Chapter2- Introduction to CSS

When tags like <font> and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page

2.1 What is CSS?

- Cascading Style Sheets (CSS) is a W3C standard style sheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML).
- CSS describes how elements should be rendered on screen, on paper, in speech, or on other media. With CSS, designer can assign font properties, colors, sizes, borders, background images, and even position 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:** The degree of formatting control in CSS is significantly better than that provided in HTML. CSS gives web authors fine-grained control over the appearance of their web content.
- **Improved site maintainability:** Websites become significantly more maintainable because all formatting can be centralized into one CSS file, or a small handful of them. This allows you to make site-wide visual modifications by changing a single file.
- **Improved accessibility:** CSS-driven sites are more accessible. By keeping presentation out of the HTML, screen readers and other accessibility tools work better, thereby providing a significantly enriched experience for those reliant on accessibility tools.

- **Improved page download speed:** A site built using a centralized set of CSS files for all presentation will also be quicker to download because 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 Versions

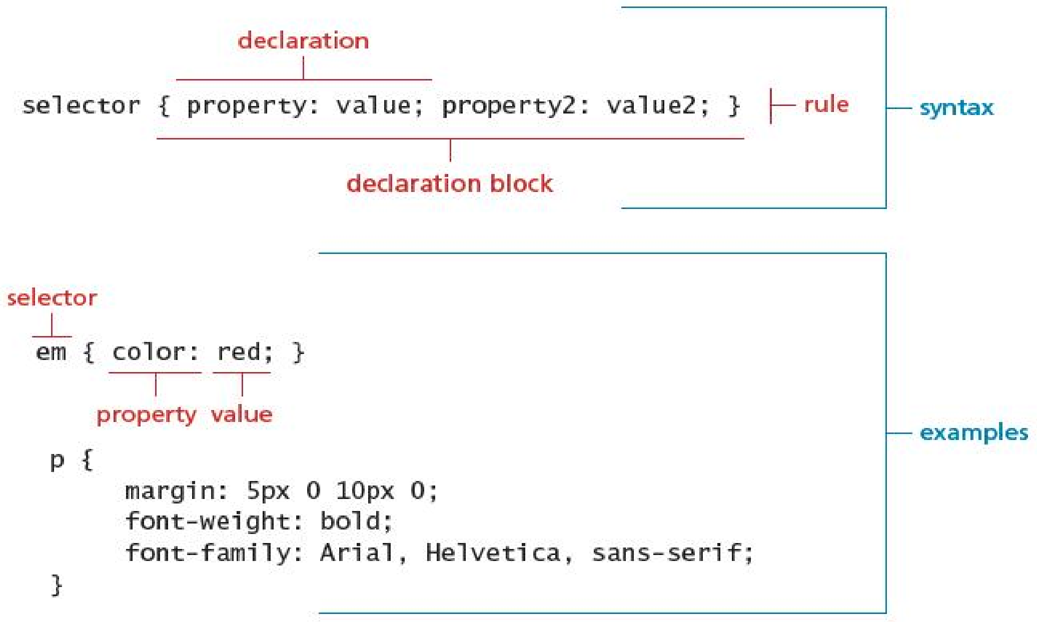
In the early 1990s, a variety of different style sheet standards were proposed, including JavaScript style sheets, which was proposed by Netscape in 1996. Netscape's proposal was one that required the use of JavaScript programming to perform style changes. Thankfully for nonprogrammers everywhere, the W3C decided to adopt CSS, and by the end of 1996 the CSS Level 1 Recommendation was published.

A year later, the CSS Level 2 Recommendation (also more succinctly labeled simply as CSS2) was published. Even though work began over a decade ago, an updated version of the Level 2 Recommendation, CSS2.1, did not become an official W3C Recommendation until June 2011. And to complicate matters even more, all through the last decade (and to the present day as well), during the same time the CSS2.1 standard was being worked on, a different group at the W3C was working on a CSS3 draft.

To make CSS3 more manageable for both browser manufacturers and web designers, the W3C has subdivided it into a variety of different **CSS3 modules**. So far the following CSS3 modules have made it to official W3C Recommendations: CSS Selectors, CSS Namespaces, CSS Media Queries, and CSS Color.

2.2 CSS Syntax

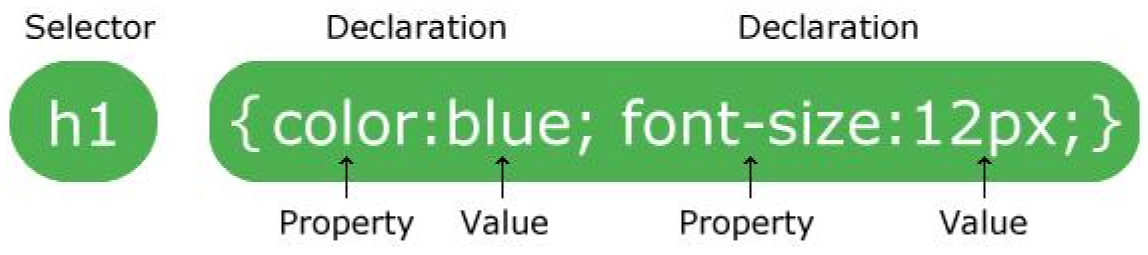
- A CSS rule-set consists of a selector and a declaration block:



```
selector { property: value; property2: value2; }

em { color: red; }

p {
     margin: 5px 0 10px 0;
     font-weight: bold;
     font-family: Arial, Helvetica, sans-serif;
}
```

CSS syntax

- 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

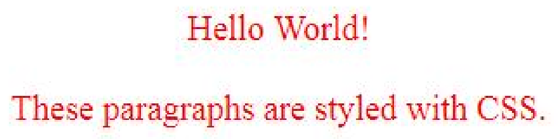


- Selectors are a pattern that is used by the browser to select the HTML elements that will receive the style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- A CSS declaration always **ends with a semicolon**, and declaration blocks are surrounded by curly braces.
- 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 (each pair is also called a declaration)
- The series of declarations is also called the declaration block.
- A declaration block can be together on a single line, or spread across multiple lines. The browser ignores white space (i.e., spaces, tabs, and returns) between your CSS rules so you can format the CSS however you want.
- Notice that each declaration is terminated with a semicolon.
- The semicolon for the last declaration in a block is in fact optional.

