Fundamentals of Web Development

Third Edition by Randy Connolly and Ricardo Hoar

Chapter 4

CSS 1: Selectors and Basic Styling

In this chapter you will learn . . .

- The rationale for CSS
- The syntax of CSS
- Where CSS styles can be located
- The different types of CSS selectors
- What the CSS cascade is and how it works
- The CSS box model
- CSS text styling

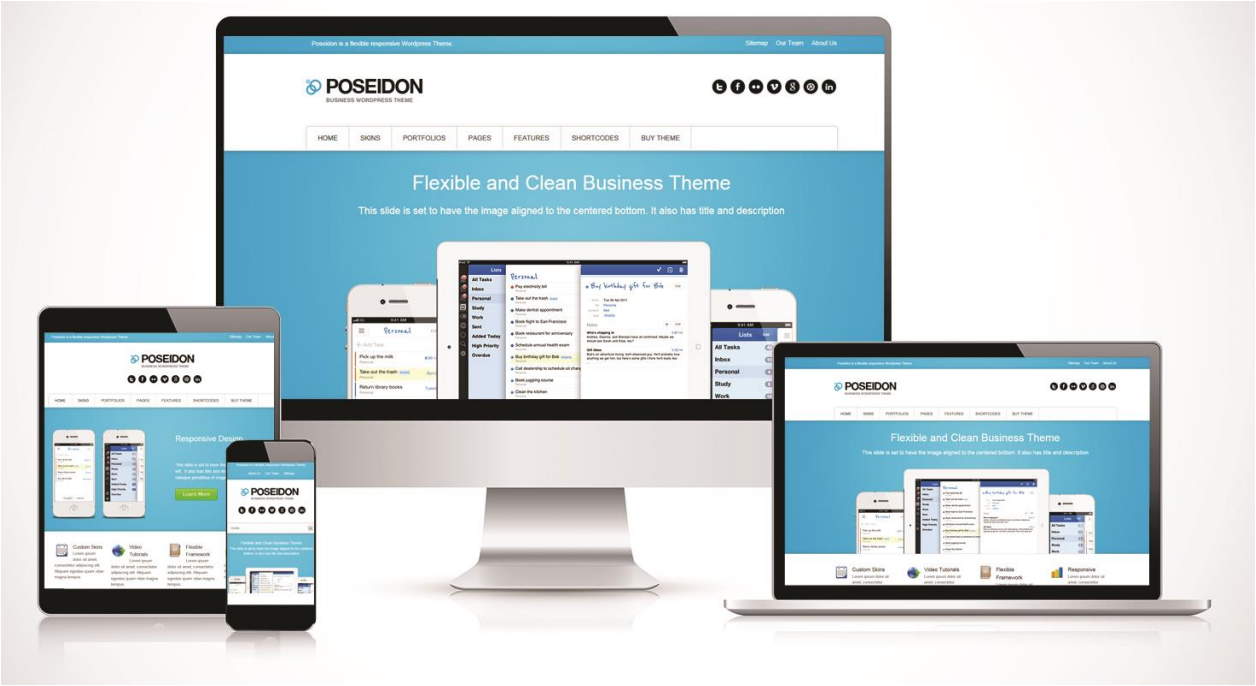
What Is CSS?

- CSS is a W3C standard for describing the appearance of HTML elements.
- With CSS, we can assign font properties, colors, sizes, borders, background images, positioning and even animate elements on the page.
- CSS can be added directly to any HTML element (via the style attribute), within the <head> element, or, most commonly, in a separate text file that contains only CSS.

Benefits of CSS

- **Improved control over formatting.**
- **Improved site maintainability.** All formatting can be centralized into one CSS file
- **Improved accessibility.** By keeping presentation out of the HTML, screen readers and other accessibility tools work better.
- **Improved page-download speed.** Each individual HTML file will contain less style information and markup, and thus be smaller.
- **Improved output flexibility.** CSS can be used to adopt a page for different output media. This approach to CSS page design is often referred to as **responsive design**.

CSS allows Responsive Design



CSS Versions

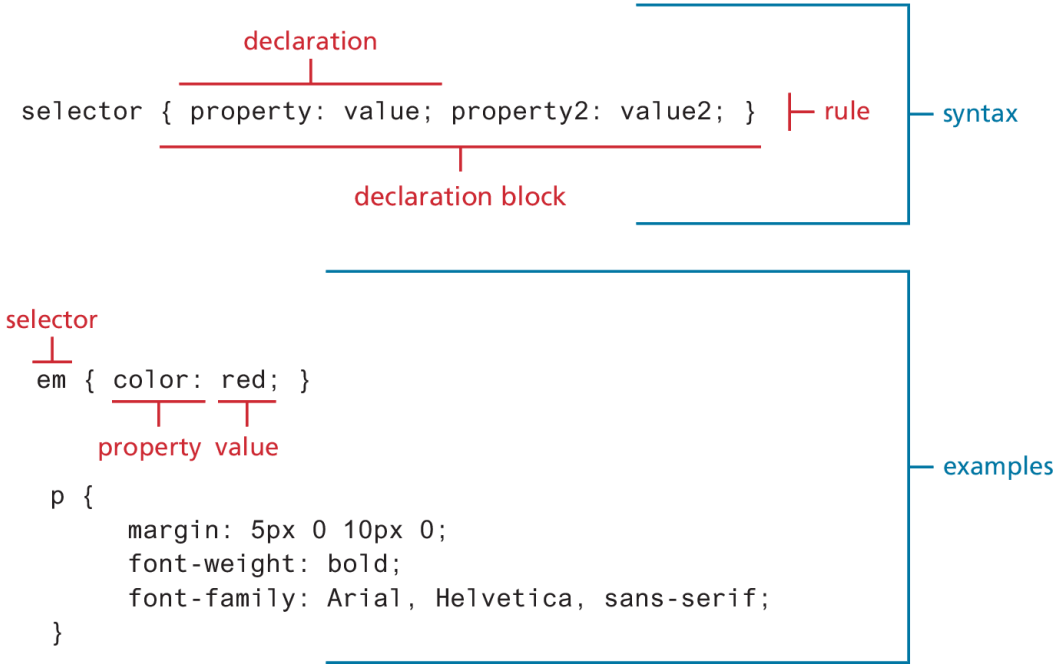
- Style sheets as a way to visually format markup predate the web.
- W3C decided to adopt CSS, and by the end of 1996, the CSS Level 1 Recommendation was published
- A year later, the CSS2 was published
- CSS2.1, did not become an official W3C Recommendation until June 2011
- At the same time the CSS2.1 standard was being worked on, a different group at the W3C was working on a CSS3 draft

Browser Adoption

- Perhaps the most important thing to keep in mind with CSS is that the different browsers have not always kept up with the W3C.
- For this reason, CSS has a reputation for being a somewhat frustrating language.
- In this book, we hope to redress that negative reputation by covering CSS basics and then incrementally introducing ideas until finally we cover modern frameworks that address many of those challenges.

CSS Syntax

- A CSS document consists of one or more **style rules**.
- A rule consists of a **selector** that identifies the HTML element or elements that will be affected, followed by a series of **property:value pairs** called **the declaration**
- The series of declarations is also called the **declaration block**.

