Learn How Fonts And Web Typography Work In HTML: A Beginner's Guide

Last Updated on December 23, 2019

The web is a place where ideas are shared. Sometimes these ideas are visual or auditory and conveyed using pictures, videos, or audio. However, more often than not, content on the web takes the form of written text. This means that the design of the visual presentation of text on the web – web typography – is central to the success of virtually every website. In this tutorial, we're going to take a detailed look at two distinct but related topics:

1. **Fonts**: Selecting fonts and techniques for adding them to a website.
2. **Web Typography**: The techniques and strategies used to style and arrange text on a web page.

The first topic deals with the technical aspects of importing fonts into a website's files so that visitors who visit the website have access to the fonts you want them to see. The second topic deals with the art and technique of using CSS to style fonts for maximum readability and design impact. However, before we dive into the technical implementation details, it's important to have a font and typography strategy in mind.

Contents [hide]

Getting Started with Fonts and Typography

When you approach the topic of web typography for the first time, you can quickly become overwhelmed. Coming up with a plan for addressing font selection and text styling throughout your site can be a daunting task if you don't know where to start. That's where this tutorial comes in. Rather than dive into the technical details we're going to start by explaining some of the fundamental ideas that need to guide your font selection and text design process. **1. Take the Time to Learn About Fonts and Typography** That's what you're doing right now, so you're off to a great start! There are several terms and concepts you need to get familiar with right off the bat:

- Font and typeface: While some experts argue that these terms mean two different things, in modern usage they are synonymous. In the context of the web, a font or typeface is a set of letters, symbols, and punctuation marks of a similar design and packaged together into a single computer file.
- Font family: A font family includes several different fonts that are identical other than stroke thickness and the use of italics. If you plan to use normal, bold, and italicized fonts, you will need to include at least those three fonts or typefaces for each font family you plan to use.
- Typography: A broad term that encompasses font selection as well as topics such as color, size, character spacing, line spacing, and alignment. *Web typography* is the discipline of designing text for presentation on the web.
- Style guide: A set of standards that describe the design, style, and branding to be applied consistently across all parts of a website. Developing a style guide helps create a consistent experience for website visitors and a style guide includes things like typography, color choice, writing style, and much more. Selecting fonts and designing typography is one aspect of developing a style guide.

2. Limit Your Website to 4 or Fewer Font Families Lots of websites get by with just two fonts: one for heading elements and one for paragraph text. It's ok to have more than two, but if you find yourself exceeding four fonts try to see if one or more can be eliminated. Using too many fonts can have a negative effect on visitor experience and on the effectiveness of your site's branding. Not to mention that using lots of fonts can slow down the performance of your website by forcing visitor's browsers to load multiple font files. **3. Use Fonts that Fit the Personality of Your Website** Fonts should enhance your website. They should never be a distraction or detract from your website visitors' experience. If you're designing a website for a lawyer's office you won't want to use a whimsical hand-lettered font, that will come across as unprofessional and inappropriately informal. Pick fonts that fit the tone of the message your website is trying to convey. **4. Typography is Ultimately About Enhancing Readability** As you begin trying out different fonts and styling techniques it will be easy to get carried away. Before settling on a font or a specific style take a moment to make sure that what you've created fits with your ultimate goal of providing the best experience possible for your website visitors. **5. Use Heading Styles to Add a Sense of Structure** Make your headings considerably larger than your paragraph text, and make the change in size from one heading level to the next level noticeable enough that the structure of the document is reinforced by the font sizing. Use a different font for heading and paragraph text to emphasize that the headings represent the beginning of a new section or topic. One common technique is to use a serif font for headings and a sans-serif font for paragraph text. Some designers prefer to do the opposite, and if you decide to do the same and use a serif font for your paragraph text be sure to select a large font size since serif fonts are not as easy to read as sans-serif fonts. **6. Get Creative with Headings and Logos** The name of your website, the tagline, navigation menu labels, and the headings throughout your site are the right great place to experiment with display fonts, hand-lettered scripts, all-caps, drop caps, and other eye-catching typography choices. Stick to simple, easy-to-read fonts for blocks of content such as paragraph, tables, and lists. **7. Pay Attention to Alignment and Spacing** The space between each line of text and the alignment of text can be controlled with CSS. Does your paragraph text seem too bunched up? Would article headings look better if centered over the body of a post? Working with fonts and typography is the right time to be detail-oriented. Go ahead and sweat the details. **8. Don't Forget to Maintain Whitespace** Assuming you've selected a readable font and font size, nothing enhances readability more than providing adequate whitespace between elements on your web page. Leaving plenty of room to breathe on a web page is a good thing. When a visitor reaches a web page only to see a wall of text staring back at them, they are most likely to click away before reading any of the content. Spread out blocks of text, intersperse graphics and lists, break up the flow with headings, and use generous line-height spacing to make reading your website's content easy on the eyes.

Working with Fonts: The Basics

Getting started with fonts is easy. One thing you should get under your belt from the beginning is how to use CSS to apply fonts to different HTML elements. First, let's look at how to apply a font to a single HTML element and then we'll look at how to apply a single font to an entire web page.