<u>Simple Example of CSS:</u>

```
<!DOCTYPE html>
 <html>
        <head>
        <style>
                p {
                color: red;
                text-align: center;
                }
        </style>
        </head>
<body>
        <p>Hello World!</p>
        <p>These paragraphs are styled with CSS.</p>
```

```
</body>
</html>
```



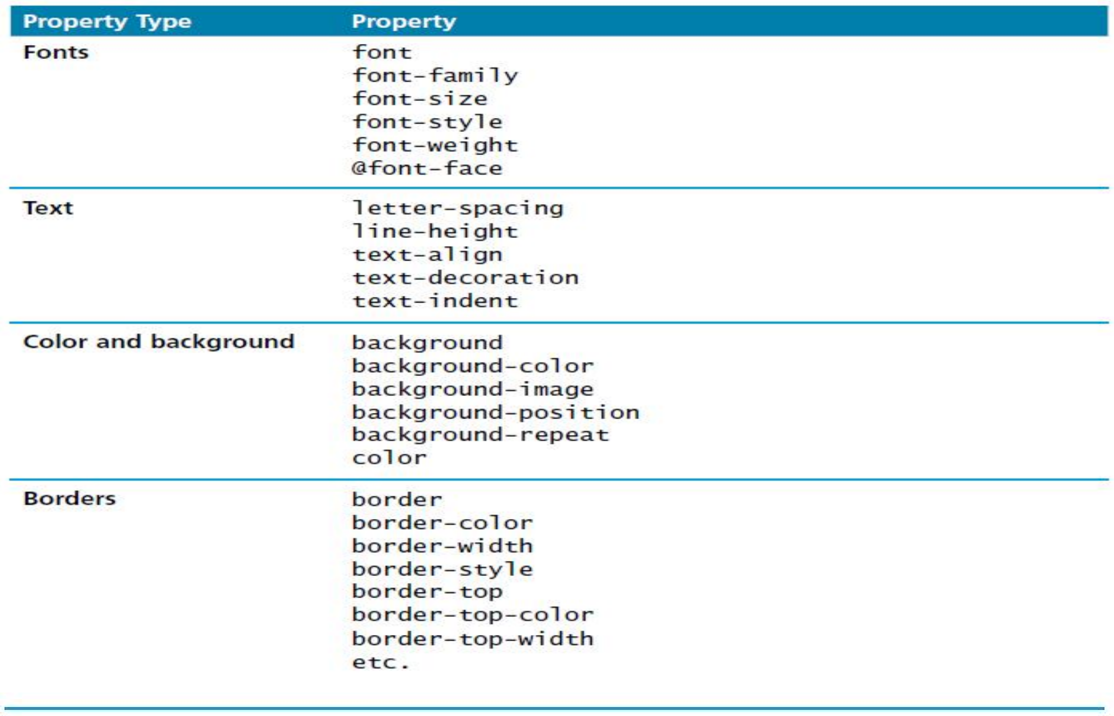
Hello World!

These paragraphs are styled with CSS.

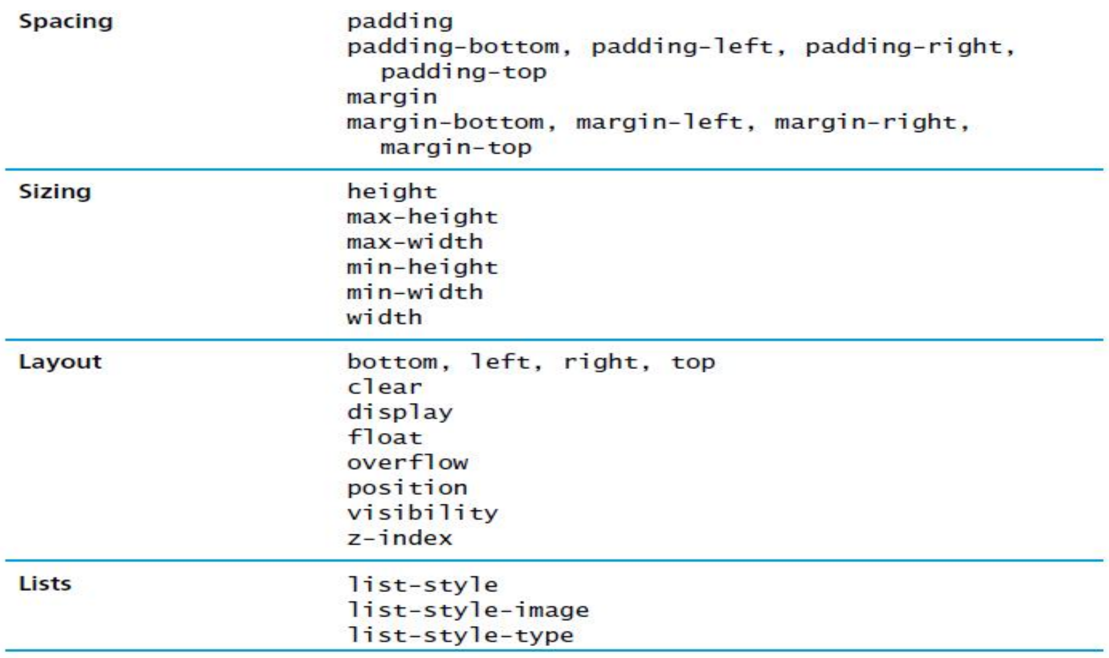
CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

Properties

- Each individual CSS declaration must contain a property.
- These property names are predefined by the CSS standard.
- The CSS2.1 recommendation defines over a hundred different property names



Property Type	Property
Fonts	font
	font-family
	font-size
	font-style
	font-weight
	@font-face
Text	letter-spacing
	line-height
	text-align
	text-decoration
	text-indent
Color and background	background
	background-color
	background-image
	background-position
	background-repeat
	color
Borders	border
	border-color
	border-width
	border-style
	border-top
	border-top-color
	border-top-width
	etc.



Spacing	`padding`
	`padding-bottom, padding-left, padding-right, padding-top`
	`margin`
	`margin-bottom, margin-left, margin-right, margin-top`

Sizing	`height`
	`max-height`
	`max-width`
	`min-height`
	`min-width`
	`width`

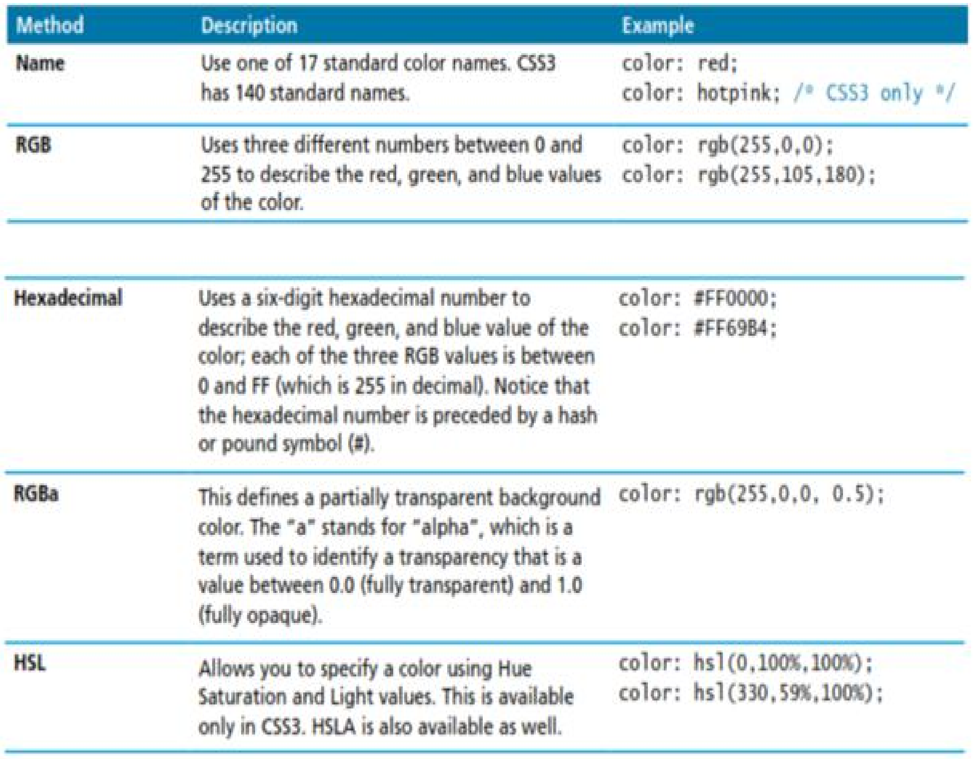
Layout	`bottom, left, right, top`
	`clear`
	`display`
	`float`
	`overflow`
	`position`
	`visibility`
	`z-index`