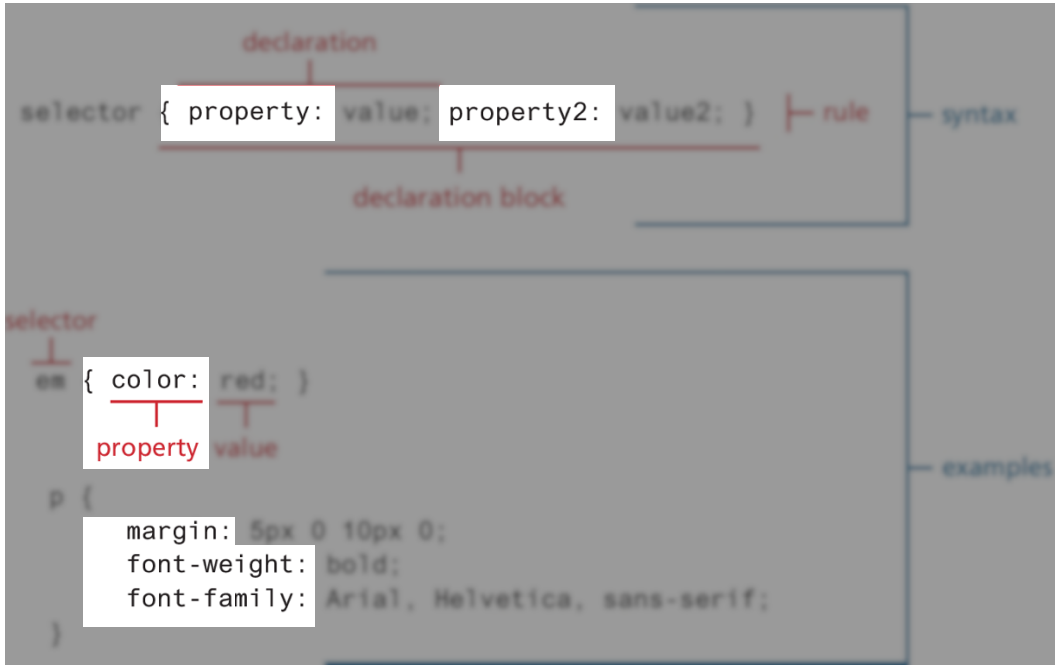


CSS Syntax: Selector

- Every CSS rule begins with a **selector**.
- The **selector** identifies which element or elements in the HTML document will be affected by the declarations in the rule.

CSS Syntax: Properties

- Each individual CSS declaration must contain a **property**
- The CSS2.1 recommendation defines over a hundred different property names!
- Table 4.1 lists many of the most commonly used CSS properties.



CSS Syntax: Values

- Each individual CSS declaration must also contain a **value**
- The unit of any given value is dependent upon the property.
- Some property values are from a predefined list of keywords. Others are values such as length measurements, percentages, numbers without units, color values, and URLs.

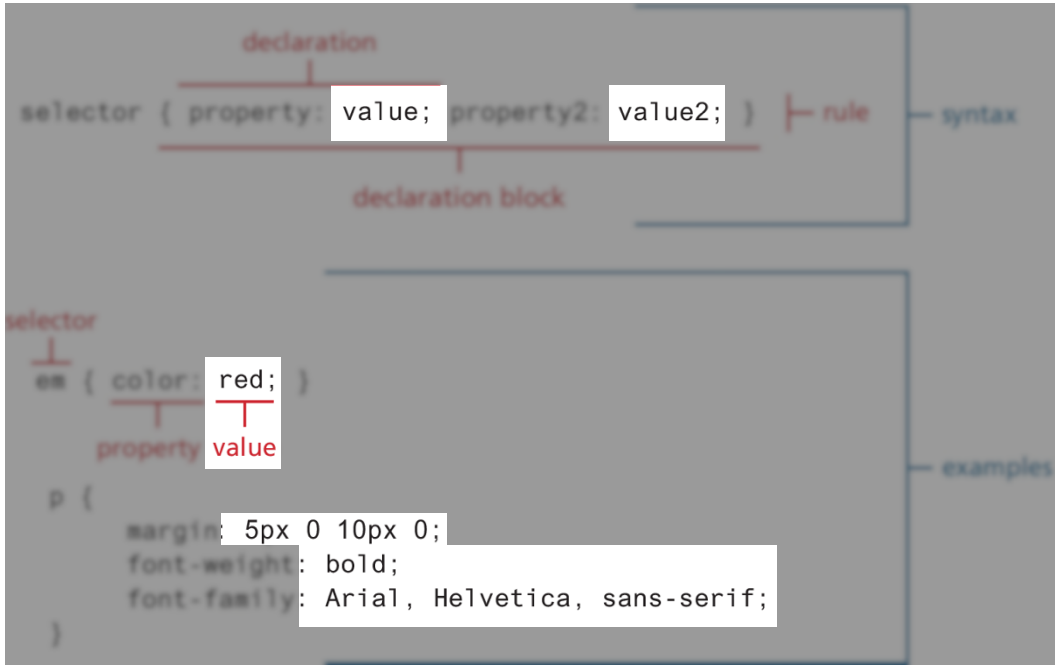


TABLE 4.2 Color Values

Method	Description	Example
Name	Use one of 17 standard color names. CSS3 has 140 standard names.	color: red; color: hotpink; /* CSS3 only */
RGB	Uses three different numbers between 0 and 255 to describe the red, green, and blue values of the color.	color: rgb(255,0,0); color: rgb(255,105,180);
Hexadecimal	Uses a six-digit hexadecimal number to describe the red, green, and blue value of the color	color: #FF0000; color: #FF69B4;
RGBa	This defines a partially transparent background color. The “a” stands for “alpha,” which is a term used to identify a transparency	color: rgba(255,0,0,0.5);
HSL	Allows you to specify a color using Hue Saturation and Light values. This is available only in CSS3. HSLA is also available as well.	color: hsl(0,100%,100%); color: hsla(330,59%,100%,0.5);

Common Units of Measure Values

Units of measure in CSS are either

- **relative units**, in that they are based on the value of something else, or
- **absolute units**, in that they have a real-world size.

Some example measures (See Table 4.3 for more**)**

- **px** Pixel. In CSS2 this is a relative measure, while in CSS3 it is absolute (1/96 of an inch)
- **em** Equal to the computed value of the font-size property of the element on which it is used.
- % A measure that is always relative to another value.
- **in** Inches
- **cm** Centimeters

CSS Comments

NOTE

It is helpful to add comments to your style sheets. Comments take the form:
/ comment goes here */*

It is a common practice to locate related style rules together, and then indicate that they are related via a comment. For instance:

```
/* main navigation */
nav#main { … }
nav#main ul { … }
nav#main ul li { … }

/* header */
header { … }
h1 { … }
```

Comments can also be a helpful way to temporarily hide any number of rules, to help with debugging.

Location of Styles

CSS style rules can be located in three different locations.

- Inline Styles
- Embedded Style Sheet
- External Style Sheet

These three are not mutually exclusive, in that you could place your style rules in all three.

Inline Styles

Inline styles are style rules placed within an HTML element via the style attribute

- An inline style only affects the element it is defined within and overrides any other style definitions for properties used
- Notice that a selector is not necessary

```
<h1>Share Your Travels</h1>
<h2 style="font-size: 24pt">Description</h2>
…
<h2 style="font-size: 24pt; font-weight: bold;">Reviews</h2>
```

LISTING 4.1 Inline styles example

Embedded Style Sheet

Embedded style sheets (also called internal styles) are style rules placed within the <style> element (inside the <head> element of an HTML document)

- While better than inline styles, using embedded styles is also by and large discouraged.

```
<head>
   <meta charset="utf-8">
   <title>Chapter 4</title>
   <style>
                h1 { font-size: 24pt; }
                h2 {
                  font-size: 18pt;
                  font-weight: bold;
                }
   </style>
   </head>
<body>
```

LISTING 4.2 Embedded styles example

External Style Sheet

External style sheets are style rules placed within an external text file with the **.css** extension.

- When you change an external style sheet, all HTML documents that reference that style sheet will automatically use the updated version.
- The browser is able to cache the external style sheet, which can improve performance as well.

```
<head>
   <meta charset="utf-8">
   <title>Chapter 4</title>
   <link rel="stylesheet" href="styles.css" />
</head>
```

LISTING 4.3 Referencing an external style sheet

Selectors

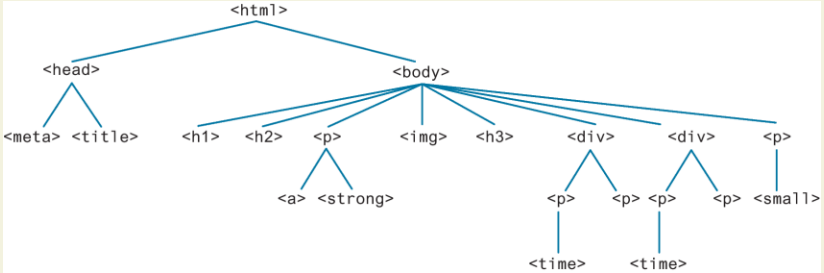
When defining CSS rules, you will need to use a selector.

