

12

FORMATTING TEXT

OVERVIEW

- **Font-related properties**
- **Text line settings**
- **Various text effects**
- **List style properties**
- **ID, class, and descendent selectors**
- **Specificity**

Designing Text

Styling text on the web is tricky because you don't have control over how the text displays for the user:

- They may not have the font you specify.
- They may have their text set larger or smaller than you designed it.

Best practices allow for flexibility in text specification.

Typesetting Terminology

- A **typeface** is a set of characters with a single design (example: Garamond).
- A **font** is a particular variation of the typeface with a specific weight, slant, or ornamentation (example: Garamond Bold Italic).
- In traditional metal type, each size was a separate font (example: 12-point Garamond Bold Italic).
- On a computer, fonts are generally stored in individual font files.

CSS Basic Font Properties

CSS font properties deal with specifying the shapes of the characters themselves:

- `font-family`
- `font-size`
- `font-weight`
- `font-style`
- `font-variant`
- `font` (a shorthand that includes settings for all of the above)

Specifying the Font Family

font-family

Values: One or more font family names, separated by commas

Example:

```
body { font-family: Arial; }  
var { font-family: Courier, monospace; }
```

Specifying the Font Family (cont'd)

- Font names must be capitalized (except generic font families).
- Use commas to separate multiple font names.
- If the name has a character space, it must appear within quotation marks:

```
p { font-family: "Duru Sans", Verdana, sans-serif; }
```

Using Fonts on the Web

- The font must be available on the user's machine for it to display.
- The best practice is to provide a list of options. The browser uses the first one that is available.
- Start with the font you want and then provide backup options ending with a generic font family, as shown here:

```
p { font-family: "Duru Sans", Verdana, sans-serif; }
```
- You can also download a web font with the page, but it adds to the download and display time.

Generic Font Families

- Generic font families instruct the browser to use an available font from one of five stylistic categories:
serif, sans-serif, monospace, cursive, fantasy
- Generic font families are often used as the last backup option.

Generic Font Families (cont'd)

serif



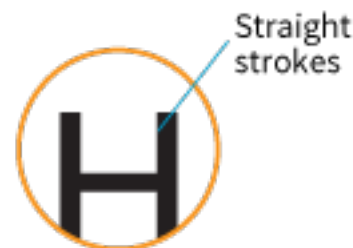
Hello
Times

Hello
Georgia

Hello
Times New Roman

Hello
Lucida

sans-serif



Hello
Verdana

Hello
Trebuchet MS

Hello
Arial

Hello
Arial Black

monospace

W i

Monospace font
(equal widths)

W i

Proportional font
(different widths)

Hello
Courier

Hello
Courier New

Hello
Andale Mono

cursive

Hello
Apple Chancery

Hello
Comic Sans

Hello
Snell

fantasy

Hello
Impact

HELLO
Stencil

HELLO
Mojo

Specifying Font Size

font-size

Values:

- CSS length units
- Percentage value
- Absolute keywords (`xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`)
- Relative keywords (`larger`, `smaller`)

Example:

```
h1 { font-size: 2.5rem; }
```

Specifying Font Size (cont'd)

Most common sizing methods:

- **rem** and **em** units
- **percentages** (based on the inherited font size for that element)
- **pixels** (px) can be used, but they're not flexible.

Font Size: rem Units

- The **rem** (**root em**) is equal to the font size of the **html** (root) element.
- In browsers, the **default** root size is 16 pixels, so:
1 rem = 16 pixels.
- If the font size of the root is changed, rem measurements change too.
- !!! Old browsers *do not* support rem units (IE8 and earlier).

Font Size: em Units

- The **em unit** is based on the current font size of the element.
- The default font size is 16 pixels. **By default, 1em = 16 pixels.**
- But if you change the font size of the element, the size of its em unit changes too.
- Ems may be different sizes in different parts of the document and may compound larger or smaller when elements are nested.
- This makes ems a little tricky to use, although they are better supported than rem units.

Font Weight (Boldness)

`font-weight`

Values: `normal`, `bold`, `bolder`, `lighter`, `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, `900`

Example:

```
h1 { font-weight: normal; }  
span.new { font-weight: bold; }
```

- Most common values are `normal` and `bold`.
- Numerical values are useful when using a font with multiple weights.

Font Style (Italics)

font-style

Values: normal, italic, oblique

Example:

```
cite { font-style: italic; }
```

- Makes text italic, normal, or oblique (slanted, but generally the same as italics).

Small Caps

font-variant

Values (in CSS2.1): normal, small-caps

Example:

```
abbr { font-variant: small-caps; }
```

- Small caps are a separate font design that uses small uppercase characters in place of lowercase letters.
- They help acronyms and other strings of capital letters blend in with the weight of the surrounding text.