Lists	`list-style`
	`list-style-image`
	`list-style-type`

TABLE . Common CSS Properties

Values

- 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.
- CSS supports a variety of different ways of describing color. Table below lists the different ways you can describe a color value in CSS.



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; each of the three RGB values is between 0 and FF (which is 255 in decimal). Notice that the hexadecimal number is preceded by a hash or pound symbol (#).	`color: #FF0000;`
		`color: #FF69B4;`

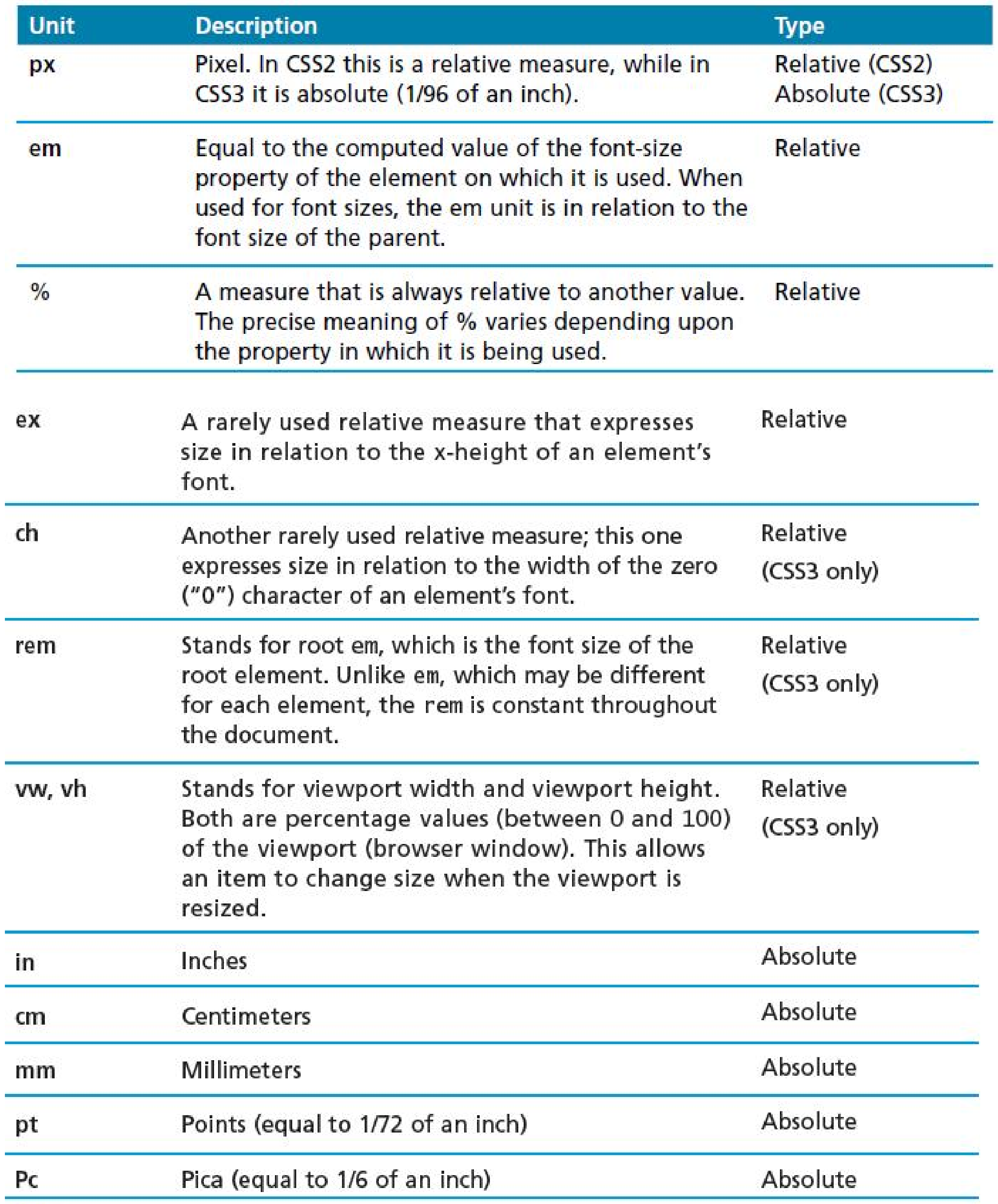
RGBa	This defines a partially transparent background color. The "a" stands for "alpha", which is a term used to identify a transparency that is a value between 0.0 (fully transparent) and 1.0 (fully opaque).	`color: rgb(255,0,0, 0.5);`

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: hsl(330,59%,100%);`

Color Values

There are multiple ways of specifying color in CSS and also for specifying a unit of measurement. These units can sometimes be complicated to work with.

When working with print design, we generally make use of straightforward absolute units such as inches or centimeters and picas or points. However, because different devices have differing physical sizes as well as different pixel resolutions and because the user is able to change the browser size or its zoom mode, these absolute units don’t always make sense with web element measures.



Unit	Description	Type
px	Pixel. In CSS2 this is a relative measure, while in CSS3 it is absolute (1/96 of an inch).	Relative (CSS2) Absolute (CSS3)
em	Equal to the computed value of the font-size property of the element on which it is used. When used for font sizes, the em unit is in relation to the font size of the parent.	Relative
%	A measure that is always relative to another value. The precise meaning of % varies depending upon the property in which it is being used.	Relative
ex	A rarely used relative measure that expresses size in relation to the x-height of an element’s font.	Relative
ch	Another rarely used relative measure; this one expresses size in relation to the width of the zero (”0”) character of an element’s font.	Relative (CSS3 only)
rem	Stands for root em, which is the font size of the root element. Unlike em, which may be different for each element, the rem is constant throughout the document.	Relative (CSS3 only)
vw, vh	Stands for viewport width and viewport height. Both are percentage values (between 0 and 100) of the viewport (browser window). This allows an item to change size when the viewport is resized.	Relative (CSS3 only)
in	Inches	Absolute
cm	Centimeters	Absolute
mm	Millimeters	Absolute
pt	Points (equal to 1/72 of an inch)	Absolute
Pc	Pica (equal to 1/6 of an inch)	Absolute

TABLE Units of Measure Values

Table lists the different units of measure in CSS. Some of these are **relative units**, in that they are based on the value of something else, such as the size of a parent element. Others are **absolute units**, in that they have a real-world size. Unless you are defining a style sheet for printing, it is recommended you avoid using absolute units. Pixels are perhaps the one popular exception

2.3 Location of Styles:

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet. There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

These three are not mutually exclusive.