Breaking the `<font>` Habit: Getting Started

Let's begin by looking at the basic structure for using styles. Styles are set by adding a `<style>` element to the `<head>` section of your page– or better yet, by adding a `<link>` to an external stylesheet. So, for example, the following code, which should be copied into the `<head>` section, sets a style rule for `<h2>` elements:

```
<style>    h2 {      color:green;      font-size:30pt;      font-style:italic;    } </style>
```

The first line 1 opens the `<style>` element. The next line begins our style rule. A style rule has two parts: the selector, which indicates which element(s) the style applies to, and a list of declarations (surrounded by curly braces) which define how the element should be presented. In this case, the selection is `h2` , which means the following rule applies to all `<h2>` elements. The selector is followed by an opening curly brace. The list of declarations is nested between the opening and closing curly braces. Note that the white space (line breaks and spaces) between each declaration are just their to make the code more readable and have no effect on how the code works. The declarations say that the font is green, that the font should be 30 points tall, and that the font should be italic. After the final declaration we close ruleset with the closing curly brace, ending the list of declarations. Finally, on the last line we see the closing `<style>` tag. This rule automatically applies to all `<h2>` elements. So if we create a header using an `<h2>` tag:

```
<h2>Education</h2>
```

That header is automatically rendered in green, 30 point, italic font:

span.example{color:green; font-size:30pt; font-style:italic;}Education

Setting the Font on the Whole Page

In the last example we looked at how to apply a font to a single `<h2>` element. Setting font properties for an entire web page isn't really any different, just have to target the right element: `<body>` . For example, suppose we want the entire page to use the generic sans-serif system font. We can do that with this CSS declaration:

```
body {   font-family: sans-serif;   }
```

A popular reason to make a global CSS declaration about fonts on a webpage is to make font colors are compatible with a global background color. In that case, we can set all the style properties using CSS and globally applying them to the entire webpage. For example, the following code sets the background color of the page to black, the font to white, and the font family to sans-serif. It also sets the link color to white and the visited link color to yellow.

```
body {   color: white;    background-color: black;    font-family: sans-serif;  }  a {   color: white; } a:vis
ited {   color: yellow; }
```

If we apply that CSS to a webpage, here's what we get:

Part 1: Selecting and Using Fonts

There are a few different categories of fonts that you will encounter in your search for the perfect fonts:

- Serif and sans-serif: Have you noticed that some letters have plain ends while others have a short stroke at the end of each letter? Those short strokes are called serifs. Serif fonts have them and sans-serif fonts don't.
- Scripts and hand lettering: These fonts appear to have been written by hand. Some are formal while others are quite informal. These may be a good choice for a certain type of website with a lot of personality, but should only be used for headings and logos and not for paragraphs or lists.
- Monospace: Fonts where every character takes up the same amount of horizontal space are called monospace fonts.
- Display and decorative fonts: Display and decorative fonts are striking font designs intended for limited usage for logos and top-level headings.

Serif and sans fonts make up the bulk of fonts you see on the web because they're easy to read, render well on most devices, and fallback to default web safe fonts without causing major design issues. Scripts, hand lettering, display, and decorative styles are often used for logos, navigation menu labels, page titles, and top-level headings. Monospace fonts are typically only used to display block of computer programming code although some technically-focused websites may use monospace fonts for other purposes.

Building a Font Stack

Not all browsers and operating systems are capable of rendering all fonts. As a result, when selecting fonts it's important to specify fallback fonts styles. This is done by building a *font stack* using the `font-family` CSS property. Typically a font-stack will include at least three font options: a web font you've selected as the design font, a fallback generic font pre-installed on nearly all browsers and operating systems, and finally a generic font family that the browser can use to select a font from the fonts available on the system. In practice, a font-stack for paragraph elements might look like this:

```
p {   font-family: "Open Sans", Tahoma, sans-serif; }
```

When a browser processes this bit of CSS it will first try to use the `"Open Sans"` font – a sans-serif web font. If it can't find or render *Open Sans* it will try the next option: `Tahoma` . Virtually all devices are able to render *Tahoma*, but the miniscule portion of visitors using unsupported devices will see a generic font selected by their browser or operating system that falls into the `sans-serif` family. There's no reason why we have to limit our stack to three options. If you'd like to leave even less to chance, you could always add a few additional fonts such as Arial and Helvetica to the font stack to provide additional fallback options before letting the visitor's device choose a sans-serif alternative.

Font Stack Syntax

You may have noticed that one of the font names in the `font-family` property is surrounded by double-quotes while the others are not. When a font name contains spaces it must be wrapped in quotation marks. It's also worth mentioning that if you ever happen to use the `style` HTML attribute in combination with the `font-family` property, then single quotes should be used rather than the double quotes used in the example above. However, there are very few instances where adding inline fonts makes any sense and in most cases you will want to declare fonts in an external CSS stylesheet.

Web Safe Fonts

In the font stack above we progressed from a web font delivered along with the website files, to a common font installed on most operating systems, and finally simply told the browser the font family from which it should select an available font. The common font, in this case *Tahoma*, is called a web safe font. Web safe fonts are those that are supported by the majority of computer operating systems and browsers and are therefore safe to use on the web. There are quite a few web safe fonts that you can use in a font stack with a high degree of confidence that your website visitors will not have any trouble viewing your website with the font you intended:

- Web safe **serif** fonts: Times New Roman, Georgia, Palatino, Lucida Bright.

