreement.
- Click the "Download your kit" button, and after a few seconds you'll be prompted to save your web font kit. Save it in your example files.
- Unzip the kit, and you'll several files. the ones you are interested in are the different font format files, and the `stylesheet.css`.
- Copy all the font files (`.eot`, `.svg`, `.ttf` and `.woff`) to a sensible location in your example files. In the same directory as the zappa HTML file would be easiest for now.
- Open the `stylesheet.css` file and copy the CSS rule inside it into the top of your example file styles. It should look like so:

```
@font-face {
  font-family: 'RomantiquesRegular';
  src: url('romantiques-webfont.eot');
  src: url('romantiques-webfont.eot?#iefix') format('embedded-opentype'),
        url('romantiques-webfont.woff') format('woff'),
        url('romantiques-webfont.ttf') format('truetype'),
        url('romantiques-webfont.svg#RomantiquesRegular') format('svg');
  font-weight: normal;
  font-style: normal;
}
```

1. Again, the `font-family` dictates what we need to call the font in the code.
2. The `src` values specify the locations of the font files. Different browsers coming

across this keep going through the list until they find a format they understand, at which point they download that font file and use it in the page (although some browsers are a bit buggy, and may download more than just the one they need, wasting bandwidth in the process). IE uses the `.eot` version; most modern browsers will use the `.woff`, which is smaller in file size than the others; older browsers that aren't IE and don't support `.woff` will use the `.ttf` or `.svg` files.

3. the `font-weight` and `font-style` specify things like the weight of the font, and if it is italic, but don't worry about those for now.

Note: if you didn't put the font files in the same directory as your example file, you'll have to update the file paths to suit.

- Delete the `h1` rule we added previously, and replace it with this:

```
h1, h2 {
  font-family: RomantiquesRegular;
}
```

Now save and reload, and you'll see the font applied to all first and second level headings in the page. Very useful, and much more flexible than image replacement! Try adding a color to your headings:

```
color: #530FAD;
```

There are some issues with using custom fonts in this manner, to be aware of:

- **File size:** I deliberately chose this font for the example to highlight file size problems! Most western font files aren't this big, but the Romantique files were mostly a few hundred KB, with the SVG weighing in at almost 1MB! This is a lot of extra weight to add to a page, especially for those downloading on slower bandwidth such as mobile networks. And while western glyph sets usually only have tens of characters in them, maybe a few hundred if a set is really complete, some language font sets (especially `<abbr title="Chinese Japanese Korean">CJK`) can have literally thousands of characters, and be many MB in size. Be aware!
- **Font size:** You'll notice that the font size is a lot smaller than with the image replacement version. Different base font sizes can also vary dramatically, so you need to think about this when setting font sizes, and thinking about how fallback font sizes will match up to the primary fonts you want to use.
- **FOUT:** This is when the page is loading, and for a short while you see the text without the web font applied, before the web font finishes loading. When it does load, the page shifts to show the new font, which can be a bit of a jarring effect for the user. You can mitigate this using a library such as [Google web font loader http://code.google.com/apis/webfonts/docs/webfont_loader.html] or FOUT-b-Gone
- **Number of glyphs/font quality:** there are a lot of free fonts available, from sites like [Font squirrel](#), [DaFont](#), [My Fonts](#), and [Google web fonts](#), however they are not all high quality, and some may have a very limited glyph set, so you'll get horrible little blank squares displayed instead of characters if you have characters on a page that aren't present in your chosen font. This presents far more of a problem in body text than

headings, and you need to be especially careful of this if you are using web fonts on sites with user generated content, as you can't control exactly what characters they'll use. Some fonts may also look bad on certain operating systems (they can look bad on Windows if the user doesn't have cleartype enabled). The only answer here is to test your fonts thoroughly before using them in your design. For example, the comma in the first<he> looked terrible after I set my fonts to Romantique, so I removed it.

Note: There are professional paid font services you can use for your web font needs, such as [Fontdeck](#) and [Typekit](#). If you have the money to spend, their font options will give you higher quality.

I also set my emphasis font so something a bit more interesting, a "slab serif" font called Copse, to make things a bit more interesting.

See [zappa4.html](#) for the additions at this point.

Choose your units! sizing with font-size

The next CSS property we'll look at is `font-size`. This allows you to set the size of the text inside selected elements, using any CSS units available, such as pixels, ems, percentage, etc.

You can also use size keywords: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, and `xx-large`. These tend to be best used when you want to set font sizes relatively to each other, so for example the body size is `medium`, then you could set the top level heading to `xx-large`, second level heading to `x-large`, perhaps.