- Selectors tell the browser which elements will be affected by the property values
- Selectors allow you to select individual or multiple HTML elements.
- Three basic selector types have been around since the earliest CSS2 specification.
  - Element Selectors
  - Class Selectors
  - Id Selectors

The Document Object Model (DOM)

NOTE

The **Document Object Model** (DOM) is how a browser represents an HTML page internally. This DOM is akin to a tree representing the overall hierarchical structure of the document.



In the last chapter, we learned some of the familial terminologies (such as descendants, ancestors, siblings, etc.) in an HTML document. You will at times have to think about the elements in your HTML document in terms of their position within the hierarchy.

Element Selectors

Element selectors select all instances of a given HTML element.

You can also select all elements by using the **universal element selector** (*)

You can select a group of elements by separating the different element names with commas.

```
/* commas allow you to group selectors */
p, div, aside {
  margin: 0;
  padding: 0;
}
/* the above single grouped selector is equivalent to the following: */
p {
  margin: 0;
  padding: 0;
}
div {
  margin: 0;
  padding: 0;
}
aside {
  margin: 0;
  padding: 0;
}
```

LISTING 4.4 Sample grouped selector

Class and ID Selectors

A **class selector** allows you to simultaneously target different HTML elements regardless of their position in the document tree.

HTML elements labeled with the same class attribute value, can be targeted for styling by using a class selector, which takes the form:

period (.) followed by the class name.

An **ID selector** allows you to target a specific element by its id attribute regardless of its type or position in the document tree.

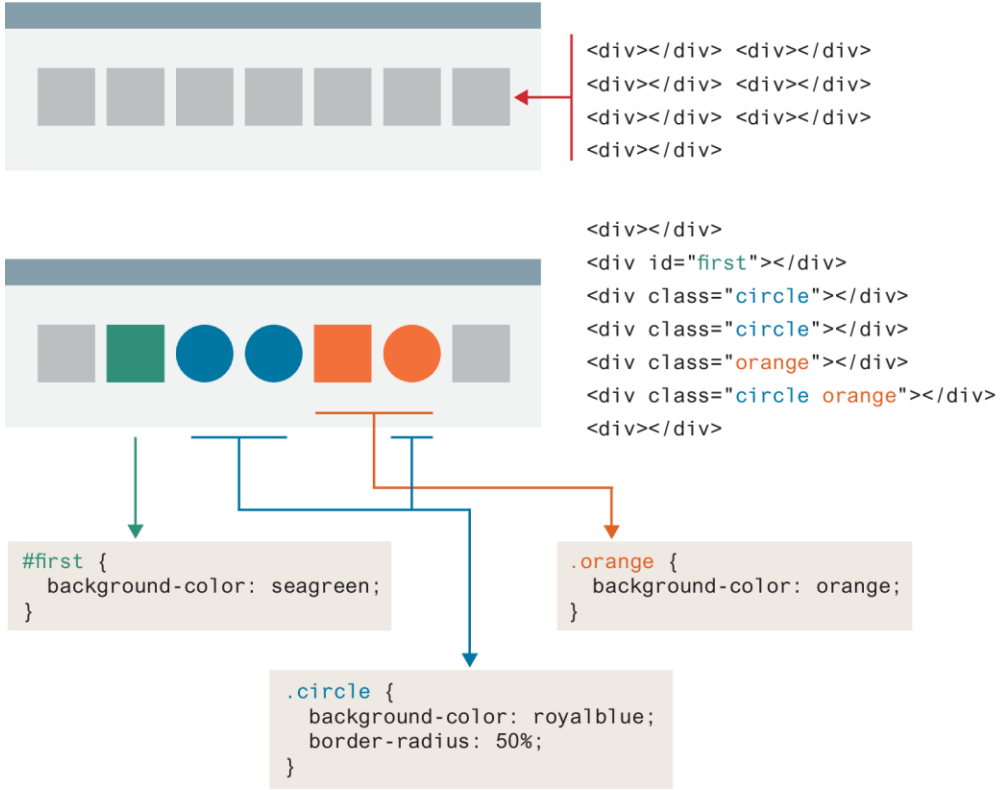
HTML elements labeled with an id attribute, can be targeted for styling by using an id selector, which takes the form:

pound/hash (#) followed by the id name.

Class and ID Selector Example

Consider Figure 4.4. The original HTML can be modified to add class and id values, which are then styled in CSS.

- Id selector #**first** matches the div with id “first”
- Class selectors .**orange** and .**circle**, match all divs with those class values.
- Notice that an element can be tagged with multiple classes.



Attribute Selectors

An **attribute selector** provides a way to select HTML elements either by the presence of an element attribute or by the value of an attribute.

- Attribute selectors can be helpful in the styling of hyperlinks and form elements (which come up in the next chapter)
- These selectors can be combined with the element id and class selectors.
- You use square brackets to create attribute selectors (see Table 4.3)

TABLE 4.4 Attribute Selectors

	Matches	Example
[ ]	A specific attribute.	**[title]** Matches any element with a title attribute
[=]	A specific attribute with a specific value.	**a[title="posts from this country"]** Matches any <a> element whose title attribute is exactly “posts from this country”
[~=]	A specific attribute whose value matches at least one of the words in a space-delimited list of words.	**[title~="Countries"]** Matches any title attribute that contains the word “Countries“
[^=]	A specific attribute whose value begins with a specified value.	**a[href^="mailto"]** Matches any <a> element whose href attribute begins with “mailto“
[*=]	A specific attribute whose value contains a substring.	**img[src*="flag"]** Matches any <img> element whose src attribute contains somewhere within it the text “flag“
[$=]	A specific attribute whose value ends with a specified value.	**a[href$=".pdf"]** Matches any <a> element whose href attribute ends with the text “.pdf“

Attribute Selector Example

Here we show an attribute selector style links to show PDF files differently than other links (from Figure 4.5)

```
a[href$=".pdf"] {
   background: url(pdf_icon.svg) no-repeat left center;
   padding-left:19px;
}
```

This attribute selector selects only those <a> elements whose `href` attribute value ends with the characters `".pdf"`.

```
<a href="abc.html">First link</a>
<a href="sample.pdf">One PDF</a>
<a href="another.pdf">Two PDF</a>
```

Pseudo-Element and Pseudo-Class Selectors

A **pseudo-element selector** is a way to select something that does not exist explicitly as an element in the HTML document tree.

- You can select the **first line** or **first letter** of any HTML element using a pseudo-element selector.

A **pseudo-class selector** targets either a particular state or a variety of family relationships

- This type of selector is for targeting link states and for adding hover styling for other elements

Common Pseudo-Class and Pseudo-Element Selectors

- **a:link** Selects links that have not been visited.
- **a:visited** Selects links that have been visited.
- **:hover** Selects elements that the mouse pointer is currently above.
- **:active** Selects an element that is being activated by the user. A typical example is a link that is being clicked.
- **:first-child** Selects an element that is the first child of its parent.
- **:first-letter** Selects the first letter of an element.
- **:first-line** Selects the first line of an element.
- Table 4.5 for more common selectors.