- Web safe **sans-serif** fonts: Arial, Tahoma, Trebuchet MS, Verdana.
- Web safe **monospaced** fonts: Courier, Courier New, Lucida Sans Typewriter.
- Web safe **specialty** fonts: Copperplate, Papyrus, Arial Black, Arial Narrow.

These are the web safe fonts with the broadest support across all operating systems. However, this abbreviated list is not exhaustive, and you can find more web safe fonts at CSS Font Stack.

Testing a Font Stack

It's important to remember that even when sized by the same value, different fonts still render in ways that make them appear to be significantly different sizes. Take *Times New Roman* and *Lucida Bright* for example. These web safe fonts are commonly used as a fallback option for serif style web fonts. However, if we compare them side by side we can see some significant differences.

```
<style> .timestext {   font-family: "Times New Roman", Times, serif;   font-size: 1em; } .lucidatext {   font-family: "Lucida Bright", Lucida, serif;   font-size: 1em; } </style> <p class="timestext">This sentence is written in Times New Roman.</p> <p class="lucidatext">This sentence is written in Lucida Bright.</p>
```

The code above defines two HTML paragraph elements. To the first we have applied a CSS rule causing it to be displayed in 1em size Times New Roman font. The second sentence will be displayed in 1em size Lucida Bright font. On the surface, we might think these two sentences should look extremely similar.

.timestext { font-family: "Times New Roman", Times, serif; font-size: 1em; } .lucidatext { font-family: "Lucida Bright", Lucida, serif; font-size: 1em; }

This sentence is written in Times New Roman.

This sentence is written in Lucida Bright.

So what happened? If you open up a word processing application and display the same sentence in two different fonts you'll see the same behavior. Different fonts display quite differently even when sized to the same font size. This demonstrates why it's important to test your website using every font in the font stack. While your website content may look great in *Lucida Bright*, if you use *Times New Roman* as a fallback your text may end up a little too small for most visitors to read comfortably if their system doesn't support *Lucida Bright*. Here's how you can test your font stack to make sure the fallback font options render well:

1. Comment out the all but one of the fonts in the font stack and view your website with just one font in the font stack active.
2. Repeat the process until you have viewed your site with each of your fallback fonts.
3. If any fonts produce readability issues, remove them from the stack.
4. Make sure you keep at least three or four fonts that render well in each font stack to provide adequate browser and operating system coverage in your font stack.

Using Web Fonts

While it's important to use web safe fonts in your font stack to provide fallback options for your website's visitors, once you discover the wealth of web fonts available for you to use, you won't be satisfied limiting yourself to just one or two dozen web safe fonts. Web fonts aren't preinstalled on your website vistors' devices. Instead, they are delivered to your website visitors along with the rest of your website's content. While they can have a modest effect on website load speed, broadband Internet connections minimize the impact of this delay and most modern websites use web fonts. When you use web fonts, they should be built into a font stack. The web font will appear first in the stack, just

as we listed the *Open Sans* web font first in our example font stack. The fallback fonts will be used if the web font is unavailable and when a website visitor is using an unsupported browser. There are two different ways to add web fonts to your website.

Self-Hosted Fonts

Self-hosted web fonts are hosted along with your website files. When a visitor reaches your website, the font files are sent from the server to the visitor's browser along with the rest of the files that make up your site. A CSS at-rule is used to load web fonts that exist on your website's server. The web font we mentioned earlier, *Open-Sans*, can be downloaded for free from Font Squirrel. To use *Open Sans* as a self-hosted font we would first download the font in all available formats and upload them to the same directory as our CSS files. The CSS at-rule used to load the font would be added to one of the CSS stylesheet already linked to the HTML documents where we want to use the font. The at-rule to load the self-hosted font looks like this:

```
/*The following rule is a modified version of the rule included with the download of Open Sans from Font Squir
rel. Do not copy and paste this rule. Download the font from a font service and use the code provided with the
font.*/  @font-face{    font-family: 'Open Sans';   src: url('OpenSans-Regular-webfont.eot');   src: url('Open
Sans-Regular-webfont.eot?iefix') format('embedded-opentype'),        url('OpenSans-Regular-webfont.woff') form
at('woff'),        url('OpenSans-Regular-webfont.ttf') format('truetype'),        url('OpenSans-Regular-webfon
t.svg#open_sans') format('svg'); }
```

Now we could add *Open Sans* to any font stack in our CSS stylesheet and the selected HTML elements would be displayed in *Open Sans*. Font Squirrel is just one of many places where you can find free and premium downloadable fonts to use as self-hosted fonts. Two other font marketplaces to check out include Creative Market and Font Spring.

CDN-Hosted Fonts

There are at least three services that host fonts on public facing content delivery networks (CDN). To use these fonts you don't have to have the font files. Instead, you add a bit of code to the `head` of your HTML document and the browser downloads the files from the CDN. The three most well-known services that use this model include:

- Google Fonts: This completely free service is probably the most common way that web fonts are delivered. It's very easy to use and includes more 700 font families.
- Typekit: Typekit is considered to be the premier source of fonts. It comes bundled up with Adobe's Creative Suite, which most web designers use, and includes over one thousand high-quality font families. Full access to Typekit requires either an Adobe Creative Suite license or a paid Typekit account. However, a limited number of fonts can be used for free.
- Adobe Edge Web Fonts: This free service from Adobe includes about 500 free font families. Think of this service as a significantly scaled-down version of Typekit.

