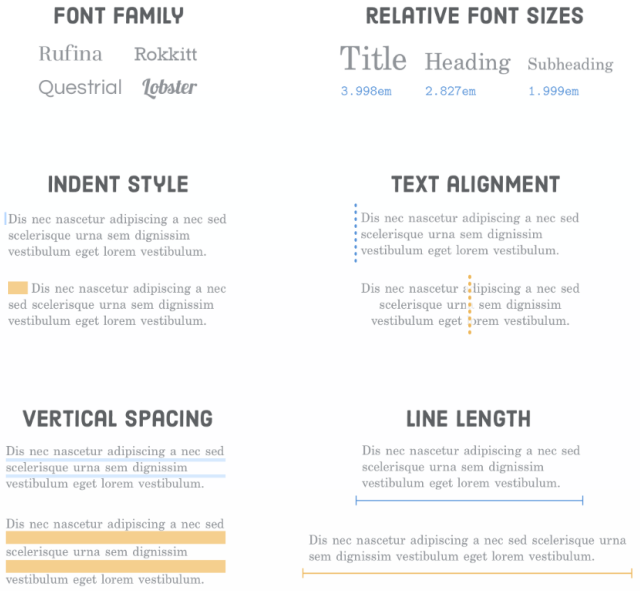
WEB TYPOGRAPHY

Nº 14. OF HTML & CSS IS HARD

A friendly tutorial about web fonts and basic typographic principles

"Web typography" refers to the appearance of all the text on your website. It includes basic CSS text properties like what font to use and whether it should be italic or not, but typography is much more than that. It's about the space between and around letters, words, and lines. It's the size of different runs of text in relation to one another, and the history behind each font family.



A lot of your typography decisions will come from a designer. The only problem is that typography is an invisible art. To actually understand what your designer is asking for, you need to be able to *see* typography the same way they do.

This chapter isn't just about the mechanics of adding web fonts to your site or the CSS properties to move your text around. We'll also explain how to properly leverage all these tools to make beautiful, professional websites. By the end of the chapter, you should not only know what your designer is talking about when they say something like, "Can we increase the leading of that paragraph?", but also understand *why* they want you to increase it.

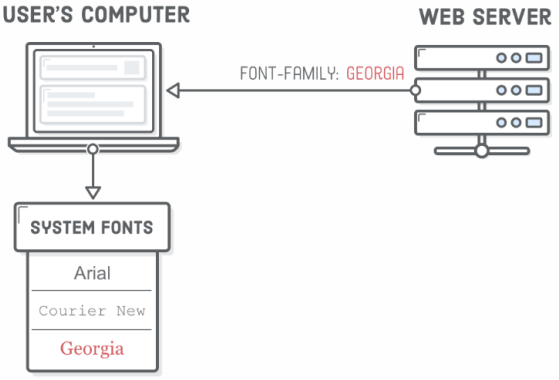
You may forget the specific CSS properties, but the typographic concepts we're going to cover will stay with you for the rest of your life because they aren't arbitrary rules—they're grounded in function. They make your content more readable and help you communicate your message more effectively.

A BRIEF HISTORY OF WEB FONTS

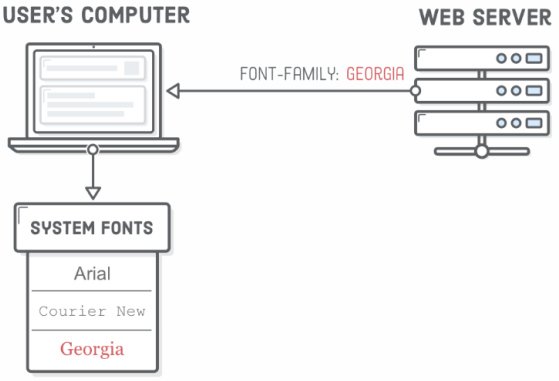
We're going to start this chapter by learning how to display your web pages in a custom font because that's the most exciting aspect of modern web typography. However, web fonts have changed a lot over the last few years, so before we can start building out our example, we need a little primer on the various font formats floating around the Internet.

WEB SAFE FONTS

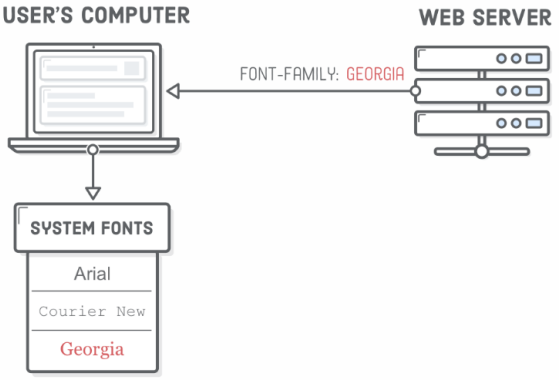
Long, long ago, web developers only had "web safe fonts" at their disposal. These were a collection of a dozen or so fonts that were pre-installed on most computers. There was no such thing as a custom font file that you could send to browsers to use on your website.