Condensed and Extended Text

`font-stretch`

Values (in CSS2.1): normal, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded

Example:

```
abbr { font-variant: small-caps; }
```

- Tells the browser to select a normal, condensed, or extended font variation from a typeface if it is available

Design

Universe Ultra Condensed

Design

Universe Condensed

Design

Univers

Design

Universe Extended

The Shortcut font Property

font

Values (in CSS2.1): A list of values for all the individual properties, in this order:

```
{font: style weight stretch variant size/line-height font-family}
```

At minimum, it must contain **font-size** and **font-family**, in that order. Other values are optional and may appear in any order prior to **font-size**.

Example:

```
p { font: 1em sans-serif; }
```

```
h3 { font: oblique bold small-caps 1.5em Verdana, sans-serif; }
```

Advanced Typography

The **CSS3 Font Module** offers properties for fine-tuned typography control, including:

- Ligatures
- Superscript and subscript
- Alternate characters (such as a swash design for an *S*)
- Proportional font sizing using x-height
- Kerning
- OpenType font features

Text Line Treatments

Some properties control whole lines of text:

- Line height (`line-height`)
- Indents (`text-indent`)
- Horizontal alignment (`text-align`)

Line Height

line-height

Values: *Number, length, percentage, normal*

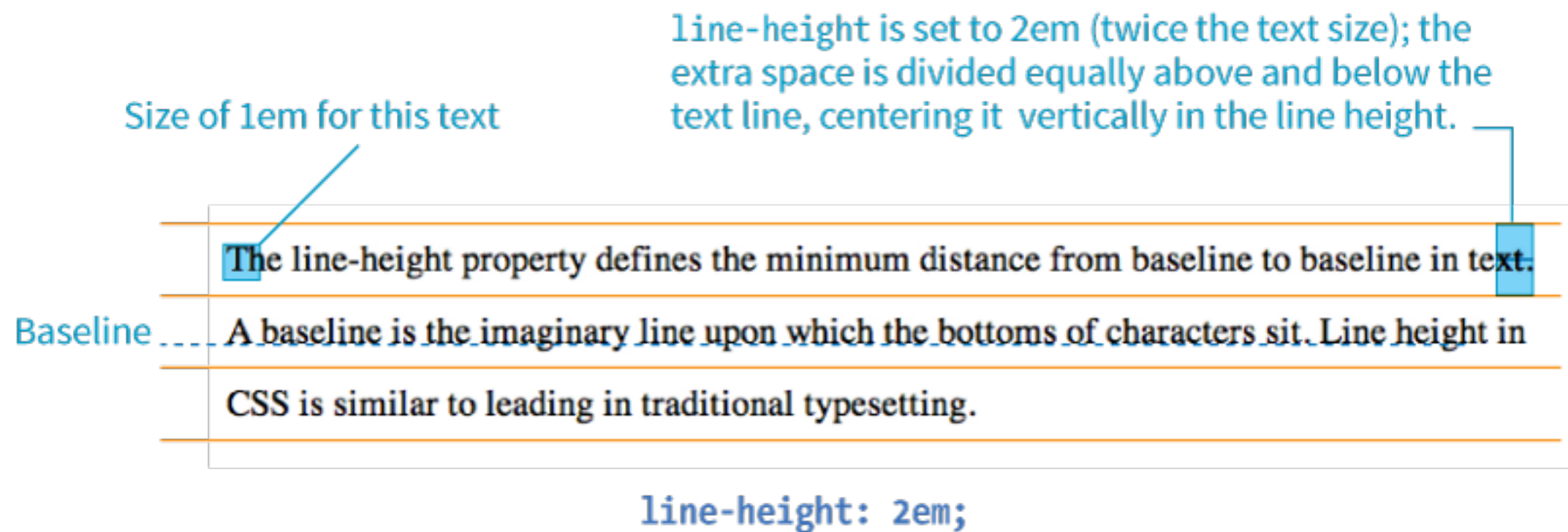
Example:

```
p { line-height: 1.4em; }
```

- Line height defines the minimum distance from baseline to baseline in text.

Line Height (cont'd.)

- The **baseline** is the imaginary line upon which the bottoms of characters sit.
- If a large character or image is on a line, the line height expands to accommodate it.