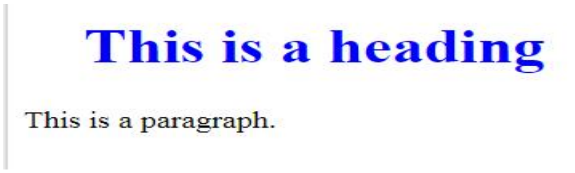
Inline styles: An inline style may be used to apply a unique style for a single element. To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

An inline style only affects the element it is defined within and overrides any other style definitions for properties used in the inline style. Notice that a selector is not necessary with inline styles and that semicolons are only required for separating multiple rules.

Using inline styles is generally discouraged since they increase bandwidth and decrease maintainability (because presentation and content are intermixed and because it can be difficult to make consistent inline style changes across multiple files). Inline styles can, however, be handy for quickly testing out a style change.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;margin-left:30px;">This is a
heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Output:


This is a heading

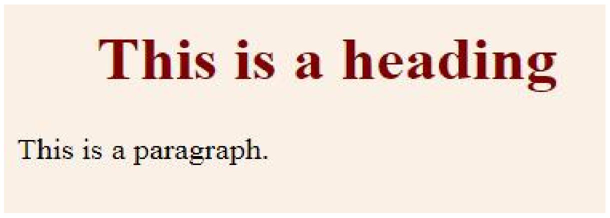
This is a paragraph.

<u>**Embedded style sheet**</u>: **Embedded style sheets** (also called **internal styles**) are style rules placed within the <style> element (inside the <head> element of an HTML document) as shown in example. Since each HTML document has its own <style> element, it is more difficult to consistently style multiple documents when using embedded styles.

Just as with inline styles, embedded styles can, however, be helpful when quickly testing out a style that is used in multiple places within a single HTML document.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Output:


This is a heading

This is a paragraph.

<u>**External style sheet**</u>: **External style sheets** are style rules placed within an external text file with the **.css** extension. This is by far the most common place to locate style rules because it provides the best maintainability. When you make a change to an external style sheet, all HTML documents that reference that style sheet will automatically use the updated version.

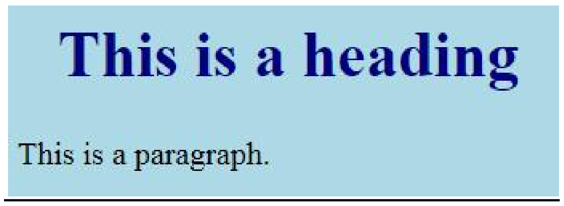
The browser is able to cache the external style sheet, which can improve the performance of the site as well. To reference an external style sheet, you must use a <link> element (within the <head> element. Several style sheets can be linked at a time; each linked style sheet will require its own <link> element.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Mystyle.css

```
body                                                    {
                         background-color:          lightblue;
}

h1                                                      {
                                 color:                  navy;
                             margin-left:                20px;
}
```

Output:



This is a heading

This is a paragraph.

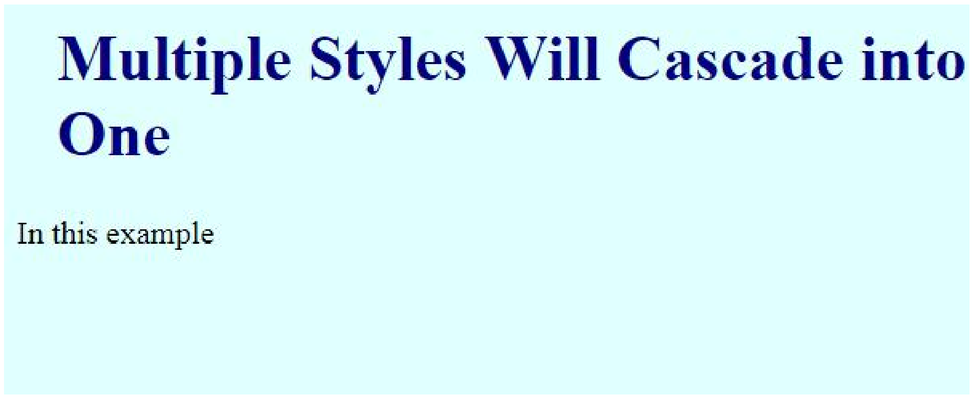
Cascading Order: All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
body {background-color: linen;}
</style>
</head>
<body style="background-color: lightcyan">

<h1>Multiple Styles Will Cascade into One</h1>
<p>In this example</p>

</body>
</html>
```

Output:



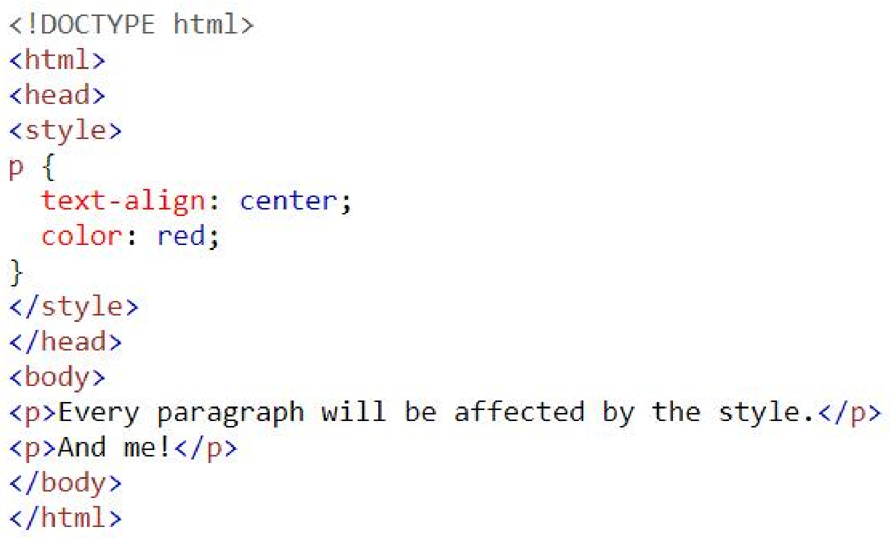
```
Multiple Styles Will Cascade into
One