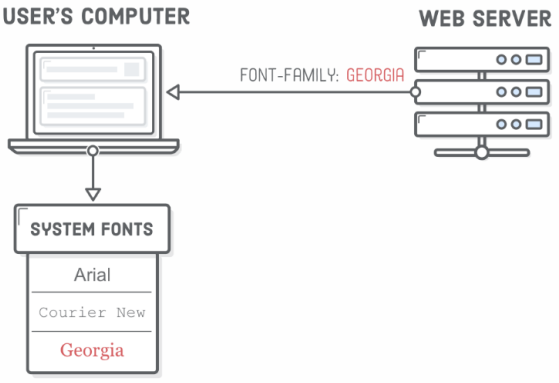
If you *needed* a special font, your only option was to export an image of the text you wanted to display and include it in your web page with an `<img/>` element. This was ridiculously limiting for web designers and resulted in



If you *needed* a special font, your only option was to export an image of the text you wanted to display and include it in your web page with an `<img/>` element. This was ridiculously limiting for web designers and resulted in some pretty hacky situations for developers. Honestly, we don't know how everybody survived through that era of HTML and CSS.



If you *needed* a special font, your only option was to export an image of the text you wanted to display and include it in your web page with an `<img/>` element. This was ridiculously limiting for web designers and resulted in some pretty hacky situations for developers. Honestly, we don't know how everybody survived through that era of HTML and CSS.



If you *needed* a special font, your only option was to export an image of the text you wanted to display and include it in your web page with an `<img/>` element. This was ridiculously limiting for web designers and resulted in some pretty hacky situations for developers. Honestly, we don't know how everybody survived through that era of HTML and CSS.

CUSTOM WEB FONTS

Around 2010, browsers began supporting custom web fonts, which was great, except for the fact that each browser and device required a different file format. Accordingly, most websites provided 4 different web font files:

FILE FORMAT	BROWSER/DEVICE
`.svg`	Very old Safari (iOS and Desktop)
`.eot`	Internet Explorer
`.ttf`	Everything except Internet Explorer
`.woff`	Newer browsers

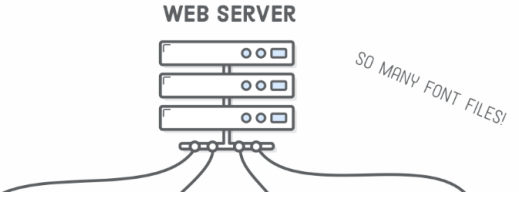
This resulted in the "Bulletproof `@font-face` syntax", which you'll likely encounter at some point in your web development career.

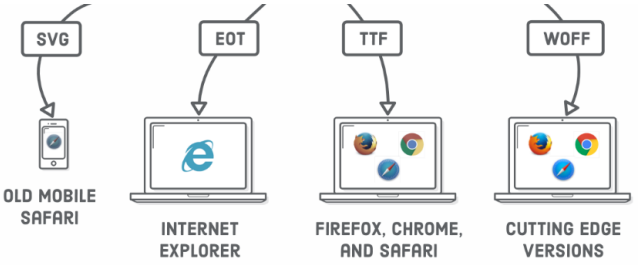
CUSTOM WEB FONTS

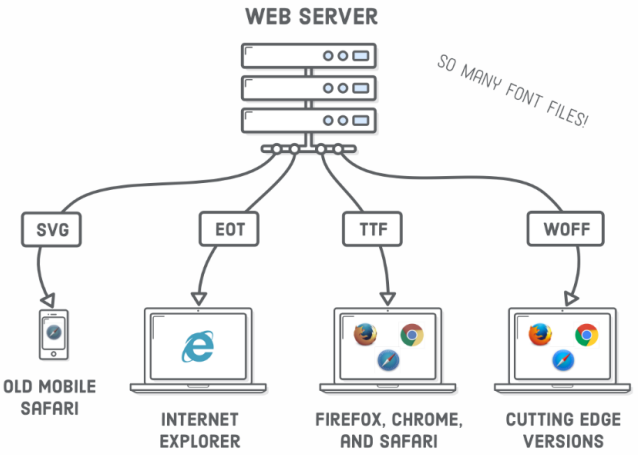
Around 2010, browsers began supporting custom web fonts, which was great, except for the fact that each browser and device required a different file format. Accordingly, most websites provided 4 different web font files:

FILE FORMAT	BROWSER/DEVICE
`.svg`	Very old Safari (iOS and Desktop)
`.eot`	Internet Explorer
`.ttf`	Everything except Internet Explorer
`.woff`	Newer browsers

This resulted in the "Bulletproof `@font-face` syntax", which you'll likely encounter at some point in your web development career.

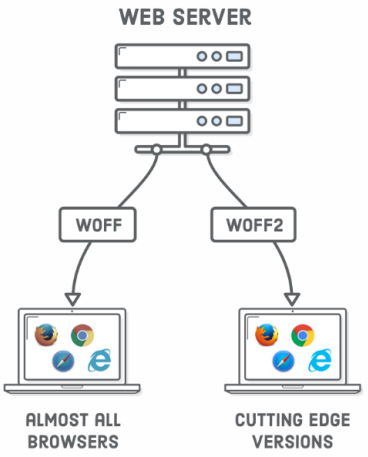




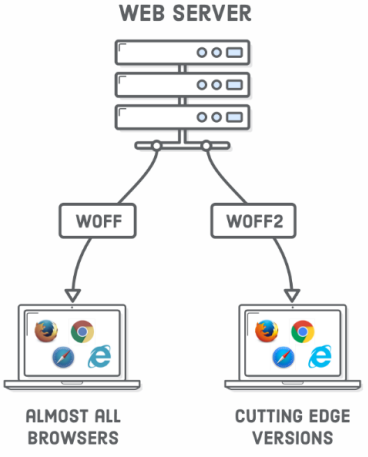


WOFF FONTS

Recently, the industry has standardized on the Web Open Font Format (WOFF), so things have gotten a little bit simpler for us. Over 90% of modern browsers support `.woff` fonts, and support for its next evolution, `.woff2`, is growing. WOFF2 is similar to the original WOFF format, but offers a significant reduction in file size (which means better performance).



