Web fonts

- Font families recap
- Web fonts
- Adding your own web fonts
  - Finding fonts
  - Generating the required code
  - Implementing the code in your demo
- Using an online font service
- @font-face in more detail
- Summary
- See also

In the first article of the module, we explored the basic CSS features available for styling fonts and text. In this article, we will go further, exploring web fonts in detail. We'll see how to use custom fonts with your web page to allow for more varied, custom text styling.

Prerequisites:	Structuring content with HTML, CSS Styling basics, Fundamental text and font styling.
Objective:	• Understand that web fonts allow developers to go beyond the web safe font set and use custom fonts on their web apps. • Basic setup — the `@font-face` at-rule, and common descriptors. • Using a web font with the `font-family` property. • Using online services to find web fonts and generate web font code.

Font families recap

As we looked at in Fundamental text and font styling, the fonts applied to your HTML can be controlled using the font-family property. This takes one or more font family names. When

displaying a webpage, a browser will travel down a list of font-family values until it finds a font available on the system it is running on:

```
p {
  font-family: "Helvetica", "Trebuchet MS", "Verdana", sans-serif;
}
```

This system works well, but traditionally web developers' font choices were limited. There are only a handful of fonts that you can guarantee to be available across all common systems — the so-called Web-safe fonts. You can use the font stack to specify preferred fonts, followed by web-safe alternatives, followed by the default system font. However, this increases your workload because of the testing required to make sure that your designs work with each font.

Web fonts

There is an alternative that works well. CSS allows you to specify font files, available on the web, to be downloaded along with your website as it's accessed. This means that any browser supporting this CSS feature can display the fonts you've specifically chosen. Amazing! The syntax required looks something like this:

First of all, you have a @font-face ruleset at the start of the CSS, which specifies the font file(s) to download:

```
@font-face {
  font-family: "myFont";
  src: url("myFont.woff2");
}
```

Below this, you use the font family name specified inside @font-face to apply your custom font to anything you like, as normal:

```
html {
  font-family: "myFont", "Bitstream Vera Serif", serif;
}
```

The syntax does get a bit more complex than this. We'll go into more detail below.

Here are some important things to bear in mind about web fonts:

1. Fonts generally aren't free to use without restrictions. You have to pay for them and/or follow other license conditions, such as crediting the font creator in your code (or on your

site). You shouldn't steal fonts and use them without giving proper credit.
2. All modern browsers support WOFF2 (Web Open Font Format version 2), so it is the format we'll use in this article.
3. WOFF2 supports the entirety of the TrueType and OpenType specifications, including variable fonts, chromatic fonts, and font collections.
4. It is a good idea to provide as few web font files as possible. Using only the formats you need reduces the number of files your site has to serve and helps keep font downloads smaller.
5. If you need to support older browsers in a real-world project, you may also need to provide WOFF. Check the @font-face browser compatibility table to see how far back WOFF and WOFF2 support goes.

You can use the Firefox Font Editor to investigate and manipulate the fonts in use on your page, whether they are web fonts or not.

Adding your own web fonts

With this in mind, let's build up a basic web font example from first principles. You should use the web-font-start.html and web-font-start.css files as a starting point to add your code to (see the live example). Make a copy of these files in a new directory on your computer now. In the `web-font-start.css` file, you'll find some minimal CSS to deal with the basic layout and typesetting of the example.

Finding fonts

For this example, we'll use two web fonts: one for the headings and one for the body text. To start with, we need to find the font files that contain the fonts. Fonts are created by font foundries and are stored in different file formats. There are generally three types of sites where you can obtain fonts:

- A free font distributor: This is a site that makes free fonts available for download (there may still be some license conditions, such as crediting the font creator). Examples include DaFont and Everything Fonts.
- A paid font distributor: This is a site that makes fonts available for a charge, such as myfonts.com. You can also buy fonts directly from font foundries, for example Linotype, Monotype, or Exljbris.
- An online font service: This is a site that stores and serves the fonts for you, making the whole process easier. See the Using an online font service section for more details.

Let's find some fonts! Go to DaFont and choose two fonts: a nice interesting font for the headings (maybe a nice display or slab serif font), and a slightly less flashy and more readable font for the paragraphs. When you've found a font, press the download button and save the file inside the same directory as the HTML and CSS files you saved earlier. It doesn't matter whether they are TrueType Fonts (TTF) or OpenType Fonts (OTF).

Unzip the two font packages (Web fonts are usually distributed in ZIP files containing the font file(s) and licensing information). You may find multiple font files in the package — some fonts are distributed as a family with different variants available — for example, thin, medium, bold, italic, thin italic, etc. For this example, we just want you to concern yourself with a single font file for each choice.

Generating the required code

Now you'll need to generate the required code (and font formats). Follow these steps:

1. Make sure you have satisfied any licensing requirements if you are going to use this in a commercial and/or Web project.
2. Go to the Transfonter webfont generator.
3. Upload your two font files using the *Upload your fonts* button.
4. Make sure only the **WOFF2** checkbox is checked; uncheck **WOFF**.
5. Click *Convert*.
6. Click *Download*.