In this example
```

2.4 Selectors: Selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

When defining CSS rules, selector must be used first to tell the browser which elements will be affected by the property values. CSS selectors allow selecting individual or multiple HTML elements. The three basic selector types that have been around since the earliest CSS2 specification are:

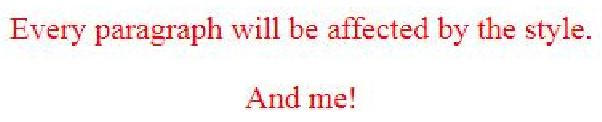
1. **Element selectors:** **The element selector selects elements based on the element name. It** selects all instances of a given HTML element. The example CSS rules in example illustrate two element selectors. All elements can be selected by using the **universal element selector**, which is the * (asterisk) character.

Example 1:



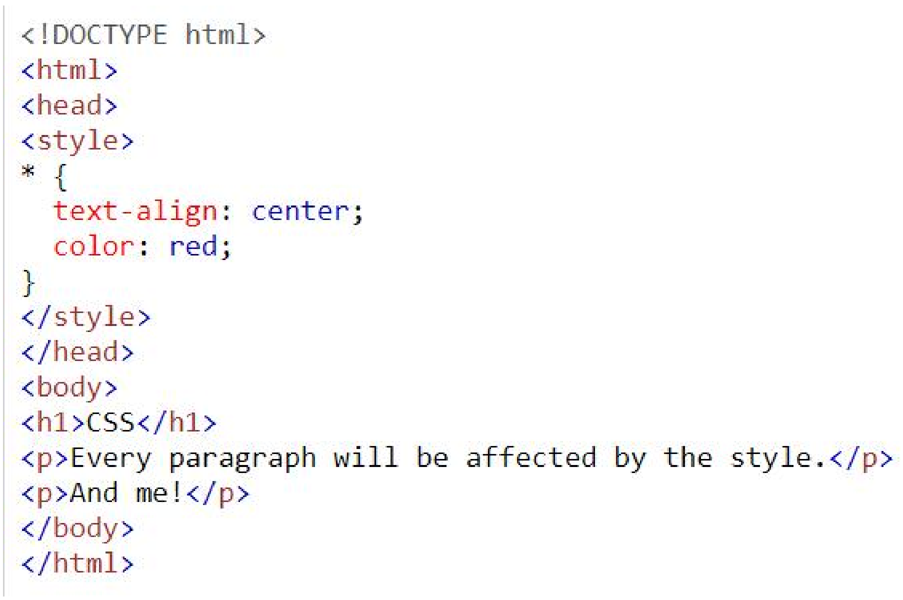
```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<p>Every paragraph will be affected by the style.</p>
<p>And me!</p>
</body>
</html>
```

Output:



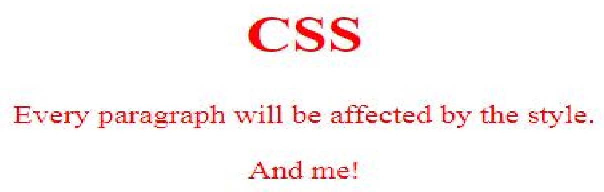
```
Every paragraph will be affected by the style.

                  And me!
```

Example 2:



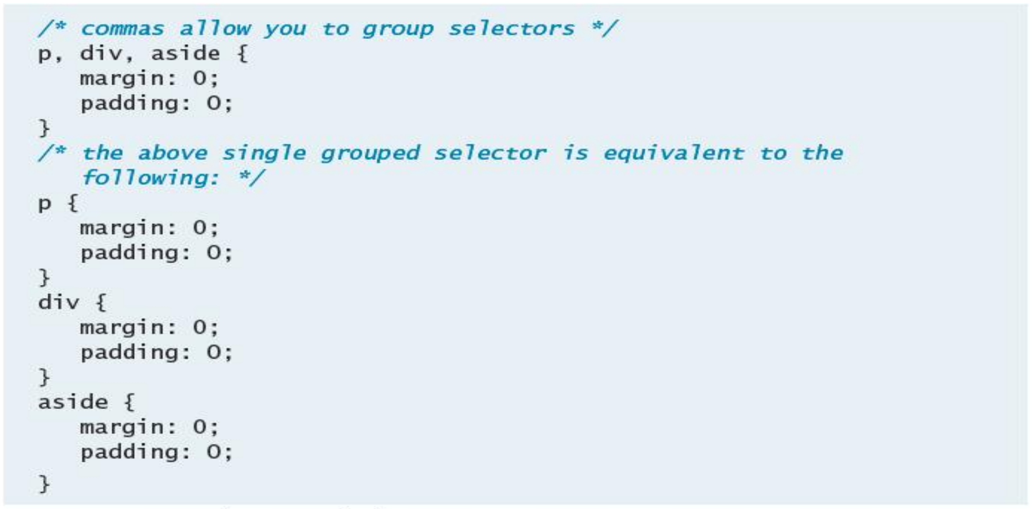
```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<h1>CSS</h1>
<p>Every paragraph will be affected by the style.</p>
<p>And me!</p>
</body>
</html>
```

Output:



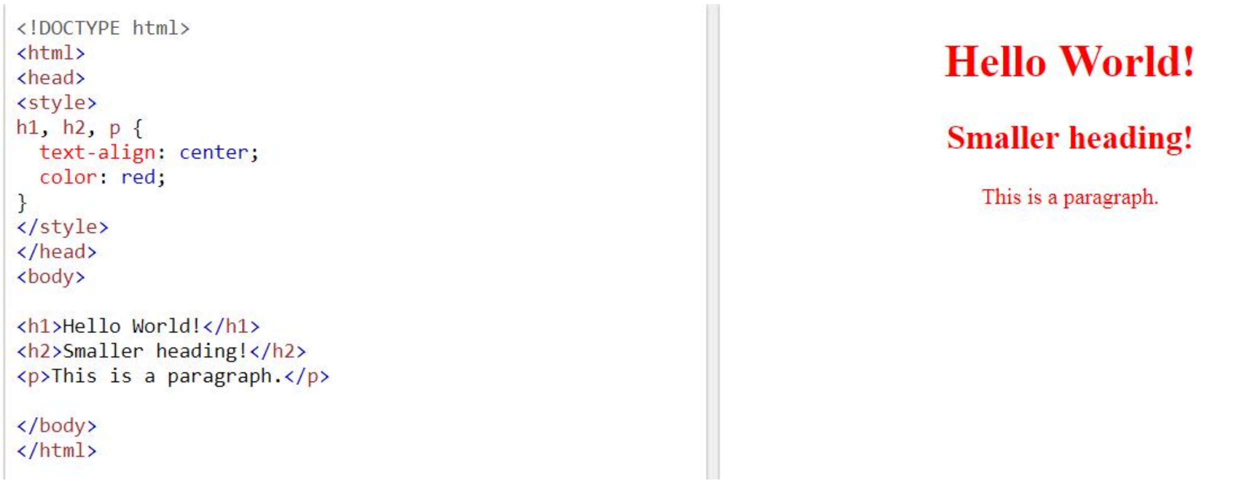
```
CSS

Every paragraph will be affected by the style.

And me!
```

2. **Grouped selector**: **Group selectors are used for** the same declaration in **CSS** to shrink style sheets. **CSS** allows grouping multiple **selectors** that share the same declaration. This optimization technique allows applying the same style to multiple elements to save space Grouping is used when elements are with the same style definitions.