Using a hosted font service has the added benefit that you don't have to worry about getting the `@font-face` code syntax perfect, or think about including the right font file format for every user. The font service takes care of all of that. All you have to do it copy and paste a line or two of code from the font service directly into your HTML document, and then use the `font-family` CSS property to apply the font to different elements of your website. For example, *Open Sans* is available from all three of the font services mentioned above. Here's the code you would use to add this font to your web page using all of these sources.

```
<head>   <!--Google Fonts are added using the link element-->   <link href='https://fonts.googleapis.com/css?f
amily=Open+Sans' rel='stylesheet' type='text/css'>   <!--Edge Fonts code uses a JavaScript file to deliver the
font-->   <script src="//use.edgefonts.net/open-sans.js"></script>   <!--Typekit Fonts are a bit more complica
ted to use-->   <script src="https://use.typekit.net/font_kit.js"></script>   <script>try{Typekit.load({ asyn
c: true });}catch(e){}</script> </head>
```

Each of these font services works a little differently. With Google Fonts, you can add a single font if you like, or build a collection of fonts and import all of the fonts using a single `link` element. Edge Fonts have to be added one at a time with a new `script` element for each font you want to use on your website. Typekit fonts can only be added by building a font kit at the Typekit website, which is similar to a Google Fonts

collection. The entire font kit is then added to your website at once. In the example above, the file name `font_kit.js` would actually be a unique series of numbers and letters automatically generated by Typekit to identify the font kit you created. In practice, you wouldn't use all three of these services at the same time. It is most common for websites to get all of their fonts from a single source or to self-host the fonts used by their website.

Icon Fonts

Icons are used extensively on the web. While some icons are image files, the most flexible way to work with icons is to use an icon font like Font Awesome. In the past, using icon fonts was a bit complicated and required that the icon font files be self-hosted. However, thanks to CDN hosting sponsored by MaxCDN and Bootstrap, anyone can add Font Awesome icons to their website without having to host the font files on their own server. Full instructions for getting started are available from Font Awesome's GitHub repository. However, it's really as easy as taking these two steps:

1. Add a single `link` element to the `head` of your HTML document.
2. Use the icon names to add the icons to HTML elements.

Here's an example of those two steps in action.

```
<head>   <!--This link is available at Font Awesome's "Get Started" page-->   <link rel="stylesheet" href="htt
ps://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css"> </head> <body>   <p><i class="fa fa
-lightbulb-o"></i> Using the i element produces the cleanest code.</p>   <p><span class="fa fa-check-square-
o"></span> Using the span element is more semantically correct.</p> </body>
```

Let's see what happens when we render that code on our web page.

> Using the i element produces the cleanest code. Using the span element is more semantically correct.

What makes icon fonts so much better to work with than their image icon counterparts is that they can be styled with all of the same CSS properties used to style fonts. For example, if we want to increase the icon size, add a round background color, change the icon color, and add some additional padding all the way around the icon we can do so with some fairly straightforward CSS.

```
.fa {   padding: 15px;   color: white;   background-color: gray;   border-radius: 50%;   font-size: 1.5em; }
```

Now when we render that same paragraph we can see that our rules have taken effect.

> .fa-example { padding: 15px; color: white; background-color: gray; border-radius: 50%; font-size: 1.5em; } Using the i element produces the cleanest code.
>
> Using the span element is more semantically correct.

Part 2: Web Typography

Designing the presentation of text on a web page involves two parts:

1. Creating a logical structure for the text content of your website.

2. Styling text for readability and personality.

The first task is accomplished with HTML and the second with CSS.

HTML and Text

It's important to know how to use all of the different HTML elements that impact the appearance of text on a web page for two reasons:

1. Using the *right* HTML elements to markup text on your website ensures that your markup is semantically accurate and meaningful.
2. When you style your markup with CSS, it will be a lot of easier if you've used the correct tags to start with.

There are two different categories of HTML elements that come into play when dealing with text: text-grouping elements and those that add text-level semantic meaning.

Text-Grouping Elements

There are a few elements used to add text-based content to a web page. These elements include paragraphs, headings, lists, and tables. Paragraphs start with a `<p>` tag, and end with a closing `</p>` tag. Paragraphs are used to add blocks of text content to your website: blog posts, articles, and web page copy. There are six heading elements. The highest level is the `<h1>` and the lowest level is the `<h6>` . In most cases, no more than three heading levels are needed on any given web page, but the others are available if you need them. Headings should be used heirarchically – that is, every page should have one `h1` with `h2` subheadings, each section that starts with an `h2` should contain subheadings marked by `h3` elements, and so forth. Don't skip heading levels and never select a heading level based on it's visual appearance when rendered in a browser. Use the right heading level based on the structural heirarchy of your content and use CSS to make the headings look the way you want them to look. There are three types of lists you can use on your web page: ordered lists, unordered lists, and description lists. Lists should be used to group together list items that are somehow correlated. They're also a great way to break up the flow of a blog post or article. Our list tutorial provides a detailed look at the proper use of these elements. Tables are another way to add structured text to a web page. In the past, tables were commonly and incorrectly used to create web page layouts. They are a lot less common today since they should only ever be used to display tabular data. Our tables tutorial provides a detailed demonstration of how to create and style HTML tables.