After the ZIP file is downloaded, unzip it and move it into the same directory as your HTML and CSS.

Implementing the code in your demo

Inside the unzipped directory, you'll see some useful items:

- A `.woff2` version of each font.
- A demo HTML file for each font — load these in your browser to see what the font will look like in different usage contexts.
- A `stylesheet.css` file, which contains the generated @font-face code you'll need.

To implement these fonts in your demo, follow these steps:

1. Rename the unzipped directory to something easy and simple, like `fonts`.
2. Open up the `stylesheet.css` file and copy the two `@font-face` rulesets into your `web-font-start.css` file — you need to put them at the very top, before any of your CSS, as the fonts need to be imported before you can use them on your site.

3. Each of the `url()` functions points to a font file that we want to import into our CSS. We need to make sure the paths to the files are correct, so add `fonts/` to the start of each path (adjust as necessary).
4. Now you can use these fonts in your font stacks, just like any web-safe or default system font. For example:

   ```
   @font-face {
     font-family: "zantrokeregular";
     src: url("fonts/zantroke-webfont.woff2") format("woff2");
     font-weight: normal;
     font-style: normal;
     font-display: swap;
   }
   ```

   ```
   font-family: "zantrokeregular", serif;
   ```

   The generic `serif` family provides a system-font fallback if the web font cannot be loaded. Keeping the number of web font files and formats to a minimum also helps performance by reducing the resources the browser needs to fetch.

You should end up with a demo page with some nice fonts. Because different fonts are created at different sizes, you may have to adjust the size, spacing, etc., to improve the look and feel.

The finished design of a Web font exercise. The page has two headings and three paragraphs. The p

> **NOTE:** If you have any problems getting this to work, feel free to compare your version to our finished files — see web-font-finished.html and web-font-finished.css. You can also download the code from GitHub or run the finished example live.

Using an online font service

Online font services generally store and serve fonts for you, so you don't have to worry about writing the `@font-face` code. Instead, you generally just need to insert a simple line or two of code into your site to make everything work. Examples include Adobe Fonts and Cloud.typography. Most of these services are subscription-based, with the notable exception of Google Fonts, a useful free service, especially for rapid testing work and writing demos.

Most of these services are easy to use. Let's have a quick look at Google Fonts so you can get the idea. Again, use copies of `web-font-start.html` and `web-font-start.css` as your starting point.

1. Go to Google Fonts.
2. Find a couple of fonts you like using the filters and search bar.
3. Click on a font to open its details page.
4. When you find a font you like, click on the **Get font** button on its details page to add it to the selected fonts page. If you wish to add another font, click your browser's back button and search again.
5. Once you are done selecting fonts, click the **Get embed code** button on the selected fonts page and copy the provided `<link>` elements.
6. Paste the `<link>` elements into the `<head>` of your HTML document, above any existing stylesheet links.
7. Copy the provided `font-family` CSS rules and use them in your CSS to apply the fonts, in a similar way to the previous walkthrough.

> **NOTE:** You can find a completed version at google-font.html and google-font.css, if you need to check your work against ours (see it live).

@font-face in more detail

Let's explore that `@font-face` syntax generated for you by Transfonter. The rulesets will look something like this:

```
@font-face {
  font-family: "zantrokeregular";
  src: url("zantroke-webfont.woff2") format("woff2");
  font-weight: normal;
  font-style: normal;
  font-display: swap;
}
```

Let's go through it to see what it does:

- `font-family`: This line specifies the name you want to refer to the font as. This can be anything you like as long as you use it consistently throughout your CSS.
- `src`: This line specifies the path to the font file to be imported into your CSS (the `url` part), and the font format (the `format` part). The latter part is optional, but it is useful to declare because it allows browsers to determine whether they can use the font more

quickly. If you need to support older browsers, you can list additional font sources separated by commas, with your preferred format (usually WOFF2) first.

- font-weight/font-style: These lines specify what weight the font has and whether it is italic or not. If you are importing multiple weights of the same font, you can specify what their weight/style is and then use different values of `font-weight` / `font-style` to choose between them, rather than having to give all the members of the font family different names. @font-face tip: define font-weight and font-style to keep your CSS simple by Roger Johansson shows what to do in more detail.
- font-display: This line specifies how the font is displayed while it is loading.

> **NOTE:** You can also specify particular font-variation-settings and font-stretch values for your web fonts. In newer browsers, you can also specify a unicode-range value, which is a specific range of characters you might want to use out of the web font. In supporting browsers, the font will only be downloaded if the page contains those specified characters, saving unnecessary downloads. Creating Custom Font Stacks with Unicode-Range by Drew McLellan provides some useful ideas on how to make use of this.

Summary

Now that you have worked through our articles on text styling fundamentals, it's time to test your comprehension with our challenge for the module: Typesetting a community school homepage.

Once you're done with the challenge, you can move on to learning about CSS layout.

See also

- Variable fonts guide
- Fonts knowledge, Google Fonts

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:33:01 UTC