Indents

`text-indent`

Values: *Length, percentage*

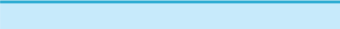
Examples:

```
p {text-indent: 2em;}
```




Paragraph 1. The text-indent property indents only the first line of text by a specified amount. You can specify a length measurement or a percentage value.

```
p {text-indent: 25%;}
```



Paragraph 2. The text-indent property indents only the first line of text by a specified amount. You can specify a length measurement or a percentage value.

```
p {text-indent: -35px;}
```



Paragraph 3. The text-indent property indents only the first line of text by a specified amount. You can specify a length measurement or a percentage value.

Horizontal Text Alignment

`text-align`

Values: left, right, center, justify, start, end

Examples:

`text-align: left;`

Paragraph 1. The text-align property controls the horizontal alignment of the text within an element. It does not affect the alignment of the element on the page. The resulting text behavior of the various values should be fairly intuitive.

`text-align: right;`

Paragraph 2. The text-align property controls the horizontal alignment of the text within an element. It does not affect the alignment of the element on the page. The resulting text behavior of the various values should be fairly intuitive.

`text-align: center;`

Paragraph 3. The text-align property controls the horizontal alignment of the text within an element. It does not affect the alignment of the element on the page. The resulting text behavior of the various values should be fairly intuitive.

`text-align: justify;`

Paragraph 4. The text-align property controls the horizontal alignment of the text within an element. It does not affect the alignment of the element on the page. The resulting text behavior of the various values should be fairly intuitive.

Underlines (Text Decoration)

`text-decoration`

Values: none, underline, overline, line-through, blink

Examples: I've got laser eyes.

```
text-decoration: underline;
```

I've got laser eyes.

```
text-decoration: overline;
```

~~I've got laser eyes.~~

```
text-decoration: line-through;
```

NOTE:

`text-decoration` is often used to turn **off** underlines under links:

```
a {  
    text-decoration: none;  
}
```

Text Decoration Tips

- If you turn off underlines under links, be sure there is another visual cue to compensate.
- Underlining text that is not a link may be misleading. Consider italics instead.
- Don't use `blink`. Browsers don't support it anyway.

Capitalization

`text-transform`

Values:

none, capitalize, lowercase, uppercase, full-width

Examples:

`text-transform: none;`
(as it was typed in the source)

And I know what you're thinking.

`text-transform: capitalize;`

And I Know What You're Thinking.

`text-transform: lowercase;`

and i know what you're thinking.

`text-transform: uppercase;`

AND I KNOW WHAT YOU'RE THINKING.

Spacing

letter-spacing

Values: *length*, normal

word-spacing

Values: *length*, normal

Examples:

B l a c k G o o s e B i s t r o S u m m e r M e n u

```
p { letter-spacing: 8px; }
```

Black Goose Bistro Summer Menu

```
p { word-spacing: 1.5em; }
```

Text Shadow

`text-shadow`

Values: *'horizontal-offset' 'vertical-offset' 'blur-radius' 'color', none*

The value is two offset measurements, an optional blur radius, and a color value (with no commas between).

Example:

The text "The Jenville Show" is displayed in a bold, green, sans-serif font. It has a subtle silver shadow applied, which is barely visible against the white background.

```
text-shadow: .2em .2em .1em silver;
```

The text "The Jenville Show" is displayed in a bold, green, sans-serif font. It has a more pronounced silver shadow applied, which is clearly visible and adds depth to the text.

```
text-shadow: .2em .2em .3em silver;
```

List Style Properties

There are three properties for affecting the display of lists:

- **`list-style-type`**
Chooses the type of list marker
- **`list-style-position`**
Sets the position of the marker relative to the list element box
- **`list-style-image`**
Allows you to specify your own image for use as a bullet

LIST STYLES

Choosing a Marker

`list-style-type`

Values:

`none`, `disc`, `circle`, `square`, `decimal`, `decimal-leading-zero`, `lower-alpha`, `upper-alpha`, `lower-latin`, `upper-latin`, `lower-roman`, `upper-roman`, `lower-greek`

Unordered lists: `ul { list-style-type: keyword; }`

disc	circle	square
<ul style="list-style-type: none">• radish• avocado• pomegranite• cucumber• persimmon	<ul style="list-style-type: none">○ radish○ avocado○ pomegranite○ cucumber○ persimmon	<ul style="list-style-type: none">■ radish■ avocado■ pomegranite■ cucumber■ persimmon

LIST STYLES

Choosing a Marker (cont'd)

Ordered lists: `ol { list-style-type: keyword; }`

Keyword	System
decimal	1, 2, 3, 4, 5...
decimal-leading-zero	01, 02, 03, 04, 05...
lower-alpha	a, b, c, d, e...
upper-alpha	A, B, C, D, E...
lower-latin	a, b, c, d, e... (same as lower-alpha)
upper-latin	A, B, C, D, E... (same as upper-alpha)
lower-roman	i, ii, iii, iv, v...
upper-roman	I, II, III, IV, V...
lower-greek	α, β, γ, δ, ε...