Then they will always stay proportionate to one another, regardless of what fonts are used on what operating system. But they don't offer a great amount of control — you don't see them used that often. Let's look at a typical font-sizing exercise.

Setting a base font size

The first thing you'll want to do is set a base font size for the whole document. This is usually done on the `<body>` element, so add the following to your `body` rule:

```
font-size: 62.5%
```

Why 62.5%? The answer is that the default font-size for most browsers is 16px. 62.5% of 16 is 10, so by doing this we are setting the base font for the whole site to 10px, which makes subsequent maths easier.

Setting heading and body sizes

Next, let's turn our attention to the font-sizes for our different content. Now the base font size is 10px, we can easily work out the sizes we want the rest of our text to be, for example if we want our first level headings to be 42px, we can set the font-size to 420%, or 4.2ems. Percentages and ems work pretty much equivalently for our purposes. The differences aren't really that important, at least for now. They both set sizes proportionately to the size of parent font size, and the added bonus is that you can also set other dimensions in your design to percentages of ems, eg. margins, widths of content blocks, etc. It is nice to be able to control a whole site relative to the text sizes.

The other option you'll come across often is using pixels. These are seen to give more

control, as you set an absolute pixel size, that doesn't change proportionally to anything else. But this is not the best idea, as older browsers like IE6 can't resize text set in pixels, which is a big accessibility problem.

We'll stick to ems for now.

First of all, add the following rules to your styles:

```
h1 {  
    font-size: 5.5em;  
}  
  
h2 {  
    font-size: 3em;  
}
```

If you try this in different browsers, you'll notice that this rather complex font looks better in some than others, and none look quite as good as the image replacement version, with the heading created in Adobe Fireworks. This is the kind of testing you should do before embarking too far into a project with a chosen font. Also, it doesn't look that great at a smaller size on the <h2>s. But we'll leave as is for the purposes of this example.

Next, let's set the body text and list to something a bit more readable:

```
p, ul {  
    font-size: 1.6em;  
}
```

Here we've actually just set the body copy back to the default browser size, but it's good and readable.

We'll also set the emphasis slightly smaller, to make it stand out even more nicely:


```
em {  
    font-size: 0.8em;  
}
```

You might be wondering why we had to set it to 0.8em to get it slightly smaller, rather than say 1.5 or 1.4? Well, try it and see!

The reason is that the parent elements of (either <p> or) have been set to 1.5em. The sizing is relative to the font size of the immediate parent, so in this context 16px is 1em. We have set 0.8em for the emphasised text, so the size comes out as 16px x 0.8 = about 13px.

Note: the pixel sizes I am talking about here are what is called the **computed values**.

Regardless of what units and values you use to set your font sizes, computers display graphics in pixels, so the browser must convert everything into pixel values before it can display them to your users.

See [zappa5.html](#)  for the additions at this point.

Changing the details with font-weight, font-style and font-variant

The next properties we'll look at are as follows:

- `font-weight` allows you to set the boldness of text in selected elements.
- `font-style` allows you to set an element's text to be oblique or italic.
- `font-variant` allows you to set an element's text to be small-caps, also known as copperplate letters.

font-weight

This property is what you need if you want to set text to be bolder. Possible values are:

- `bold` and `bolder` provide two levels of extra boldness.
- `lighter` provides a level of less boldness.
- `100`, `200` ... all the way up to `900` provide incremental levels of boldness.
- `normal` is the default non-bold setting.

Most of these don't really do anything, especially not at small sizes, and when the font you are using doesn't have different levels of boldness defined in it.

For this example, let's add the following to highlight the first lines below each heading:

```
h1 + p:first-line, h2 + p:first-line {  
  font-weight: bold;  
}
```

save, reload and check it out.

Note: `bold` is equivalent to `600`.

font-style

`font-style` can take the values `italic`, `oblique`, and `normal`. Normal is the default, as before, `italic` tells the browser to use the italic version of the font (if available), and `oblique` tells the browser to use the normal version of the font, but slant it. If you specify `italic`, unless the font you are using has a specific italic version available the browser will fall back to generating an oblique version and using that.


Let's add the following line to the rule we added in the previous section:

```
font-style: italic;
```

font-variant

`font-variant` can take two values, `normal`, which is the default as you'd expect, and `small-caps`, which uses all capital characters, but large ones for the capital letters, and small ones for the lower case letters. Add the following to the `em` rule and see what effect it gives:

```
font-variant: small-caps;
```

See [zappa6.html](#)  for the additions at this point.

Summary

Here we've gone through the fundamentals of styling text on the web. But text styling doesn't stop there. In the next article we will cover the rest of the CSS features associated with styling text, including advanced features and new things appearing in CSS3.