Styling a link using pseudo-class selectors

Home Mens Womens Kids House Garden Contact

```
li:hover { ... }
```

Home Mens Womens Kids House Garden Contact

```
li:first-child { ... }
```

Home Mens Womens Kids House Garden Contact

```
li:nth-child(2n) { ... }
```

Home Mens Womens Kids House Garden Contact

```
li:nth-child(2n-1) { ... }
```

Arsenal

Chelsea

Liverpool

Manchester United

click

West Ham United

```
a:link { ... }
a:visited { color: royalblue }
a:hover { color: lavender; background-color: hotpink}
a:active { font-weight: bold }
a:link:last-child { text-decoration: none}
```

Pseudo selectors can be combined

Syntax of a descendant selection

context selected element

```
div   p { … }
```

Selects a <p> element somewhere within a element

```
#main div p:first-child { … }
```

Selects the first <p> element somewhere within a element that is somewhere within an element with an id="main"

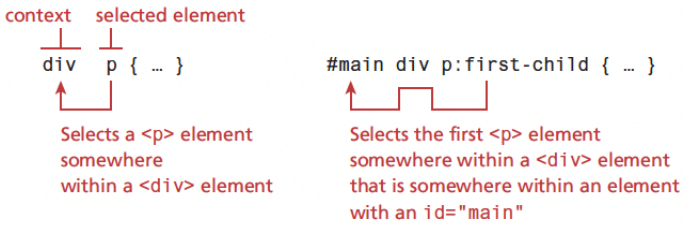
Contextual Selectors

A **contextual selector** (in CSS3 also called **combinators**) allows you to select elements based on their *ancestors*, *descendants*, or *siblings*.

- A **descendant selector** matches all elements that are contained within another element. The character used to indicate descendant selection is a space " "
- A **child selector** matches a specified element that is a direct child of the specified element. The character used is a ">"
- An **Adjacent selector** matches a specified element that is the next sibling of the specified element. The character used is a "+"
- A **General selector** matches All following elements that shares the same parent as the specified element. The character used is a "~"

Contextual Selectors

Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element.	`div p` Selects a `<p>` element that is contained somewhere within a `<div>` element. That is, the `<p>` can be any descendant, not just a child.
Child	A specified element that is a direct child of the specified element.	`div>h2` Selects an `<h2>` element that is a child of a `<div>` element.
Adjacent sibling	A specified element that is the next sibling (i.e., comes directly after) of the specified element.	`h3+p` Selects the first `<p>` after any `<h3>`.
General sibling	All following elements that shares the same parent as the specified element.	`h3~p` Selects all the `<p>` elements after an `<h3>` and that share the same parent as the `<h3>`.



Contextual selectors in action

```
<nav>
  <ul>
    <li><a href="#">Canada</a></li>
    <li><a href="#">Germany</a></li>
    <li><a href="#">United States</a></li>
  </ul>
</nav>
<div id="main">
  <time>2019-12-25</time>
  <hr>
  <p>By Ricardo on <time>2020-05-23</time></p>
  <p>Easy on the HDR buddy.</p>
  <hr>
  <p>By Susan on <time>2020-11-18</time></p>
  <p>I love Central Park.</p>
  <hr>
  <p>Usage stats are here</p>
  <h3>Social</h3>
  <p>Visits: 2300</p>
  <p>Shares: 5</p>
</div>
<footer>
  <ul>
    <li><a href="#">Home</a> | </li>
    <li><a href="#">Browse</a> | </li>
  </ul>
</footer>
```

```
ul a:link {
  color: blue;
}
```

descendant selector

```
#main>time {
  color: purple;
}
```

child selector

```
hr+p {
  color: green;
}
```

adjacent sibling selector

descendant selector

```
div time { color: red; }
```

```
h3~p {
  color: orange;
}
```

general sibling selector

The Cascade: How Styles Interact

The “Cascade” in CSS refers to how conflicting rules are handled.

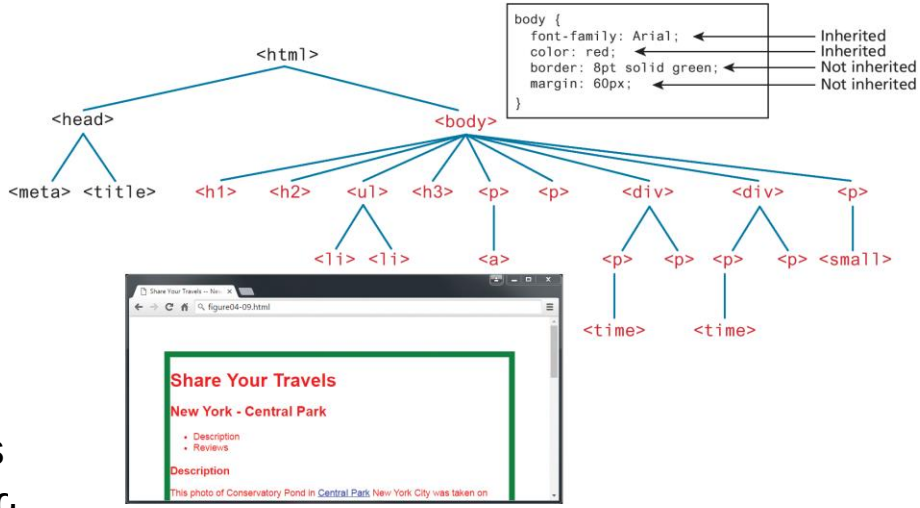
CSS uses the following cascade principles to help it deal with conflicts:

- inheritance,
- specificity, and
- location.

The Cascade: Inheritance

Inheritance is the principle that many CSS properties affect their descendants as well as themselves.

- Font, color, list, and text properties (from Table 4.1) are inheritable;
- Layout, sizing, border, background, and spacing properties are not.
- it is also possible to inherit properties that are normally not inheritable using **inherit**



The Cascade: Specificity

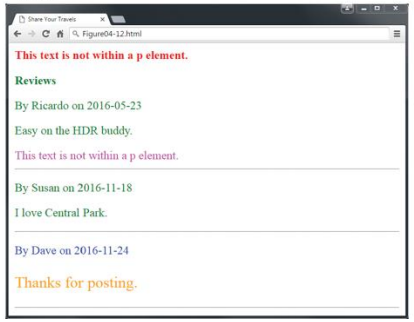
Specificity is how the browser determines which style rule takes precedence when more than one style rule could be applied.

- The more specific selector takes precedence
- Class selectors take precedence over element selectors, and id selectors take precedence over class selectors

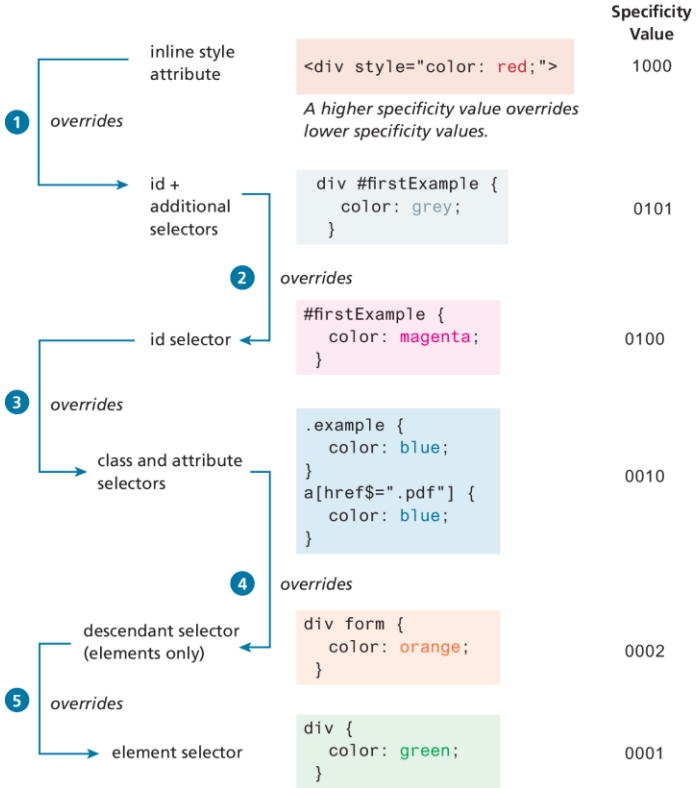
Note: In figure 4.14 <body> color and font-weight properties are overridden by the more specific and <p> selectors.

```
body {
  font-weight: bold;
  color: red;
}

div {
  font-weight: normal;
  color: magenta;
}

p {
  color: green;
}

.last {
  color: blue;
}

#verylast {
  color: orange;
  font-size: 16pt;
}
```

```
<body>
   This text is not within a p element.
   <p>Reviews</p>
   <div>
      <p>By Ricardo on <time>2016-05-23</time></p>
      <p>Easy on the HDR buddy.</p>
      This text is not within a p element.
   </div>
   <hr/>

   <div>
      <p>By Susan on <time>2016-11-18</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>

   <div>
      <p class="last">By Dave on <time>2016-11-24</time></p>
      <p class="last" id="verylast">Thanks for posting.</p>
   </div>
   <hr/>
</body>
```