Eventually, you'll only need to support WOFF2, but right now, we suggest



Eventually, you'll only need to support WOFF2, but right now, we suggest providing both WOFF and WOFF2 web fonts to get decent coverage for older browsers and improved performance on modern ones. Unless legacy browsers make up a large chunk of your target audience, `.ttf`, `.svg`, and `.eot` fonts are a thing of the past.

WHERE TO FIND WEB FONTS

There's a ton of places on the web where you can download both free and premium web fonts, but our three favorites are listed below. Again, which font to use is usually up to your designer (and their budget), but as a developer, it's still good to know the trade-offs between these options.

WEBSITE	PRICE	QUALITY	SELECTION
Font Squirrel	Free	Hit-or-Miss	Huge
Google Fonts	Free	Good	Decent
Fontspring	Expensive	Excellent	Huge

Note that Font Squirrel and Fontspring offer both web fonts and desktop fonts (`.otf` and `.ttf` files). WOFF is designed specifically for the needs of the modern web, while desktop fonts contain extra functionality useful for graphics editing programs like Adobe Illustrator. Be sure to download or purchase the web font version of the fonts you want to use—not just the desktop version.

WEBSITE	PRICE	QUALITY	SELECTION
Font Squirrel	Free	Hit-or-Miss	Huge
Google Fonts	Free	Good	Decent
Fontspring	Expensive	Excellent	Huge

Note that Font Squirrel and Fontspring offer both web fonts and desktop fonts (`.otf` and `.ttf` files). WOFF is designed specifically for the needs of the modern web, while desktop fonts contain extra functionality useful for graphics editing programs like Adobe Illustrator. Be sure to download or purchase the web font version of the fonts you want to use—not just the desktop version.

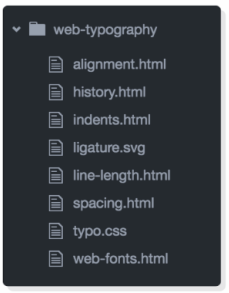
SETUP

Ok! We're ready to experiment with web fonts. We're going to be building this example website. We figured you probably don't want to start this one from scratch, so go ahead and download the initial project. Unzip it and open up the `web-typography` folder with your favorite text editor. If you don't have a favorite text editor, you might want to check out Atom.

graphics editing programs like Adobe Illustrator. Be sure to download or purchase the web font version of the fonts you want to use—not just the desktop version.

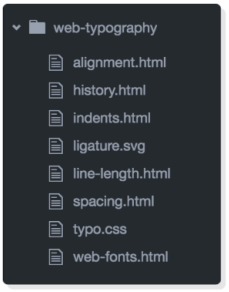
SETUP

Ok! We're ready to experiment with web fonts. We're going to be building this example website. We figured you probably don't want to start this one from scratch, so go ahead and download the initial project. Unzip it and open up the `web-typography` folder with your favorite text editor. If you don't have a favorite text editor, you might want to check out Atom.

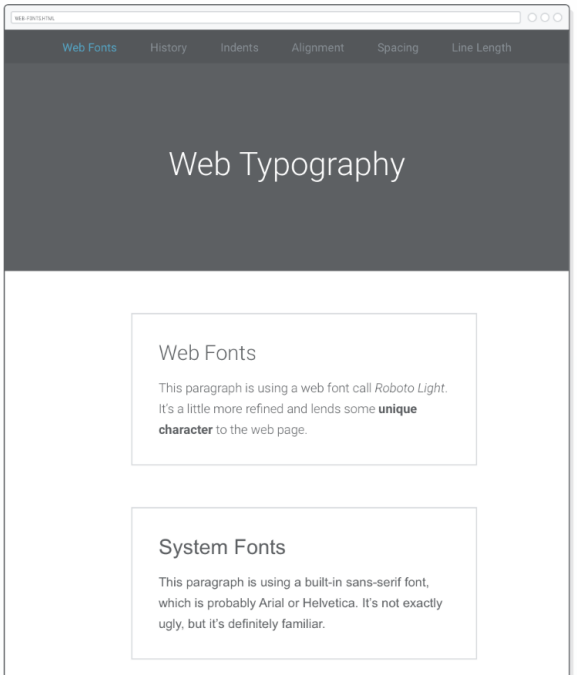


We've got 6 HTML documents all using the same `typo.css` stylesheet. We'll be demonstrating various typographic principles by adding some page-

this example website. We figured you probably don't want to start this one from scratch, so go ahead and download the initial project. Unzip it and open up the `web-typography` folder with your favorite text editor. If you don't have a favorite text editor, you might want to check out Atom.



We've got 6 HTML documents all using the same `typo.css` stylesheet. We'll be demonstrating various typographic principles by adding some page-specific styles to each of these HTML files.



Open up one of the HTML files with a web browser, and you'll find that our initial project is pretty close to the final example, minus all the web fonts and other CSS typography properties.