Adding Text-Level Semantics

There are a number of elements that can be used to add semantic meaning to text contained within one of the text-grouping elements discussed in the last section. These elements should be used on text inside of a parent text-grouping element such as a paragraph, heading, or list. **Draw Attention with Bold, Italics, and Highlighting** If you want to draw attention to a bit of text, there are several options available. Use these elements for their semantic meaning, not for their visual presentation, and then modify the appearance to match your preferences using CSS.

- The `strong` element is used to give strong importance to a word or series of words. Text wrapped in `strong` tags is typically displayed in bold typeface by web browsers.
- Text wrapped in `em` tags places emphasis on the text between the tags. Text emphasized in this way is typically displayed in italics by web browsers.
- Think of the `mark` element as a digital highlighter. If a certain piece of text has unique relevance on a web page wrapping it in `mark` tags will catch the attention of website visitors and assistive technologies.

Identify Definitions and Abbreviations When you use a technical term for the first time in a document, it is customary to define that term for website visitors who aren't familiar with it. The instance where the term occurs and is defined in the document is called the defining instance, and the defining instance of the term should be wrapped in `dfn` tags. The `abbr` element can be used to identify abbreviated terms and add a `title` attribute with the expanded version of the term. Consider the code block below for an example of how this tag might be used.

```
<p><dfn>HTML.com</dfn> (<abbr title="HTML.com">HCT</abbr>) is a free resource    for students of web technolog
ies such as <abbr title="Cascading Style Sheets">CSS</abbr>,    <abbr title="Hypertext Markup Language">HTML</
```

```
abbr>, and JavaScript.</p>
```

Keep Track of Document Changes If you want a document on the web to display the changes made to it over time you can use the `del` and `ins` tags to keep track of the changes. Text wrapped in `del` tags has been deleted from the document, and text wrapped in `ins` tags has been inserted into the document. Here's an example:

```
<p>This committee holds a teleconference at <del>10 AM</del><ins>2 PM</ins> EST on the <del>first Tuesday</del
><ins>second Monday</ins> of every month.</p>
```

When we render that sentence in a browser the deleted content and inserted content will be displayed with styling to indicate it's status as a deletion or insertion.

This committee holds a teleconference at ~~10 AM~~2 PM EST on the ~~first Tuesday~~second Monday of every month.

Tags to Know About but Use Sparingly There are three elements that are part of HTML5 but may run afoul of the first rule of web development: maintain separation between website content and presentation. These controversial elements are the `b`, `i`, and `small` elements.

- The `b` tag is used to mark text to be displayed in bold without implying increased importance.
- The `i` tag is used to mark text that is usually displayed in italics without implying increased emphasis.
- Side-comments or small print, such as copyright and legal notices, may be wrapped in `small` tags and will be displayed smaller than the standard font size of the parent element.

While all three of these tags valid HTML, many web designers and developers avoid using them, and you should only use them if there is no more appropriate tag you could use. **Dealing with Subscripts and Superscripts** If you need to mark text as being either a subscript or superscript you can use the `sub` and the `sup` tags. However, if you're creating a document that makes extensive use of subscripts and superscripts as part of mathematical expressions you would do well to research and use MathML rather than HTML as the markup language for your document. Superscripts are often used to add comments to an HTML document. For example, if we go back to our previous example of the `del` and `ins` elements, we could use superscripts to add and link notes to those changes.

```
<p>This committee holds a teleconference at <del>10 AM</del><ins>2 PM</ins><sup><a href="#note_1">1</a></sup>
EST on the <del>first Tuesday</del><ins>second Monday</ins><sup><a href="#note_2">2</a></sup> of every month.
</p> <ul>   <li><sup id="note_1">1</sup> Teleconference time change made during the January 5, 2016 teleconfer
ence.</li>   <li><sup id="note_2">2</sup> Teleconference date change made during the February 2, 2016 teleconf
erence.</li> </ul>
```

When rendered in a browser, a website visitor reading the document would be able to click on the linked superscript number to get more information about the change.

This committee holds a teleconference at ~~10 AM~~2 PM[1] EST on the ~~first Tuesday~~second Monday[2] of every month.

- [1] Teleconference time change made during the January 5, 2016 teleconference.
- [2] Teleconference date change made during the February 2, 2016 teleconference.

Acknowledging Quotations There are three elements commonly used when dealing with quotations. When quoting a person, book, website, or any other source, the quoted text should be wrapped in `q` tags if the quotation is inline with the surrounding text. If the quotation is long

enough that it occupies a standalone block of text the entire block should be wrapped in `blockquote` tags. When referencing a work, such as when giving appropriate credit for a quotation, wrap the name of the work in `cite` tags. For example, if we were to mention the book *Responsive Web Design* by Ethan Marcotte, the title of the book should be wrapped in `cite` tags – and it is. **Reversing the Direction of Text** Some languages, Arabic for example, are read from the right-to-left rather than left-to-right. To display right-to-left text in an otherwise left-to-right oriented document wrap it in `bdo` tags and use the `dir` attribute to override the default text behavior.

```
<p>If you need some text to render from right-to-left use the <code>bdo</code> tag and  apply the attribute <c
ode>dir="rtl"</code>. The code between the block will be  <bdo dir="rtl">displayed from right-to-left</bdo>.</
p>
```

Let's see how that looks when rendered in the browser.