Simplified Specificity algorithm

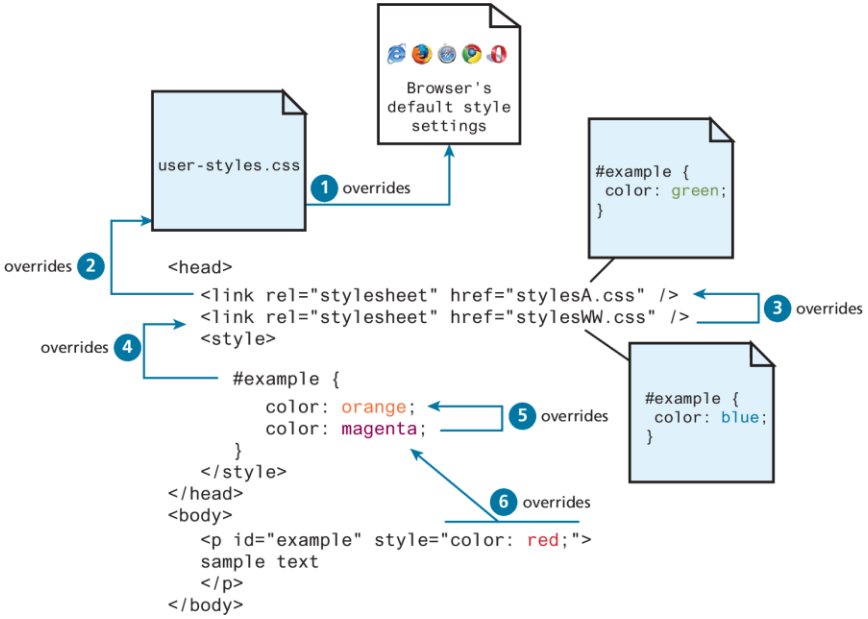


The Cascade: Location

Finally, when inheritance and specificity cannot determine style precedence, the principle of **location** will be used.

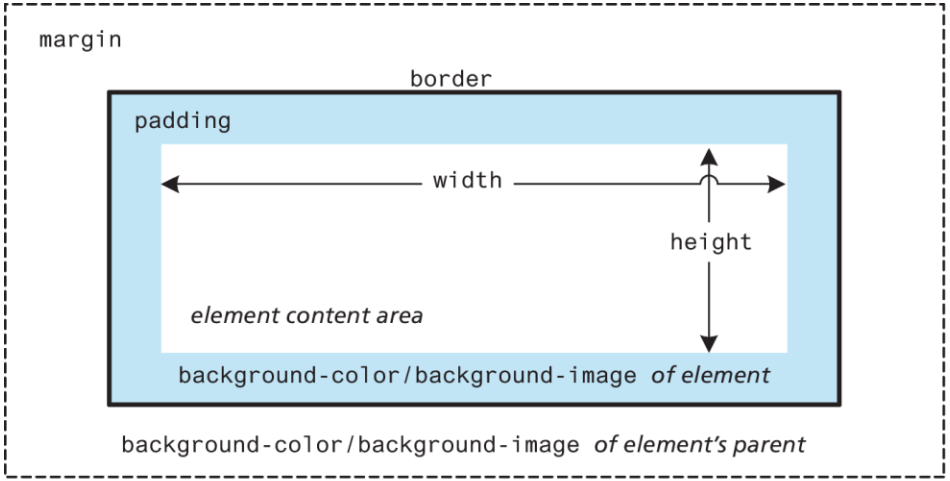
When rules have the same specificity, then the latest are given more weight.

For instance, an inline style will override one defined in an external author style sheet or an embedded style sheet.



The Box Model

In CSS, all HTML elements exist within an **element box.** In order to become proficient with CSS, you must become familiar with the box model.



Block Elements

Block-level elements such as <p>, , <h2>, <ul>, and <table> are each contained on their own line.

- without styling, two block-level elements can't exist on the same line
- Block-level elements use the normal CSS box model

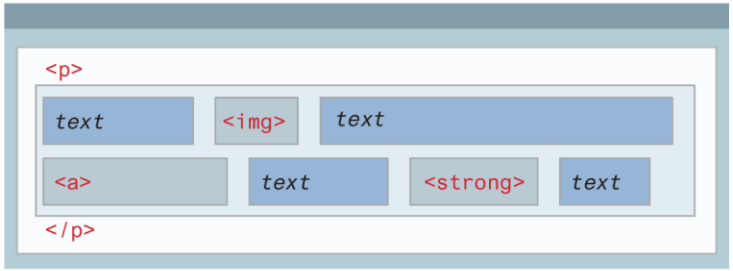
Inline Elements

Inline elements do not form their own blocks but instead are displayed within lines.

- Normal text in an HTML document is inline, as are elements such as , <a>, <img>, and .
- When there isn't enough space left on the line, the content moves to a new line

Inline Element Example

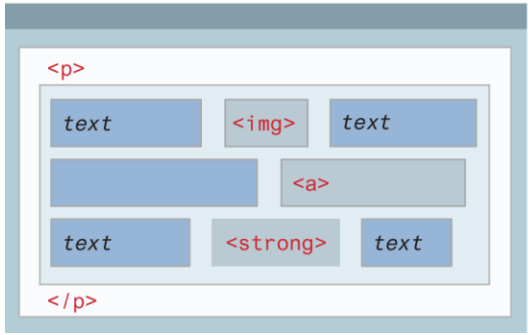
```
<p>
This photo <img src="photo-con.png"> of Conservatory Pond in
<a href="http://www.centralpark.com">Central Park</a> was
taken with a <strong>Canon EOS 30D</strong> camera.
</p>
```



Inline content is laid out horizontally left to right within its container.

Once a line is filled with content, the next line will receive the remaining content, and so on.

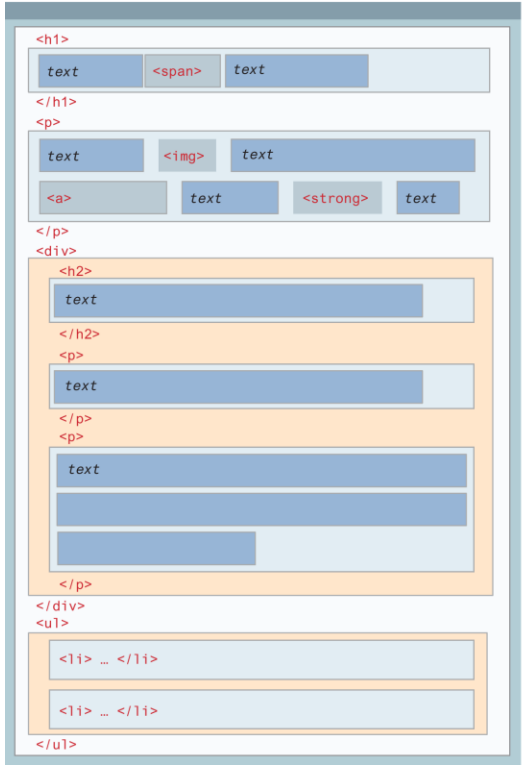
Here the content of this <p> element is displayed on two lines.