LOCALLY HOSTED WEB FONTS

There are two distinct methods of adding web fonts to your website: locally hosted or externally hosted. We'll take a look at both in this chapter. First, we'll be adding a locally hosted web font to our example project. This is a three-step process:

Open up one of the HTML files with a web browser, and you'll find that our initial project is pretty close to the final example, minus all the web fonts and other CSS typography properties.

LOCALLY HOSTED WEB FONTS

There are two distinct methods of adding web fonts to your website: locally hosted or externally hosted. We'll take a look at both in this chapter. First, we'll be adding a locally hosted web font to our example project. This is a three-step process:

1. Download a web font and add it to your project.
2. Embed the web font in your stylesheet.

Open up one of the HTML files with a web browser, and you'll find that our initial project is pretty close to the final example, minus all the web fonts and other CSS typography properties.

LOCALLY HOSTED WEB FONTS

There are two distinct methods of adding web fonts to your website: locally hosted or externally hosted. We'll take a look at both in this chapter. First, we'll be adding a locally hosted web font to our example project. This is a three-step process:

1. Download a web font and add it to your project.
2. Embed the web font in your stylesheet.
3. Use the font elsewhere in your stylesheet.

LOCALLY HOSTED WEB FONTS

There are two distinct methods of adding web fonts to your website: locally hosted or externally hosted. We'll take a look at both in this chapter. First, we'll be adding a locally hosted web font to our example project. This is a three-step process:

1. Download a web font and add it to your project.
2. Embed the web font in your stylesheet.
3. Use the font elsewhere in your stylesheet.

We'll be experimenting in the `web-fonts.html` and `typo.css` files. Go ahead and open those up in your text editor if you haven't already.

HOSTING A WOFF FILE

LOCALLY HOSTED WEB FONTS

There are two distinct methods of adding web fonts to your website: locally hosted or externally hosted. We'll take a look at both in this chapter. First, we'll be adding a locally hosted web font to our example project. This is a three-step process:

1. Download a web font and add it to your project.
2. Embed the web font in your stylesheet.
3. Use the font elsewhere in your stylesheet.

We'll be experimenting in the `web-fonts.html` and `typo.css` files. Go ahead and open those up in your text editor if you haven't already.

HOSTING A WOFF FILE

LOCALLY HOSTED WEB FONTS

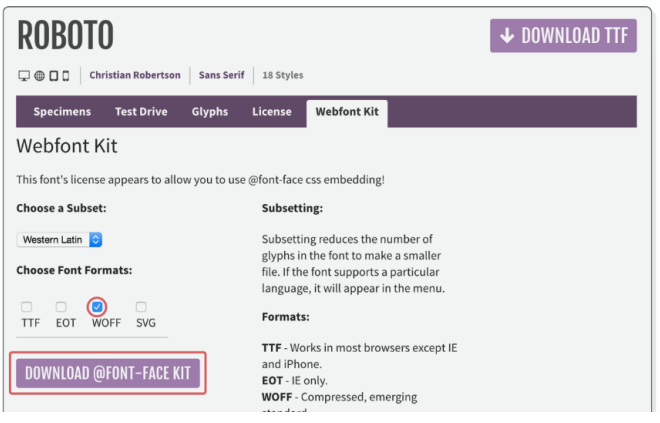
There are two distinct methods of adding web fonts to your website: locally hosted or externally hosted. We'll take a look at both in this chapter. First, we'll be adding a locally hosted web font to our example project. This is a three-step process:

1. Download a web font and add it to your project.
2. Embed the web font in your stylesheet.
3. Use the font elsewhere in your stylesheet.

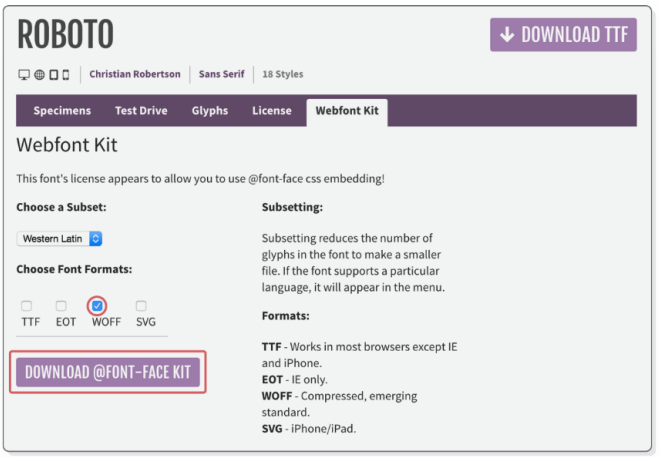
We'll be experimenting in the `web-fonts.html` and `typo.css` files. Go ahead and open those up in your text editor if you haven't already.

HOSTING A WOFF FILE

So, we need to get our hands on a web font. Our example uses the free Roboto font, which you should download from Font Squirrel. Make sure to click the **Webfont Kit** tab, not the **Download TTF** button. Unclick all the formats except **WOFF**, since that's the only one we'll be using, then click the **Download @font-face Kit** button.



This will give you a ZIP file with a license, some instructions, and a



This will give you a ZIP file with a license, some instructions, and a `web fonts` folder containing a ton of subdirectories. The Roboto font comes in a bunch of different font faces like light, regular, bold, italic, and condensed. Each of those folders contains a different face. The one we want is called `roboto_light_macroman`. Open up that folder and copy the `Roboto-Light-webfont.woff` file into our `web-typography` project.