> If you need some text to render from right-to-left use the `bdo` tag and apply the attribute `dir="rtl"` . The code between the block will be tfel-ot-thgir morf deyalpsid.

HTML5 introduced a new element, `bdi` , which can be used to accomplish the same task by leaving the directionality of the source of the text intact. However, browser support for the `bdi` element is still quite limited.

CSS and Text

There are at least three different categories of CSS properties which can be used to style text.

1. Properties used to style and format the element that contains the text.
2. Properties used to change the visual appearance of text.
3. Properties that affect the flow or layout of text within the parent element.

The properties used to style and format the element that contains the text are things like padding, margin, border, background color, columns, float, flexbox, and position. These are not text-specific properties. They can be used to style and position any element on a webpage. To learn more about using these properties visit the tutorials linked to in the paragraph above. In this tutorial, we'll focus on the CSS properties specifically used to work with text.

Styling Text

The most important CSS property to become familiar with when dealing with fonts and typography is the `font` property. This property is actually a shorthand way to specify the `font-style` , `font-variant` , `font-weight` , `font-size / line-height` , and `font-family` in one single rule. To understand each of these properties we'll take them one at a time, and then learn how to combine them into the shorthand form.

- `font-style` : The default value is `normal` . Other options include `italic` , `oblique` , `initial` , and `inherit` .
- `font-variant` : The most useful value to be applied to this property is `small-caps` . The default value is `normal` and you can also use `initial` or `inherit` .
- `font-weight` : The default value is `normal` . Other values that can be used include `bold` , `bolder` , `lighter` , `initial` , `inherit` , and any multiple of 100 up to 900 where 400 is equal to `normal` and 700 is equal to `bold` . Just keep in mind that the font you use must have each of these weights available in order for you to use them.
- `font-size` : There are many different values that can be used to adjust font size. Words like `medium` , `small` , and `large` are recognized as are numbered values expressed in pixels, ems, and percentages. In most cases, it makes the most sense to use either pixels or ems to size fonts.
- `line-height` : The default value for `line-height` is `normal` . However, this property also accepts declared values in pixels, points, ems, and percentages. If a number without a unit is provided it will be multiplied against the `font-size` to set the `line-height` .

- `font-family` : This property is used to build a font stack. One or more fonts can be specified and the browser will use the first font it processes that it is capable of rendering.

Let's apply all of these styles to a paragraph of text to see them in action. While we're at it, we'll also use Google Fonts to import *Open Sans* and apply it to our text as well.

```
<!--First, we import Open Sans in both 400 and 600 typeface--> <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,600'    rel='stylesheet' type='text/css'> <!--Here is our CSS--> <style> .unstyled_p {   font: initial; } .unstyled_span {   font: initial; } .styled_p_1 {   font-style: normal;   font-variant: normal;   font-weight: 400;   font-size: 1.2em;   line-height: 1.5em;   font-family: "Open Sans", Arial, sans-serif; } .styled_span_1 {   font-style: italic;   font-variant: small-caps;   font-weight: 600;   font-size: 2em;   line-height: 1.5em;   font-family: "Open Sans", Arial, sans-serif; } .styled_p_2 {   font: 400 1.2em/1.5em "Open Sans", Arial, sans-serif; } .styled_span_2 {   font: italic small-caps 600 2em/1.5em "Open Sans", Arial, sans-serif; } </style> <p class="unstyled_p"><span class="unstyled_span">This is an Unstyled Span</span>   <br>This is an unstyled paragraph. We set all of the font properties to initial    to demonstrate the standard styling applied by your web browser. It does not    look like our styled spans and paragraphs.</p> <hr> <p class="styled_p_1"><span class="styled_span_1">This is Styled Span 1</span>   <br>This is styled paragraph 1. We styled this span and paragraph by declaring    each of the font properties seperately. As you can see, it looks just like    span and paragraph 2.</p> <hr> <p class="styled_p_2"><span class="styled_span_2">This is Styled Span 2</span>   <br>This is styled paragraph 2. We styled this span and paragraph by declaring    all of the font properties using the font shorthand property. As you can see,    it looks just like span and paragraph 1.</p>
```

We created three different paragraphs. Each one begins with a sentence wrapped in `span` tags followed by a line break. We've added additional styles to the spans to draw attention to these bits of text. For the first paragraph, we've set all `font` values to `initial` , which will reset all CSS styling on these elements to the browser default values. For the second paragraph, we've set all `font` values by using each of the individual properties. For the last paragraph, we've applied the same styles as we applied to the second paragraph, but we've done so using the `font` shorthand property. Here's how the paragraphs render in a browser.

.unstyled_p { font: initial; } .unstyled_span { font: initial; } .styled_p_1 { font-style: normal; font-variant: normal; font-weight: 400; font-size: 1.2em; line-height: 1.5em; font-family: “Open Sans”, Arial, sans-serif; } .styled_span_1 { font-style: italic; font-variant: small-caps; font-weight: 600; font-size: 2em; line-height: 1.5em; font-family: “Open Sans”, Arial, sans-serif; } .styled_p_2 { font: 400 1.2em/1.5em “Open Sans”, Arial, sans-serif; } .styled_span_2 { font: italic small-caps 600 2em/1.5em “Open Sans”, Arial, sans-serif; }

This is an Unstyled Span
This is an unstyled paragraph. We set all of the font properties to initial to demonstrate the standard styling applied by your web browser. It does not look like our styled spans and paragraphs.