LIST STYLES

Marker Position

`list-style-position`

Values: `inside`, `outside`, `hanging`

Positions the marker relative to the content area:

`outside`

- **Radish.** Praesent in lacinia risus. Morbi urna ipsum, efficitur id erat pellentesque, tincidunt commodo sem. Phasellus est velit, porttitor vel dignissim vitae, commodo ut urna.
- **Avocado.** Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur lacinia accumsan est, ut malesuada lorem consectetur eu.

`inside`

- **Radish.** Praesent in lacinia risus. Morbi urna ipsum, efficitur id erat pellentesque, tincidunt commodo sem. Phasellus est velit, porttitor vel dignissim vitae, commodo ut urna.
- **Avocado.** Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur lacinia accumsan est, ut malesuada lorem consectetur eu.

LIST STYLES




Custom Bullets

`list-style-image`

Values: `url(location)`, `none`

Example:

```
ul {  
    list-style-type: disc;  
    list-style-image: url(/images/rainbow.gif);  
    list-style-position: outside;  
}
```

-  Puppy dogs
-  Sugar frogs
-  Kitten's baby teeth

More Selector Types

- Descendent selectors
- ID selectors
- Class selectors
- Universal selector

Descendent Selectors

A **descendent selector** targets elements contained in another element.

It's a kind of **contextual selector** (it selects based on relationship to another element).

It's indicated in **a list separated by a character space**.

```
ol a {font-weight: bold;}
```

(only the links (**a**) in ordered lists (**ol**) would be bold)

```
h1 em {color: red;}
```

(only emphasized text in h1s would be red)

Descendent Selectors (cont'd)

They can appear as part of a grouped selector:

```
h1 em, h2 em, h3 em {color: red;}
```

(only emphasized text in h1, h2, and h3 elements)

They can be several layers deep:

```
ol a em {font-variant: small-caps;}
```

(only emphasized text in links in ordered lists)

ID Selectors

ID selectors (indicated by a **#** symbol) target elements based on the value of their ID attributes:

```
<li id="primary">Primary color t-shirt</li>
```

To target just that item:

```
li#primary {color: olive;}
```

To omit the element name:

```
#primary {color: olive;}
```

It can be used as part of a compound or contextual selector:

```
#intro a { text-decoration: none;}
```

Class Selectors

Class selectors (indicated by a . symbol) select elements based on the value of their class attributes:

```
p.special { color: orange; }
```

(All paragraphs with the class name "special" would be orange.)

To target *all* element types that share a class name, omit the element name in the selector:

```
.highlight { background-color: yellow; }
```

(All elements with the class "highlight" would have a yellow background.)

Universal Selector

The **universal element selector** (*) matches any element, like a wildcard in programming languages:

```
* {border: 1px solid gray;}
```

(puts a 1-pixel gray border around every element in the document)

Can be used as part of contextual selectors:

```
#intro * {border: 1px solid gray;}
```

(selects all elements contained within an element with the ID **intro**)

Specificity Basics

Specificity refers to a system for sorting out which selectors have more weight when resolving style rule conflicts.

More specific selectors have more weight.

In simplified terms, it works like this:

- **Inline styles** with the `style` attribute are more specific than (and will override...)
- **ID selectors**, which are more specific than (and will override...)
- **Class selectors**, which are more specific than (and will override...)
- **Individual element selectors**

Calculating Specificity

There is a system used to calculate specificity. Start by drawing three boxes:

[] [] []

For each style rule:

1. Count the **IDs** in the selector and put that number in the first box.
2. Count the **class** and pseudo-class selectors and put the number in the second box.
3. Count the **element** names and put the number in the third box

[ID] [class] [elements]

4. The first box that is not a tie determines which selector wins.

Calculating Specificity (cont'd)

Example:

`h1 { color: red; }` [0] [0] [1]

`h1.special { color: lime; }` [0] [1] [1]

The second one has a class selector and the first one doesn't, therefore the second one is more specific and has more weight.

The lime color applies to `h1`s when they have the class name "special."

Using Specificity

Use specificity strategically to take advantage of overrides:

```
p { line-height: 1.2em; } [0] [0] [1]
```

(sets the line-height for all paragraphs)

```
blockquote p { line-height: 1em; } [0] [0] [2]
```

(more specific selector changes line-height when the paragraph is in a blockquote)

```
p.intro { line-height: 2em; } [0] [1] [1]
```

(paragraphs with the class “intro” have a line-height of 2em, even when they’re in a blockquote. A class name in the selector has more weight than two element names.)