EMBEDDING A WEB FONT

Sweet. We've got a WOFF file. To actually use it in our web page, we need to embed it into our stylesheet with the `@font-face` "at-rule". Web fonts must always be included at the *top* of a stylesheet, so add the following to the very beginning of `typo.css`:

```
@font-face {
  font-family: 'Roboto';
  src: url('Roboto-Light-webfont.woff') format('woff');
}
```

The `font-family` property defines how we'll refer to this font later on. This is called `roboto_light_macroman`. Open up that folder and copy the `Roboto-Light-webfont.woff` file into our `web-typography` project.

EMBEDDING A WEB FONT

Sweet. We've got a WOFF file. To actually use it in our web page, we need to embed it into our stylesheet with the `@font-face` "at-rule". Web fonts must always be included at the *top* of a stylesheet, so add the following to the very beginning of `typo.css`:

```
@font-face {
  font-family: 'Roboto';
  src: url('Roboto-Light-webfont.woff') format('woff');
}
```

The `font-family` property defines how we'll refer to this font later on. This operates as an internal label, so it can be anything you want. It doesn't *need* to relate to the official name of the font, but it's usually more intuitive if it does. As we'll see in a moment, it's a good idea to keep the name as generic as possible (e.g., `Roboto` instead of `Roboto Light`).

Next, we have the `src` property, which defines the path to the `.woff` file via the `url()` notation. The path can be absolute, relative, or root-relative. If you use a relative path like we did here, it will always be relative to the `.css` file—not the HTML document. The `format()` notation lets browsers

know which web font file format it is.

If you reload `web-fonts.html` page, you won't see any change because `@font-face` only gave us *access* to our `.woff` file. We still need to *use* it somewhere else in our stylesheet.

The `font-family` property defines how we'll refer to this font later on. This operates as an internal label, so it can be anything you want. It doesn't *need* to relate to the official name of the font, but it's usually more intuitive if it does. As we'll see in a moment, it's a good idea to keep the name as generic as possible (e.g., `Roboto` instead of `Roboto Light`).

Next, we have the `src` property, which defines the path to the `.woff` file via the `url()` notation. The path can be absolute, relative, or root-relative. If you use a relative path like we did here, it will always be relative to the `.css` file—not the HTML document. The `format()` notation lets browsers know which web font file format it is.

If you reload `web-fonts.html` page, you won't see any change because `@font-face` only gave us *access* to our `.woff` file. We still need to *use* it somewhere else in our stylesheet.

USING A WEB FONT

Remember from Defining Fonts that the CSS `font-family` property defines which font a particular HTML element uses. After adding our `@font-face` generic as possible (e.g., `Roboto` instead of `Roboto Light`).

Next, we have the `src` property, which defines the path to the `.woff` file via the `url()` notation. The path can be absolute, relative, or root-relative. If you use a relative path like we did here, it will always be relative to the `.css` file—not the HTML document. The `format()` notation lets browsers know which web font file format it is.

If you reload `web-fonts.html` page, you won't see any change because `@font-face` only gave us *access* to our `.woff` file. We still need to *use* it somewhere else in our stylesheet.

USING A WEB FONT

Remember from Defining Fonts that the CSS `font-family` property defines which font a particular HTML element uses. After adding our `@font-face` at-rule, we can use `Roboto` as a valid value for `font-family` anywhere else in our stylesheet.

Let's make Roboto Light the default font for our entire example project by changing the `font-family` in the `body` selector of `typo.css`:

```
body {
  font-family: 'Roboto', sans-serif;   /* Add 'Roboto' here */
  font-size: 18px;
  line-height: 1.8em;
  color: #5D6063;
}
```

USING A WEB FONT

Remember from Defining Fonts that the CSS `font-family` property defines which font a particular HTML element uses. After adding our `@font-face` at-rule, we can use `Roboto` as a valid value for `font-family` anywhere else in our stylesheet.

Let's make Roboto Light the default font for our entire example project by changing the `font-family` in the `body` selector of `typo.css`:

```
body {
  font-family: 'Roboto', sans-serif;   /* Add 'Roboto' here */
  font-size: 18px;
  line-height: 1.8em;
  color: #5D6063;
}
```

Everything should now render as Roboto Light, which means we lost our comparison with the sans-serif system font in `web-fonts.html`. Fix this by Remember from Defining Fonts that the CSS `font-family` property defines which font a particular HTML element uses. After adding our `@font-face` at-rule, we can use `Roboto` as a valid value for `font-family` anywhere else in our stylesheet.