This is Styled Span 1
This is styled paragraph 1. We styled this span and paragraph by declaring each of the font properties seperately. As you can see, it looks just like span and paragraph 2.

This is Styled Span 2
This is styled paragraph 2. We styled this span and paragraph by declaring all of the font properties using the font shorthand property. As you can see, it looks just like span and paragraph 1.

One thing to keep in mind when using the `font` shorthand property is that the `font-size` and `font-family` values are mandatory and should always appear last in the sequence of declared values. If either value is missing, the entire rule will be ignored. All other values are

optional. If you don't specify the optional values, the default value set by the browser or inherited from parent elements will apply. There are three more CSS properties frequently used to style the visual presentation of text. They are:

- `text-decoration` : This property is most frequently used to underline text. The default value is `none` . Other acceptable values include `underline` , `overline` , `line-through` , `initial` , and `inherit` .
- `text-shadow` : Add a shadow behind the selected text, move it horizontally and vertically by specifying the offset distance, blur the shadow, and change the color of the shadow. The values must be ordered in the following way: horizontal offset | vertical offset | blur | color.
- `text-transform` : Use this property along with the value `capitalize` , `uppercase` , or `lowercase` to transform the way the selected text is capitalized.

Let's apply some of these styles to the span from our previous example to see them in action.

```
text-decoration: underline; text-shadow: 5px 5px 2px gray; text-transform: lowercase;
```

If we add these three rules to our span styling rules we get the following result.

.styled_p_3 { font: 400 1.2em/1.5em "Open Sans", Arial, sans-serif; } .styled_span_3 { font: italic small-caps 600 2em/1.5em "Open Sans", Arial, sans-serif; text-decoration: underline; text-shadow: 5px 5px 2px gray; text-transform: lowercase; }

This is Styled Span 2
This is styled paragraph 2. We styled this span and paragraph by declaring all of the font properties using the font shorthand property. As you can see, it looks just like span and paragraph 1.

Initial & Inherit

Initial and inherit are two values that can be applied to text styling CSS properties. There is a lot of understandable confusion surrounding these terms. Here's what they mean:

- Initial is reset a property to the value included in the official CSS definition of that property. Think of initial as a hard-reset button.
- Inherit is used to allow styles to cascade down from a parent element to a child element.

Adjusting Text Layout and Flow

When you use a word processing application to edit text you have several different alignment options: left, right, center, and justify. Those same options are available when adding text to a web page by using the `text-align` CSS property. You can also add an automatic indentation to paragraphs of text with the `text-indent` property by declaring any length value in pixels, points, or ems. We can show the `text-indent` and `text-align` properties in action by adding them to the paragraph of text from the previous example. Here's the CSS we'll add to our stylesheet:

```
/*We'll apply these rules to the paragraph*/   text-align: justify;   text-indent: 3em;  /*We'll apply this ru
le to the span*/   display: block; /*To force the span to fill the full width*/   text-align: center;   text-i
ndent: 0; /*To remove the indent applied to the paragraph*/
```

Now when we render the paragraph of text in our browser we can see these styles take effect.

.styled_p_4 { font: 400 1.2em/1.5em “Open Sans”, Arial, sans-serif; text-align: justify; text-indent: 3em; } .styled_span_4 { font: italic small-caps 600 2em/1.5em “Open Sans”, Arial, sans-serif; text-decoration: underline; text-shadow: 5px 5px 2px gray; text-transform: lowercase; display: block; text-align: center; text-indent: 0; }

This is Styled Span 2

This is styled paragraph 2. We styled this span and paragraph by declaring all of the font properties using the font shorthand property. As you can see, it looks just like span and paragraph 1.

Styling Lists

All of the styling techniques we've covered so far can also be applied to text that appears in a table or lists. However, there is one styling technique unique to lists: list marker styling. List markers can be styled using the `list-style` property. This shorthand property can be used to simultaneously declare an image to use as a marker, a backup marker type that will be displayed if the image is unavailable, and the position of the marker. These values must be specified in this order: type | position | image. You can learn more about this property by reviewing our lists tutorial.

Next Steps

We've covered a lot of ground in this tutorial. Working with fonts and designing typography for the web is not an easy task to master. However, it's worth taking the time to learn about fonts and typography since getting them right is critical to the success of your website's content. If you haven't done so already, a good next step would be to review our CSS tutorial. In it, you will learn the fundamental concepts behind this important fundamental web technology and be better prepared to apply the concepts and techniques described in this tutorial.

Jon Penland
Jon is a freelance writer, travel enthusiast, husband and father. He writes about web technologies such as WordPress, HTML, and CSS.