```
/* 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;
}
```

Sample grouped selector



```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>

</body>
</html>
```

Hello World!

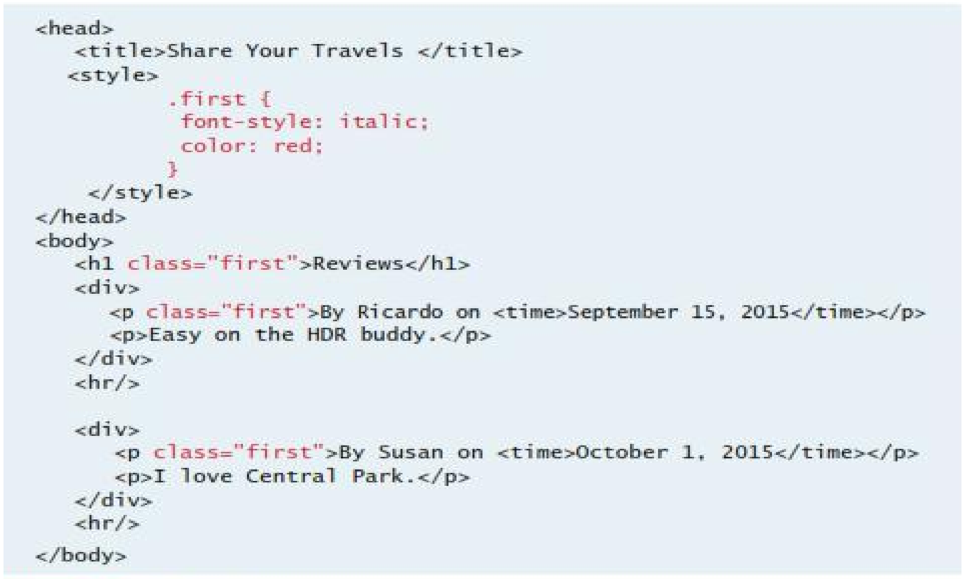
Smaller heading!

This is a paragraph.

3. **Class selectors**: A **class selector** allows to simultaneously targeting different HTML elements regardless of their position in the document tree. If a series of HTML elements have been labeled with the same class attribute value, then styling is by using a class selector

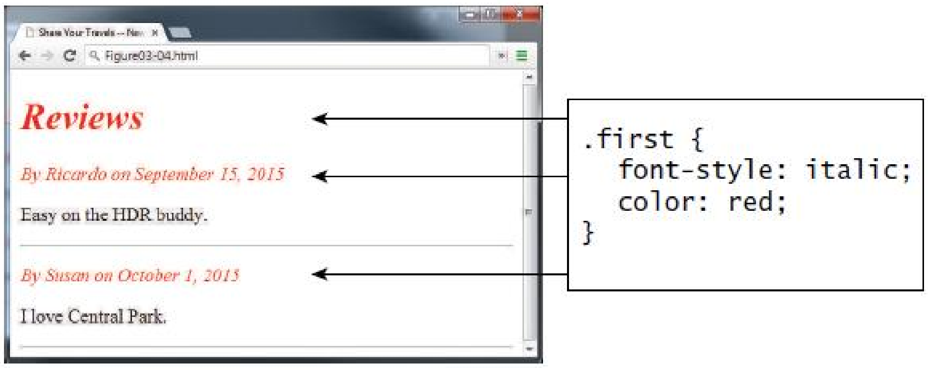
Syntax: period (.) followed by the class name.

Example 1:



```
<head>
   <title>Share Your Travels </title>
  <style>
          .first {
           font-style: italic;
           color: red;
          }
   </style>
</head>
<body>
   <h1 class="first">Reviews</h1>
   <div>
      <p class="first">By Ricardo on <time>September 15, 2015</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <hr/>

   <div>
      <p class="first">By Susan on <time>October 1, 2015</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>
</body>
```

LISTING 3.5 Class selector example



Reviews

By Ricardo on September 15, 2015

Easy on the HDR buddy.

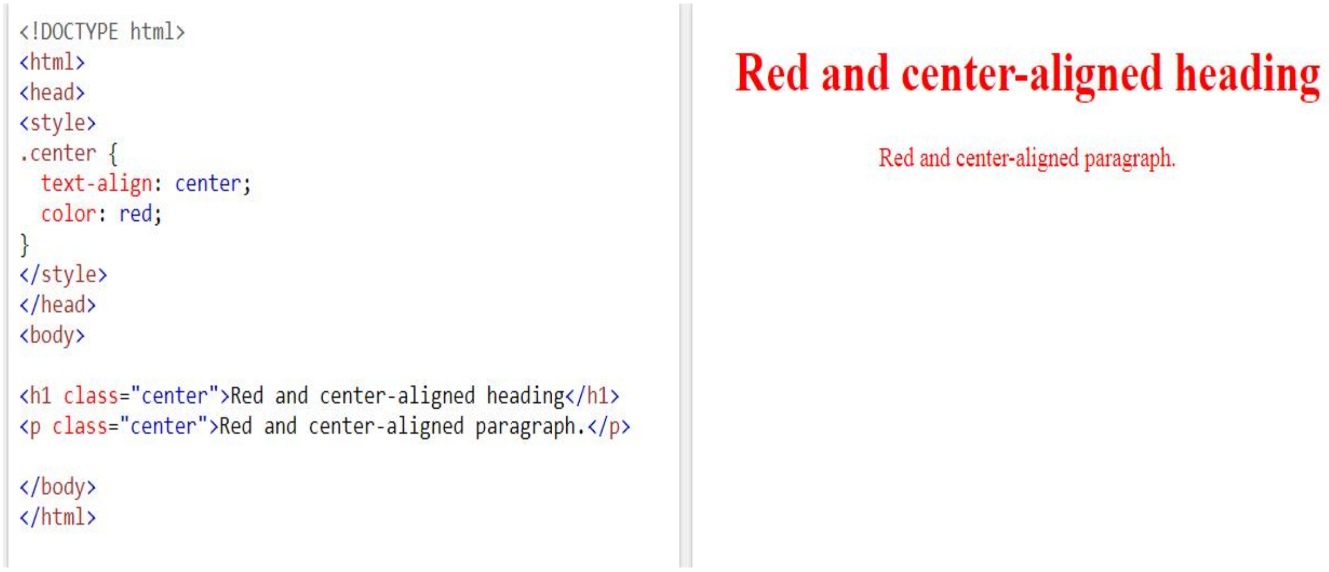
By Susan on October 1, 2015

I love Central Park.