Let's make Roboto Light the default font for our entire example project by changing the `font-family` in the `body` selector of `typo.css`:

```
body {
  font-family: 'Roboto', sans-serif;   /* Add 'Roboto' here */
  font-size: 18px;
  line-height: 1.8em;
  color: #5D6063;
}
```

Everything should now render as Roboto Light, which means we lost our comparison with the sans-serif system font in `web-fonts.html`. Fix this by adding a page-specific style to the `<head>` of our `web-fonts.html` file:

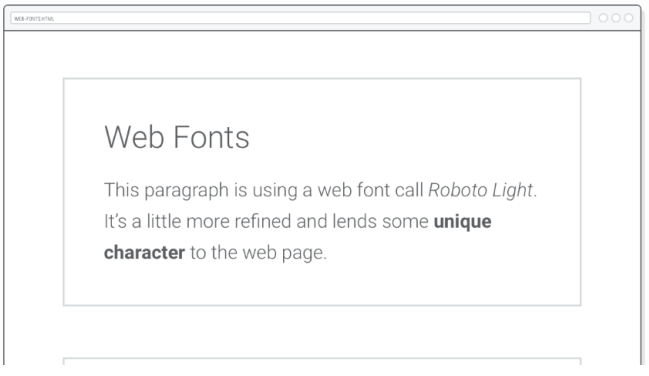
```
<style>
  .system-fonts {
    font-family: sans-serif;
  }
</style>
```

```
body {
  font-family: 'Roboto', sans-serif;   /* Add 'Roboto' here */
  font-size: 18px;
  line-height: 1.8em;
  color: #5D6063;
}
```

Everything should now render as Roboto Light, which means we lost our comparison with the sans-serif system font in `web-fonts.html`. Fix this by adding a page-specific style to the `<head>` of our `web-fonts.html` file:

```
<style>
  .system-fonts {
    font-family: sans-serif;
  }
</style>
```

The `.system-fonts` class is applied to the second box in `web-fonts.html`. The above rule takes precedence over the `body` rule in `typo.css`, so when you open up `web-fonts.html` in a browser, you should see our Roboto Light web font on the top and the default system font on the bottom:



System Fonts

This paragraph is using a built-in sans-serif font, which is probably Arial or Helvetica. It's not exactly ugly, but it's definitely familiar.

Web Fonts

This paragraph is using a web font call *Roboto Light*. It's a little more refined and lends some **unique character** to the web page.

System Fonts

This paragraph is using a built-in sans-serif font, which is probably Arial or Helvetica. It's not exactly ugly, but it's definitely familiar.

FONT FAMILIES AND FONT FACES

A single font "family" is made up of multiple font "faces". Each font face is a different weight or style in the family. "Weight" refers to the boldness of a particular face, and "style" refers to whether it's roman (upright), italic, condensed, extended, or some other variant in the family.

In our example, Roboto Light is one font face in the Roboto family. The other 17 faces in the ZIP file we downloaded earlier can be visualized like so:

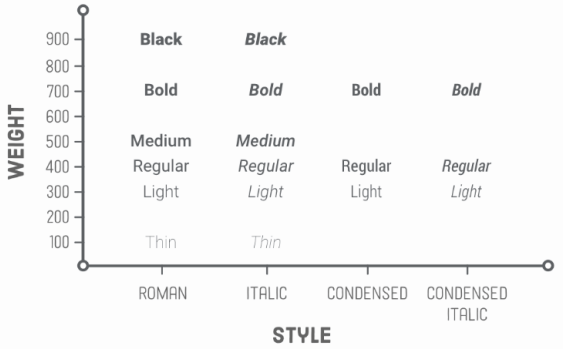
FONT FAMILIES AND FONT FACES

A single font "family" is made up of multiple font "faces". Each font face is a different weight or style in the family. "Weight" refers to the boldness of a particular face, and "style" refers to whether it's roman (upright), italic, condensed, extended, or some other variant in the family.

In our example, Roboto Light is one font face in the Roboto family. The other 17 faces in the ZIP file we downloaded earlier can be visualized like so:

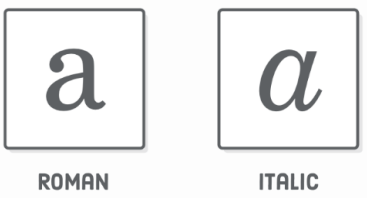
In our example, Roboto Light is one font face in the Roboto family. The other 17 faces in the ZIP file we downloaded earlier can be visualized like so:

In CSS, font weights are expressed as numeric values between 100 and 900. Fortunately, there are relatively standardized, human-friendly terms for so:



In CSS, font weights are expressed as numeric values between 100 and 900. Fortunately, there are relatively standardized, human-friendly terms for each of these numeric values. “Black” usually means 900, “bold” is 700, “regular” is 400, etc. As you can see above, most families don’t supply a face for every single weight. Roboto is missing “extra light” (200), “semi bold” (600), and “extra bold” (800).

It’s worth noting that each style and weight combination is designed as an entirely distinct face. In a high-quality font family, the condensed styles aren’t simply squashed versions of the roman faces, nor is the bold face merely a thicker version. Each letter in every face is hand-crafted to ensure it provides a uniform flow to its text.



This is particularly apparent in the italic and roman faces of many serif fonts. For instance, the lowercase “a” in Century Schoolbook FS (the font you’re reading right now) takes on a completely different shape when it’s *italicized*.

FAKIN' IT

Why does this weight and style stuff matter to us? The design of most websites utilizes multiple faces in the same family, so we need to know how to embed several `.woff` files that represent related faces.

But first, let's take a look at what happens when we *don't* offer multiple faces. Update the left-hand paragraph in `web-fonts.html` to include an `<em>` and a `<strong>` element:

```
<section class='section section--gray'>
  <h2>Web Fonts</h2>

  <p>This paragraph is using a web font call <em>Roboto Light</em>. It's a
  little more refined and lends some <strong>unique character</strong> to
  the web page.</p>
</section>
```


FAKIN' IT

Why does this weight and style stuff matter to us? The design of most websites utilizes multiple faces in the same family, so we need to know how to embed several `.woff` files that represent related faces.