If the browser window resizes, then inline content will be "reflowed" based on the new width.

Here the content of this <p> element is now displayed on three lines.

Block and inline elements together



A document consists of block-level elements stacked from top to bottom.

Within a block, inline content is horizontally placed left to right.

Some block-level elements can contain other block-level elements (in this example, a `<div>` can contain other blocks).

In such a case, the block-level content inside the parent is stacked from top to bottom within the container (`<div>`).

Background

The background of an element fills an element out to its border

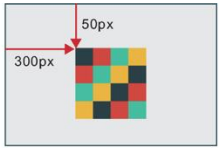
Background image properties include:

- **Background, background-attachment, background-color, background-image, background-origin, background-position, background-repeat, background-size**

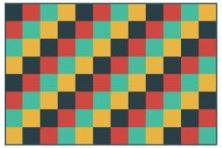
Some of these properties work together, others do not.

Background image examples

```
background-image: url(checkers.png);
background-repeat: no-repeat;
```




```
background-image: url(checkers.png) no-repeat;
background-position: 300px 50px;
```



```
background-repeat:repeat;
```

```
background-repeat:repeat-y;
```

```
background-repeat:repeat-x;
```

```
background-size:cover;
```

```
background-size:contain;
```

```
background-size:200px 100px;
```

```
background-origin:border-box;
background-size:cover;
border-size: 5px;
border-color: rgba(242,112,90,0.5)
```

```
background-origin:padding-box;
```

```
background-origin:content-box;
padding: 10px;
```

Border and Shadow Examples

```
border-style: solid;
```

```
border-style: dotted;
```

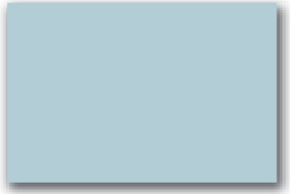
```
border-style: dashed;
```

```
border-width: 0;
```

```
border-width: 1px;
```

```
border-width: 1px 2px 0 1px;
```

```
box-shadow: 2px 2px gray;
```



```
box-shadow: 2px 2px 2px 2px gray;
```

```
border: solid 2px gold;
```

```
border-radius: 3px;
border-width: 1px;
```

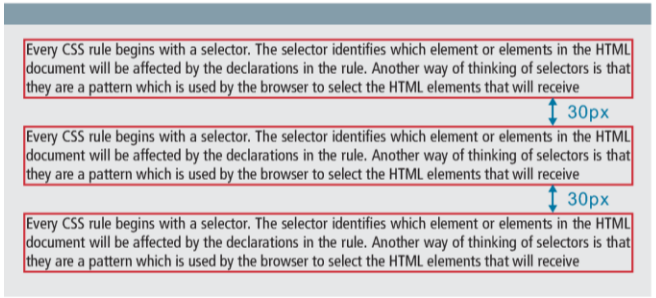
```
border-radius: 3px;
border-width: 0;
```

Margins and Padding

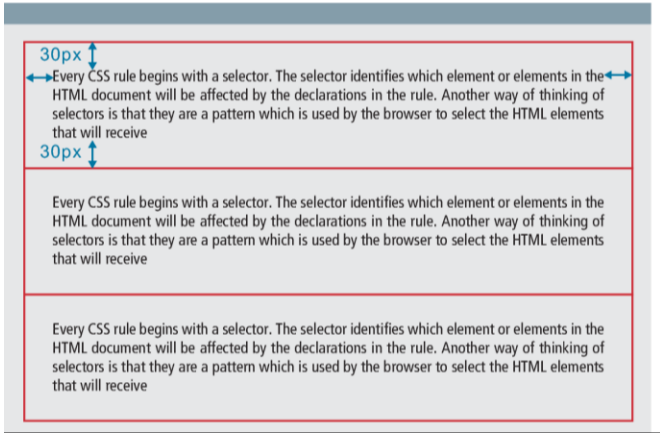
Margins add spacing around an element's content

Padding adds spacing within elements.

Borders divide the margin area from the padding area.




```
p {
  border: solid 1pt red;
  margin: 30px;
  padding: 0;
}
```




```
p {
  border: solid 1pt red;
  margin: 0;
  padding: 30px;
}
```

Collapsing margins

The W3C defines **collapsing margins** as:

- *In CSS, the adjoining margins of two or more boxes (which might or might not be siblings) can combine to form a single margin. Margins that combine this way are said to collapse, and the resulting combined margin is called a collapsed margin.*

What this means is that when the **vertical** margins of two elements touch, only the largest margin value of the elements will be displayed

Horizontal margins, on the other hand, **never** collapse.

Box Model: All or one

NOTE

With border, margin, and padding properties, it is possible to set the properties for one or more sides in a single property, or individually

border-top-color: red; / sets just the top side */*
border-right-color: green; / sets just the right side */*
border-bottom-color: yellow; / sets just the bottom side */*
border-left-color: blue; / sets just the left side */*

Alternately, we can set all four sides to a single value via:
border-color: red; / sets all four sides to red */*

Or we can set all four sides to different values via:
border-color: red green orange blue;

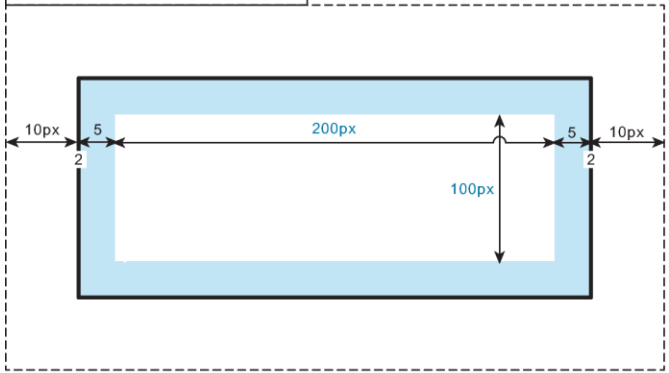
Another shortcut is to use just two values; in this case the first value sets top and bottom, while the second sets the right and left.
border-color: red yellow; */* top+bottom=red, right+left=yellow */*

Box Dimensions

- Block-level elements have **width** and **height** properties.
- They also have a **min-width, min-height, max-width**, and **max-height** properties as well to help when the width or a height might be specified as a % of its parent container
- Since the width and the height only refer to the size of the content area, the **total size** of an element is equal to the size of its content area plus the sum of its padding, borders, and margins.

```
div {
  box-sizing: content-box;
  width: 200px;
  height: 100px;
  padding: 5px;
  margin: 10px;
  border: solid 2pt black;
}
```

left margin	10	10	top margin
left border	2	2	top border
left padding	5	5	top padding
width	200	100	height
right padding	5	5	bottom padding
right border	2	2	bottom border
right margin	10	10	bottom margin
True element width	234px	134px	True element height

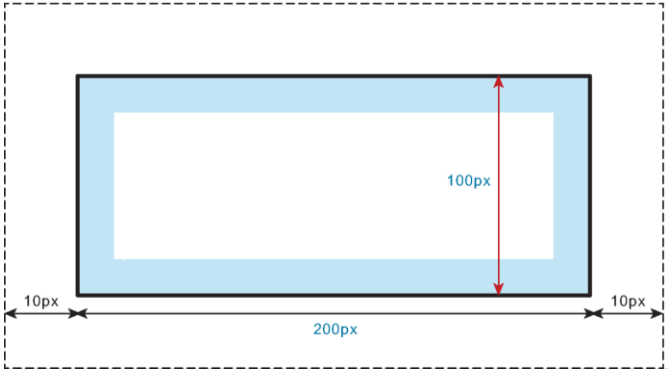


The border-box approach

To reduce complexity of adding margins, content and borders, you can use the newer border-box approach that is more intuitive.