```
.first {
  font-style: italic;
  color: red;
}
```

Class selector example in browser

Example 2:



```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

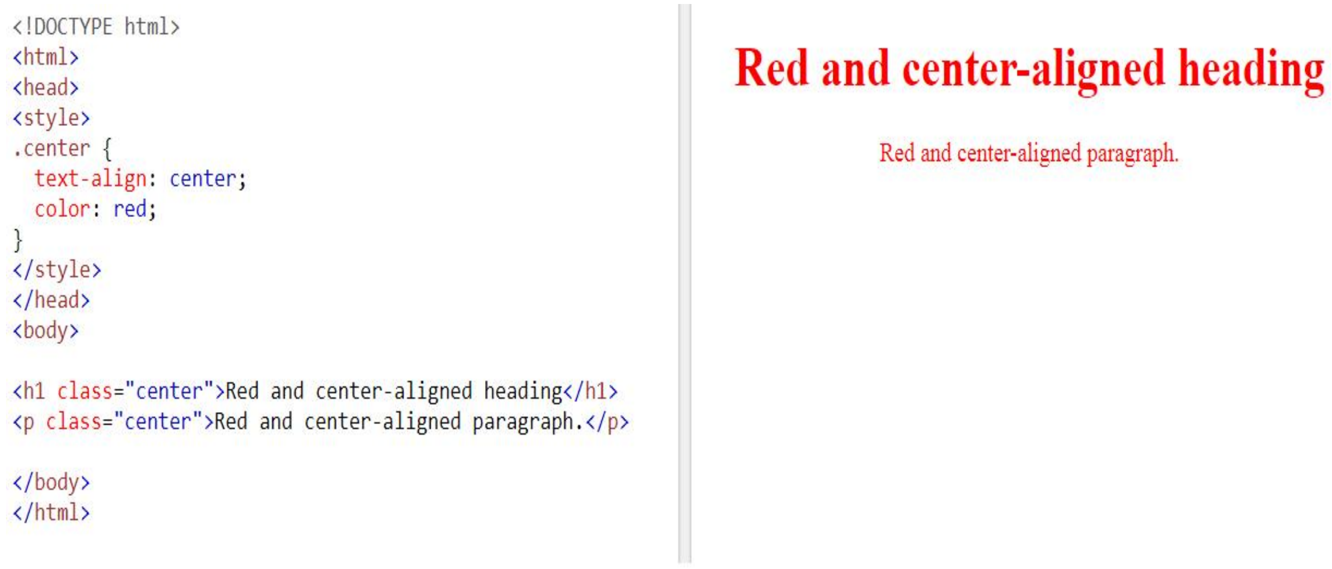
Red and center-aligned heading

Red and center-aligned paragraph.

4. Id Selectors: An **id selector** allows targeting a specific element by its id attribute regardless of its type or position.

Id selectors should only be used when referencing a single HTML element since an id attribute can only be assigned to a single HTML element. Class selector can be used when referencing several related elements.

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

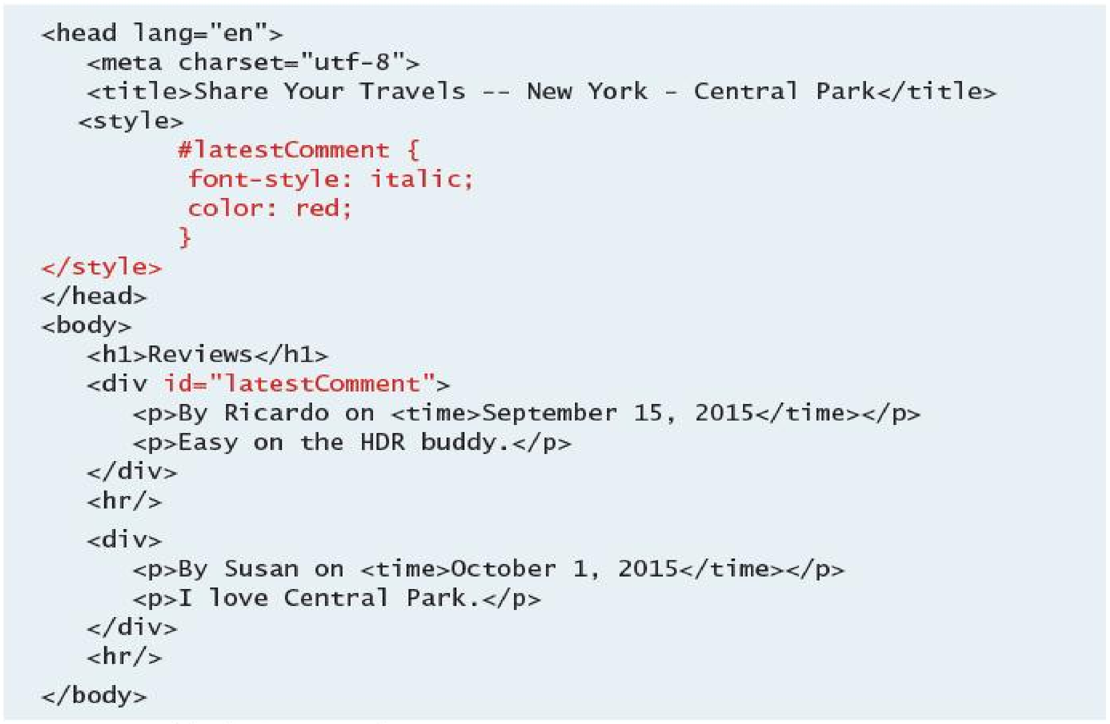


```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

Red and center-aligned heading

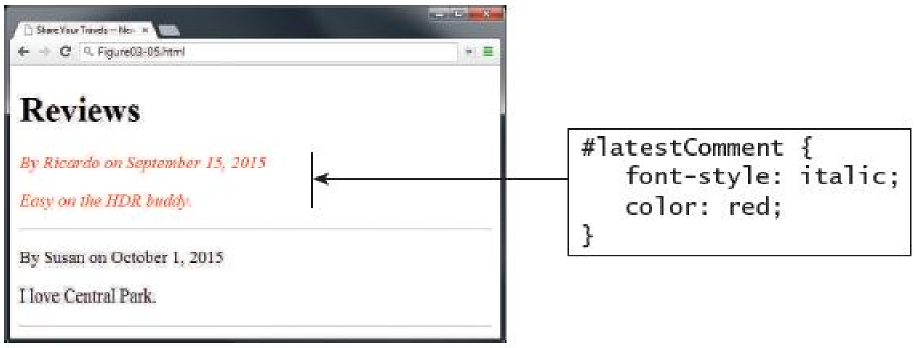
Red and center-aligned paragraph.

Example 2:



```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
  <style>
         #latestComment {
          font-style: italic;
          color: red;
         }
</style>
</head>
<body>
   <h1>Reviews</h1>
   <div id="latestComment">
      <p>By Ricardo on <time>September 15, 2015</time></p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <hr/>
   <div>
      <p>By Susan on <time>October 1, 2015</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>
</body>
```

Id selector example



Reviews

By Ricardo on September 15, 2015

Easy on the HDR buddy.

By Susan on October 1, 2015

I love Central Park.