But first, let's take a look at what happens when we *don't* offer multiple faces. Update the left-hand paragraph in `web-fonts.html` to include an `<em>` and a `<strong>` element:

```
<section class='section section--gray'>
  <h2>Web Fonts</h2>

  <p>This paragraph is using a web font call <em>Roboto Light</em>. It's a
  little more refined and lends some <strong>unique character</strong> to
  the web page.</p>
</section>
```


When you reload the page, you'll notice that the bold text isn't really all that bold. This is because it's being *synthesized*. We didn't supply a bold font

FAKIN' IT

Why does this weight and style stuff matter to us? The design of most websites utilizes multiple faces in the same family, so we need to know how to embed several `.woff` files that represent related faces.

But first, let's take a look at what happens when we *don't* offer multiple faces. Update the left-hand paragraph in `web-fonts.html` to include an `<em>` and a `<strong>` element:

```
<section class='section section--gray'>
  <h2>Web Fonts</h2>

  <p>This paragraph is using a web font call <em>Roboto Light</em>. It's a
  little more refined and lends some <strong>unique character</strong> to
  the web page.</p>
</section>
```

When you reload the page, you'll notice that the bold text isn't really all that bold. This is because it's being *synthesized*. We didn't supply a bold font

FAKIN' IT

Why does this weight and style stuff matter to us? The design of most websites utilizes multiple faces in the same family, so we need to know how to embed several `.woff` files that represent related faces.

But first, let's take a look at what happens when we *don't* offer multiple faces. Update the left-hand paragraph in `web-fonts.html` to include an `<em>` and a `<strong>` element:

```
<section class='section section--gray'>
```

```
  <h2>Web Fonts</h2>

  <p>This paragraph is using a web font call <em>Roboto Light</em>. It's a
  little more refined and lends some <strong>unique character</strong> to
  the web page.</p>
</section>
```

When you reload the page, you'll notice that the bold text isn't really all that bold. This is because it's being *synthesized*. We didn't supply a bold font face for the `<strong>` element to use, so the browser is trying to fake it by auto-converting Roboto Light into a thicker face. The same thing is going on with the italics in the `<em>` element, but it's a little bit harder to tell. This auto-conversion almost always results in low-quality typography.

This paragraph is using synthesized bold.

This paragraph is using **genuine bold**.

To verify that the bold and italic faces really are being synthesized, try adding the following rule to `typo.css`. The `font-synthesis` property determines if a browser is allowed to fake it or not. At the time of this writing, only Firefox actually pays attention to `font-synthesis`, so this won't work in Chrome or Safari:

```
/* This will only work in Firefox */
em, strong {
  font-synthesis: none;
}
```

Open up `web-fonts.html` in Firefox, and the `<em>` and `<strong>` elements

This paragraph is using synthesized bold.

This paragraph is using **genuine bold**.

To verify that the bold and italic faces really are being synthesized, try adding the following rule to `typo.css`. The `font-synthesis` property determines if a browser is allowed to fake it or not. At the time of this writing, only Firefox actually pays attention to `font-synthesis`, so this won't work in Chrome or Safari:

```
/* This will only work in Firefox */
em, strong {
  font-synthesis: none;
}
```

Open up `web-fonts.html` in Firefox, and the `<em>` and `<strong>` elements will no longer be italic or bold—the entire paragraph will be in roman Roboto Light.

This paragraph is using synthesized bold.

This paragraph is using **genuine bold**.

To verify that the bold and italic faces really are being synthesized, try adding the following rule to `typo.css`. The `font-synthesis` property determines if a browser is allowed to fake it or not. At the time of this writing, only Firefox actually pays attention to `font-synthesis`, so this won't work in Chrome or Safari:

```
/* This will only work in Firefox */
em, strong {
  font-synthesis: none;
}
```

Open up web-fonts.html in Firefox, and the and elements

Open up web-fonts.html in Firefox, and the and elements will no longer be italic or bold—the entire paragraph will be in roman Roboto Light.

This paragraph is using synthesized bold.

This paragraph is using **genuine bold**.

To verify that the bold and italic faces really are being synthesized, try adding the following rule to typo.css. The font-synthesis property determines if a browser is allowed to fake it or not. At the time of this writing, only Firefox actually pays attention to font-synthesis, so this won't work in Chrome or Safari:

```
/* This will only work in Firefox */
em, strong {
  font-synthesis: none;
}
```

Open up web-fonts.html in Firefox, and the and elements will no longer be italic or bold—the entire paragraph will be in roman Roboto Light.

This paragraph is using **genuine bold**.

To verify that the bold and italic faces really are being synthesized, try adding the following rule to typo.css. The font-synthesis property determines if a browser is allowed to fake it or not. At the time of this writing, only Firefox actually pays attention to font-synthesis, so this won't work in Chrome or Safari:

```
/* This will only work in Firefox */
em, strong {
  font-synthesis: none;
}
```

Open up web-fonts.html in Firefox, and the and elements will no longer be italic or bold—the entire paragraph will be in roman Roboto Light.