Related Elements

Element Name	Attributes	Notes
<summary> HTML Tag		The <summary> element is used as a child of <details> element to provide a summary of the contents of the <details> element. At this time, <summary> is not well supported across browsers.
q		The <q> element is used to identify and inline quote that does not require paragraph breaks. Longer quotations

		that do require paragraph breaks should use the <blockquote> element.
mark		The <mark> element is used to highlight text inside of another element such as a paragraph, list, or table. Text to which the <mark> element has been added is considered to be particularly relevant in a specific context.
bdo		The <bdo> element is used override the default directionality of text. It is used to display characters from languages that are read from right-to-left, such as Hebrew and Arabic.
bdi		The <bdi> element is used to isolate a small section of text which may be formatted to run in the opposite direction than the text around it (such as right-to-left in a left-to-right context). This is useful when a language with right-to-left directionality, such as Arabic or Hebrew, is used inline with left-to-right languages.
<sup> HTML Tag		
del		The <del> tag is used to identify text that has been deleted from a document but retained to show the history of modifications made to the document. Pair a <del> element with an <ins> element to identify the inserted text that replaced the deleted text.
xmp		The <xmp> element was used to surround HTML example text that should be rendered without interpreting any HTML elements between the opening and closing <xmp> tags. The element was deprecated in HTML 3.2 and is now obsolete.

<tt> HTML Tag		The <tt> element was used to identify text to be displayed using the browser's default monospace or fixed-width font as it would appear on a fixed-width device such as a teletype. This element has been deprecated and the <code> element is an appropriate modern replacement for <tt>.
<u> HTML Tag		The <u> element was originally used to identify text that should be underlined. The element was deprecated in HTML 4.01, but in HTML5 it was redefined to represent text that should be displayed in a way that is an unarticulated but stylistically distinct from the surrounding text. For example, one proper use of the <u> element is to identify misspelled terms.
<var> HTML Tag		The <var> element is used to identify a variable in a mathematical equation or computer program. Text marked with <var> tags is displayed in an italics font style by most browsers.
<wbr> HTML Tag		The <wbr> element is used to define a word break opportunity in a string of text. It is particularly useful when you wish to define word break opportunities in a long unbroken string of text that might otherwise break improperly.
kbd		The <kbd> element is used to identify text that represents user keyboard input. Text surrounded by <kbd> tags is typically displayed in the browser's default monospace font.
nobr		The <nobr> element identifies text that should not be allowed to break into multiple lines which can force users to scroll horizontally to view the content. This element is obsolete and should be used.

p	clear	The <p> element is used to identify blocks of paragraph text. The closing <p> tag is optional and is implied by the opening tag of the next HTML element encountered in an HTML document after an opening <p> tag.
pre		The <pre> element is used to identify text that should be rendered with all line breaks and spaces intact. It is often used to preserve indenting and line breaks when displaying code blocks.
s		The <s> element is used to identify text that is no longer accurate or relevant. It is similar to, but semantically distinct from, the <del> element which is used to identify document edits. By default, browsers render the contents of an <s> element with a strikethrough.
small		The <small> element identifies text to display one size smaller than the surrounding text. In HTML5 the element is intended to be used to identify items of secondary importance such as copyright notices, side comments, and legal notices.
Proper Use Of The Strong Element In HTML (Plus Code Example)		The element is used to identify text that is of greater importance than the surrounding text. By default, all browsers render text in a bold typeface.
How To Use To Add CSS Style Rules To HTML Documents	<style type=""> <style media="">	The <style> element is used to add CSS style rules to an HTML document. The element is expected to appear in the document <head>, but will also render acceptably when used in the <body> of the document.
<sub> HTML Tag		The <sub> element is used to identify characters that should be rendered in a subscript position. The element

		should be used mark text according to typographical conventions and not stylistic purposes. Text that is to appear subscript for purely stylistic purposes should be styled with CSS.
b		The <b> element is used to draw attention to enclosed text without implying any added importance or emphasis. Text surrounded by <b> tags is displayed with a bold typeface.
basefont	color face size	The <basefont> element was used to set the default font size for an HTML document. Deprecated in HTML 4.01 and removed entirely from HTML5, <basefont> is not supported by modern browsers and font styling should be controlled with CSS.
big		The <big> element was used to cause the selected text to appear one size larger than the surrounding text. This purely presentational tag was removed from HTML5 and should not be used. Instead, use CSS to control font size.
blockquote		The <blockquote> element defines a block of text that is a direct quotation. The <quote> element should be used when a quotation is presented inline with the surrounding text, but when the quotation is presented as a separate paragraph, <blockquote> is the appropriate element to use to identify the quotation.
br	clear	The element is used to insert a line break or carriage-return within a parent element such as a paragraph without breaking out of the parent container.
code		The <code> element is used to define enclosed text as computer code. It is often paired with the <pre> element to preserve line breaks and

		indentation when presenting blocks of computer code.
em		The element is used to indicate text that should receive greater emphasis than the surrounding text.
font	color face pointsize size weight	The <font> element was used to specify typographical display styles. It has been deprecated and fonts should be styled with CSS instead.
headlines	align	The <h1>, <h2>, <h3>, <h4>, <h5>, and <h6> elements are used to create headings in descending order of importance where <h1> is the most important and <h6> the least.
hr	width size noshade color align	The <hr> element is used to represent a thematic break between paragraph-level elements. It is typically rendered as a horizontal line.
i		The <i> element is used to differentiate words from the surrounding text by styling the marked text in italics without implying any added emphasis to the italicized words.
ins		The <ins> element is used to identify text that has been inserted into a document. It is often paired with a <del> element which identifies deleted text replaced by the text contained in the <ins> element.