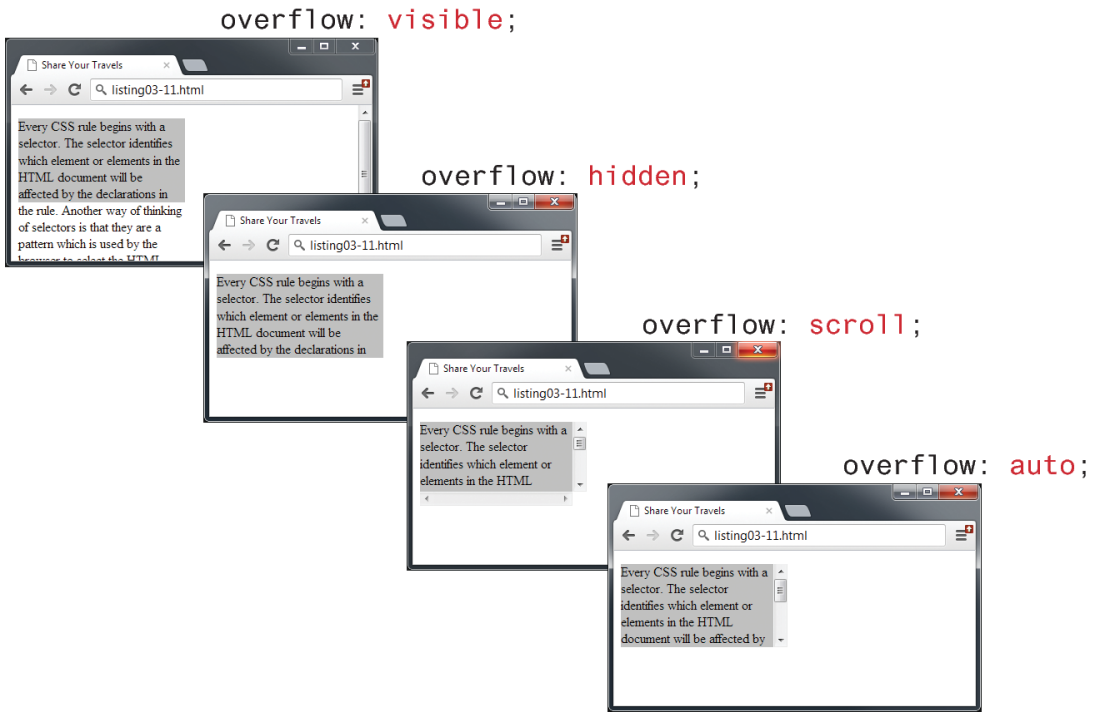
```
div {
  ...
  box-sizing: border-box;
}
```

left margin	10	10	top margin
width	200	100	height
right margin	10	10	bottom margin
True element width	220px	120px	True element height



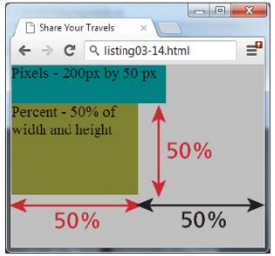
Overflow Proprerty

It is possible to control what happens with the content if the box's width and height are not large enough to display the content using the **overflow** property

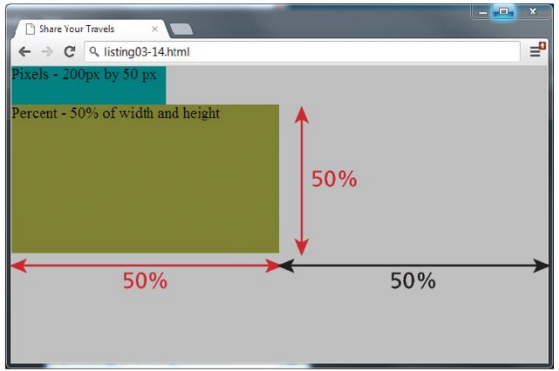


Box sizing via percent

```
<style>
  html,body {
     margin:0;
     width:100%;
     height:100%;
     background: silver;
  }
  .pixels {
     width:200px;
     height:50px;
     background: teal;
  }
  .percent {
     width:50%;
     height:50%;
     background: olive;
  }
```




```
<body>
   <div class="pixels">
     Pixels - 200px by 50 px
   </div>
   <div class="percent">
     Percent - 50% of width and height
   </div>
</body>
```



CSS Text Styling

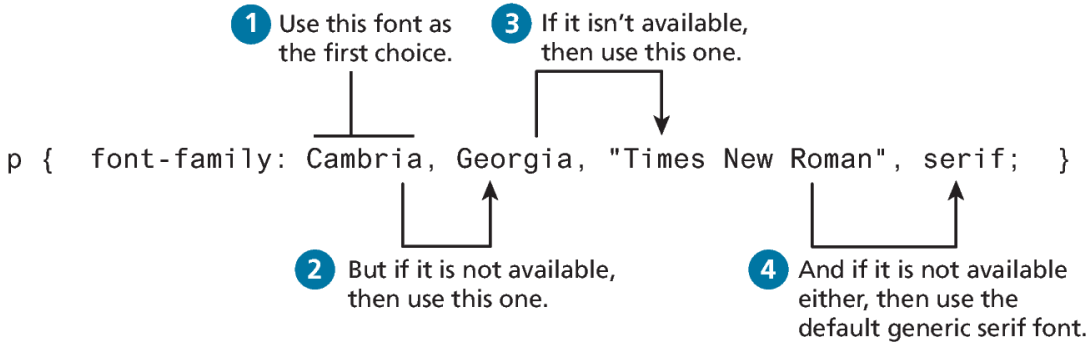
CSS provides two types of properties that affect text.

- **Font properties** that affect the font and its appearance.
- **Paragraph properties** that affect the text in a similar way no matter which font is being used
- Many of the most common font properties are shown in Table 4.9 and include
  - Font, font-family, font-style, font-size, font-variant, font-weight

Font Family

Just because a given font is available on the web developer's computer does not mean it will be available for all users.

For this reason, it is conventional to supply a so-called **web font stack**— a series of alternate fonts to use in case the original font choice is not on the user's computer



Different font families

	Generic Font-Family Name
This	serif
This	sans-serif
This	monospace
This	cursive
This	fantasy

serif

Th

Without ("sans") serif

Th

This

In a monospace font, each letter has the same width.

This

In a regular, proportionally-spaced font, each letter has a variable width.

Decorative and cursive fonts vary from system to system; rarely used as a result.

Font Sizes

If we wish to create web layouts that work well on different devices, we should learn to use relative units such as **em units** or **percentages** for our font sizes

- it can quickly become difficult to calculate actual sizes when there are nested elements
  - For this reason, CSS3 now supports a new relative measure, the **rem** (for root em unit). This unit is always relative to the size of the root element (i.e., the <html> element).

To muddy the picture even more, some developers have begun to advocate again for using the pixel as the unit of measure in CSS since modern browsers provide built-in scaling/zooming that preserve layout…

@font-face

DIVE DEEPER

The @font-face selector that allows you to use a font even if it is not installed on the user's computer. Open source font sites such as Google Web Fonts (https://fonts.google.com) and Font Squirrel (http://www.fontsquirrel.com/) have promoted the use of @font-face

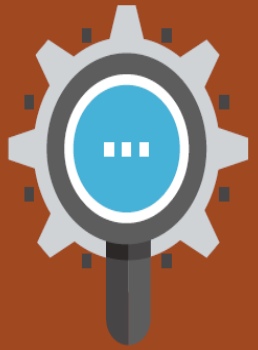
To use Droid Sans, you can add the following to your <head> section

<link href="https://fonts.googleapis.com/css?family=Droid+Sans" rel="stylesheet" type="text/css">

Or, add the following import inside one of your CSS files:

@import url(https://fonts.googleapis.com/css?family=Droid+Sans);