MULTIPLE FONT FACES (THE WRONG WAY)

Let's try adding Roboto Light Italic and Roboto Bold faces to our example project. Copy over the following files from the Roboto ZIP file we downloaded earlier into our web-typography folder:

- roboto_bold_macroman/Roboto-Bold-webfont.woff
- roboto_lightitalic_macroman/Roboto-LightItalic-webfont.woff

A .woff file represents a single face in a particular font family, and @font-face lets us embed that face in our stylesheet. The naive way to embed

Open up web-fonts.html in Firefox, and the and elements will no longer be italic or bold—the entire paragraph will be in roman Roboto Light.

MULTIPLE FONT FACES (THE WRONG WAY)

Let's try adding Roboto Light Italic and Roboto Bold faces to our example project. Copy over the following files from the Roboto ZIP file we downloaded earlier into our web-typography folder:

- roboto_bold_macroman/Roboto-Bold-webfont.woff
- roboto_lightitalic_macroman/Roboto-LightItalic-webfont.woff

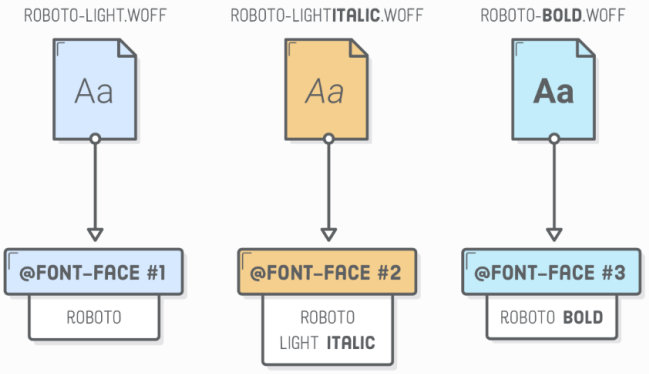
A `.woff` file represents a single face in a particular font family, and `@font-face` lets us embed that face in our stylesheet. The naive way to embed these new WOFF files would be to simply add more `@font-face` declarations and change the `font-family` and `src` properties as necessary. Try adding the following to the top of `typo.css`:

```
/* DON'T NAME FONT FAMILIES LIKE THIS */
@font-face {
  font-family: 'Roboto Light Italic';
  src: url('Roboto-LightItalic-webfont.woff') format('woff');
}

@font-face {
  font-family: 'Roboto Bold';
  src: url('Roboto-Bold-webfont.woff') format('woff');
}
```

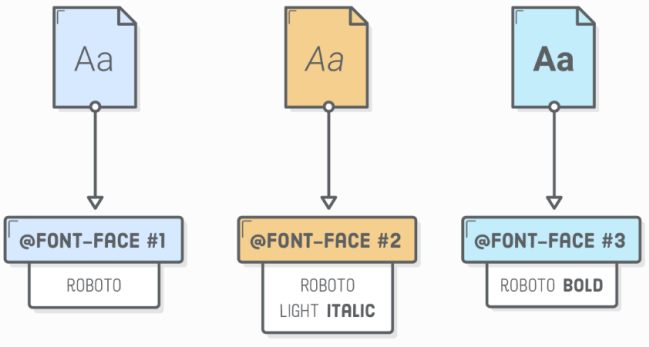
Then, to use these faces in our `<em>` and `<strong>` elements, we need the following rules:

```
/* THIS IS A LITTLE AWKWARD */
em {
  font-family: 'Roboto Light Italic', serif;
}

strong {
  font-family: 'Roboto Bold', serif;
}
```

This *will* work, and you should now see proper italic and bold fonts when you reload `web-fonts.html` in your browser. The problem is that manually specifying the `font-family` every time we want to use an italic or bold font is a little weird. We should be using the CSS `font-style` and `font-weight` properties for this.



We ended up in this awkward situation because of the way we embedded our new `.woff` files. Using separate `font-family` values in `@font-face` makes them look like entirely unrelated font faces. It doesn't reflect the fact that they are all actually part of the Roboto family.



We ended up in this awkward situation because of the way we embedded our new `.woff` files. Using separate `font-family` values in `@font-face`

makes them look like entirely unrelated font faces. It doesn't reflect the fact that they are all actually part of the Roboto family.

For this reason, you should **never use the above technique to embed multiple faces that are in the same font family**. Go ahead and delete both of the above snippets before moving on.

MULTIPLE FONT FACES (THE RIGHT WAY)

To maintain the familial relationship between our three font faces, they all need to use a shared `Roboto` value for their `font-family` property. To distinguish between our light, italic, and bold faces, we'll add `font-style` and `font-weight` properties to the at-rule. Replace all the `@font-face` declarations in `typo.css` with the following:

```
@font-face {
  font-family: 'Roboto';
  src: url('Roboto-Light-webfont.woff') format('woff');
  font-style: normal;
  font-weight: 300;
}

@font-face {
  font-family: 'Roboto';
  src: url('Roboto-LightItalic-webfont.woff') format('woff');
  font-style: italic;
  font-weight: 300;
}
```

Think of each `@font-face` at-rule as a description of the underlying `.woff` file. The first `@font-face` is saying it's a Roboto font that's roman (`normal`) and has a font weight of 300 (aka "light"). The second says it's also in the Roboto family and has a weight of 300, but it's italic. Finally, the third at-rule lets our the browser know that `Roboto-Bold-webfont.woff` contains the 700-weight (aka "bold") roman face.