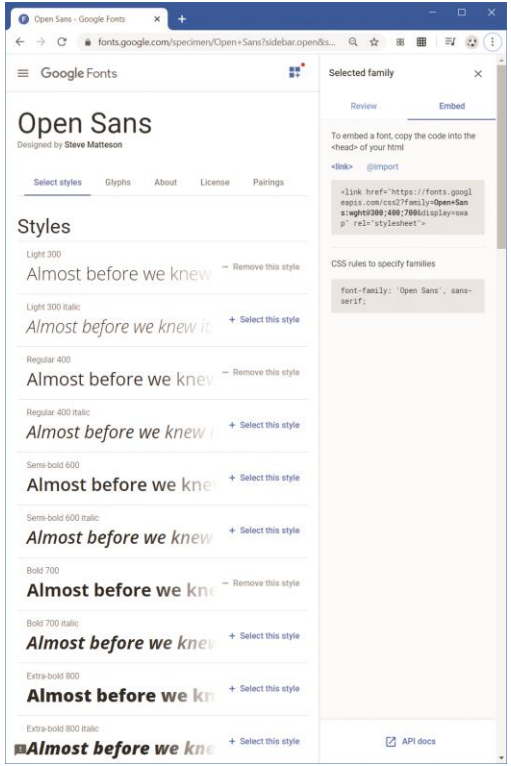
Or check out the longer-form @font-face selector (on p 170)



Font Weight

Google Fonts made web fonts available to the masses, before that **font-weight** was typically set to either normal or bold

- For instance, the popular Open Sans font has five different weights: **light, regular, semi-bold, bold,** and **extra bold**
- Within your CSS you specify which of these weights by using their numeric value, which typically ranges from 100 and 900, with larger numbers bolder than lower numbers



Paragraph Properties

Just as there are properties that affect the font in CSS, there is also a range of CSS properties that affect text independently of the font.

Table 2.10 describes the following paragraph properties in detail: **letter-spacing, line-height, list-style-image, list-style-type, text-align, text-decoration, text-direction, text-intent, text-shadow, text-transform, vertical-align** and **word-spacing**

Sample text properties

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

```
line-height: normal;
```

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

```
line-height: 1.5;
```

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

```
text-indent: 4em;
```

Every CSS rule begins with a selector.

```
text-align: left;
```

Every CSS rule begins with a selector.

```
text-align: center;
```

Every CSS rule begins with a selector.

```
text-align: right;
```

Every CSS rule begins with a selector.

```
letter-spacing: 3px;
```

```
vertical-align: top;
```

```
vertical-align: middle;
```

```
vertical-align: bottom;
```

EVERY RULE BEGINS WITH A SELECTOR

```
text-transform: uppercase;
```

every rule begins with a selector

```
text-transform: lowercase;
```

selector

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

selector

```
text-decoration: none;
```

CSS Frameworks and Variables

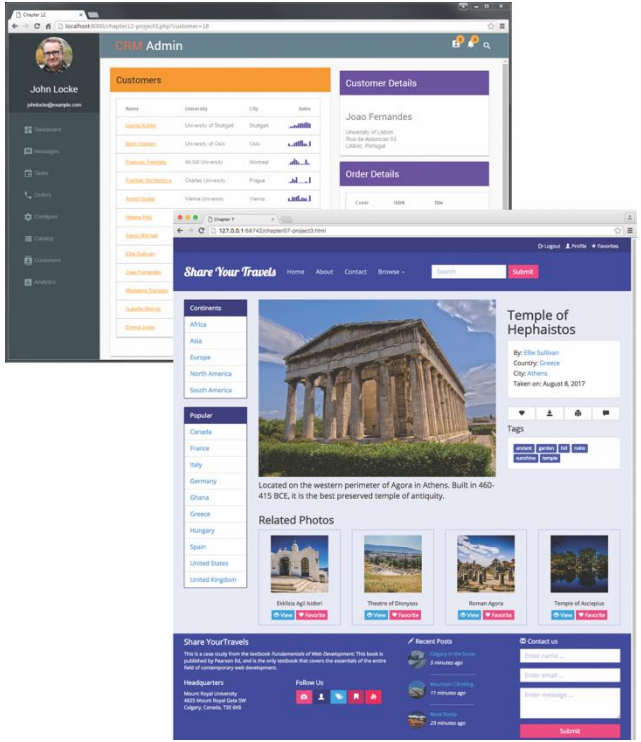
- This chapter and the exercises should make you reasonably confident about the **CSS box model** and **text formatting**
- Chapter 5 covers the CSS for working with forms and tables
- Chapter 7 covers layout, transitions, and animations as well
- If you are not feeling overwhelmed yet by CSS, you might be by the end of Chapter 7.
  - as an alternative to mastering CSS and getting an acceptable visual design, some developers instead use an already developed **CSS framework**

What is a CSS Framework?

A **CSS framework** is a set of CSS classes or other software tools that make it easier to use and work with CSS.

- Early CSS frameworks became popular chiefly as a way to more easily create complex grid-based layouts
- More sophisticated subsequent CSS Frameworks such as
  - Bootstrap (**https://getbootstrap.com/**) and
  - Foundation (**https://get.foundation/**) provide much more than a grid system; they provide a comprehensive set of predefined CSS classes, which makes it easier to construct a consistent and attractive web interface.
- Table 4.11 lists even more popular CSS Frameworks

Examples using Frameworks



CSS Variables

CSS Variables (also called custom properties) are a new feature that let you

- define variables (which must begin with a double hyphen) at the top of your CSS file usually within a special **:root** pseudo-class selector, and
- reference those variables as property values using the **var()** CSS function.

Duplication of colors, spacing, and borders, often happens numerous times within a single file (and rightly so). CSS variables address this issue without using a preprocessor (Chapter 7).

Example Part 1 (duplicate values)

```
header {
  background-color: #431c5d;
  color: #e05915;
  padding: 4px;
  box-shadow: 6px 5px 20px 1px rgba(0,0,0,0.22);
  margin: 0;
}
header button {
  background-color: #e05915;
  border-radius: 5px;
  border-color: #e6e9f0;
  padding: 4px;
  color: #e6e9f0;
  font-size: 18px;
  margin-top: 9px;
}
#results {
  background-color: #431c5d;
  font-size: 18px;
  border-radius: 5px;
  padding: 4px;
  box-shadow: 6px 5px 20px 1px rgba(0,0,0,0.22);
}
```

LISTING 4.8 Duplicate property values in CSS

Example Part 2 (CSS variables)

```
:root {
  --bg-color-main: #431c5d;
  --bg-color-secondary: #e05915;
  --fg-color-main: #e6e9f0;
  --radius-boxes: 5px;
  --padding-boxes: 4px;
  --fontsize-default: 18px;
  --shadow-color: rgba(0,0,0,0.22);
  --dropshadow: 6px 5px 20px 1px var(--shadow-color);
}
header {
  background-color: var(--bg-color-main);
  color: var(--bg-color-secondary);
  padding: var(--padding-boxes);
  box-shadow: var(--dropshadow);
  margin: 0;
}
header button {
  background-color: var(--bg-color-secondary);
  border-radius: var(--radius-boxes);
  border-color: var(--fg-color-main);
  padding: var(--padding-boxes);
  color: var(--fg-color-main);
  font-size: var(--fontsize-default);
  margin-top: calc( --fontsize-default / 2 );
}
#results {
  background-color: var(--bg-color-main);
  font-size: var(--fontsize-default:);
  border-radius: var(--radius-boxes);
  padding: var(--padding-boxes);
  box-shadow: var(--dropshadow);
}
```

LISTING 4.9 Using CSS Variables

Key Terms

absolute units

attribute selector

author-created style sheets

block-level elements

box model

browser style sheets

cascade

Cascading Style Sheets

class selector

collapsing margins

combinators

contextual selector

CSS

CSS framework

CSS variable

CSS3 modules

Declaration

declaration block

descendant selector

Document Object Model

element box

element selectors

em units

embedded style sheets

external style sheets

generic font

grouped selector

id selector

inheritance

inline styles

location

margin

Padding

percentages

property

property:value pair

pseudo-class selector

pseudo-element selector

relative units

rem

responsive design

selector

specificity

style rules

universal element selector

user style sheets

vendor prefixes

web font stack

x-height

Copyright