```
#latestComment {
   font-style: italic;
   color: red;
}
```

Id selector example in browser

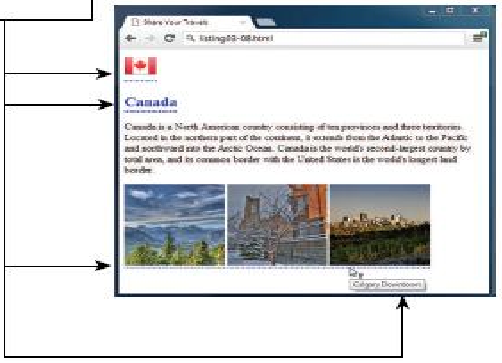
<u>5. Attribute Selectors</u>: An **attribute selector** provides a way to select HTML elements either by the presence of an element attribute or by the value of an attribute. This can be a very powerful technique, but because of uneven support by some of the browsers it is not widely used. Attribute selectors can be a very helpful technique in the styling of hyperlinks and images.

[title] { … } This will match any element in the document that has a title attribute.

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels</title>
     <style>
          [title] {
          cursor: help;
          padding-bottom: 3px;
          border-bottom: 2px dotted blue;
          text-decoration: none;
        }
   </style>
</head>
<body>
   <div>
      <img src="images/flags/CA.png" title="Canada Flag" />
      <h2><a href="countries.php?id=CA" title="see posts from Canada">
          Canada</a>
      </h2>
      <p>Canada is a North American country consisting of ... </p>
      <div>
         <img src="images/square/6114907897.jpg"
              title="At top of Sulphur Mountain" />
         <img src="images/square/6592317633.jpg"
              title="Grace Presbyterian Church" />
         <img src="images/square/6592914823.jpg"
              title="Calgary Downtown" />
     </div>
   </div>
</body>
```

Attribute selector example

```
[title] {
   cursor: help;
   padding-bottom: 3px;
   border-bottom: 2px dotted blue;
   text-decoration: none;
}
```



Canada is a North American country consisting of ten provinces and three territories. Located in the northern part of the continent, it extends from the Atlantic to the Pacific and northward into the Arctic Ocean. Canada is the world's second-largest country by total area, and its common border with the United States is the world's longest land border.

Attribute selector example in browser

Table summarizes some of the most common ways one can construct attribute selectors in CSS3.

Selector	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"`

TABLE Attribute Selectors

6. **Pseudo-Element and Pseudo-Class Selectors:** A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

Syntax

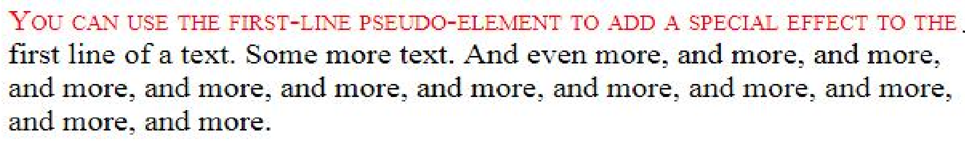
```
selector::pseudo-element {
property:value;
}
```

The most common use of this type of selectors is for targeting link states. By default, the browser displays link text blue and visited text links purple. Listing below illustrates the use of pseudo-class selectors to style not only the visited and unvisited link colors, but also the hover color, which is the color of the link when the mouse is over the link.

Do be aware that this state does not occur on touch screen devices. Note the syntax of pseudo-class selectors: the colon (:) followed by the pseudo-class selector name. Do be aware that a space is *not* allowed after the colon.

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
</style>
</head>
<body>

<p>You can use the first-line pseudo-element to add a
special effect to the first line of a text. Some more
text. And even more, and more, and more, and more, and
more, and more, and more, and more, and more, and more,
and more, and more.</p>

</body>
</html>
```

Output:



```
YOU CAN USE THE FIRST-LINE PSEUDO-ELEMENT TO ADD A SPECIAL EFFECT TO THE
first line of a text. Some more text. And even more, and more, and more,
and more, and more, and more, and more, and more, and more, and more,
and more, and more.
```

Only first line is set with css stvle

Selector	Type	Description
`a:link`	pseudo-class	Selects links that have not been visited
`a:visited`	pseudo-class	Selects links that have been visited
`:focus`	pseudo-class	Selects elements (such as text boxes or list boxes) that have the input focus.
`:hover`	pseudo-class	Selects elements that the mouse pointer is currently above.
`:active`	pseudo-class	Selects an element that is being activated by the user. A typical example is a link that is being clicked.
`:checked`	pseudo-class	Selects a form element that is currently checked. A typical example might be a radio button or a check box.
`:first-child`	pseudo-class	Selects an element that is the first child of its parent. A common use is to provide different styling to the first element in a list.
`:first-letter`	pseudo-element	Selects the first letter of an element. Useful for adding drop-caps to a paragraph.
`:first-line`	pseudo-element	Selects the first line of an element.

TABLE Common Pseudo-Class and Pseudo-Element Selectors

```
<head>
   <title>Share Your Travels</title>
  <style>
       a:link {
       text-decoration: underline;
       color: blue;
    }
       a:visited {
       text-decoration: underline;
       color: purple;
    }
       a:hover {
       text-decoration: none;
       font-weight: bold;
    }
       a:active {
       background-color: yellow;
    }
  </style>
</head>
<body>
    <p>Links are an important part of any web page. To learn more about
       links visit the <a href="#">W3C</a> website.</p>
   <nav>
    <ul>
       <li><a href="#">Canada</a></li>
       <li><a href="#">Germany</a></li>
       <li><a href="#">United States</a></li>
    </ul>
   </nav>
</body>
```

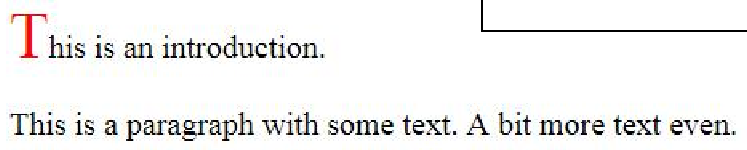
LISTING Styling a link using pseudo-class selectors

Pseudo class Selectors : Pseudo-elements can be combined with CSS classes.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.intro::first-letter {
  color: #ff0000;
  font-size:200%;
}
</style>
</head>
<body>

<p class="intro">This is an introduction.</p>
<p>This is a paragraph with some text. A bit more text
even.</p>

</body>
</html>
```

Output:

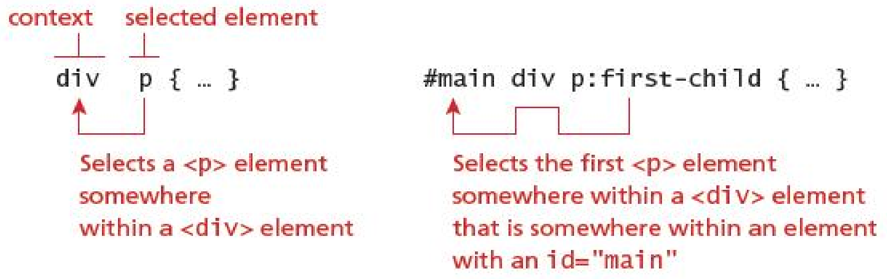
CSS only for selected para.



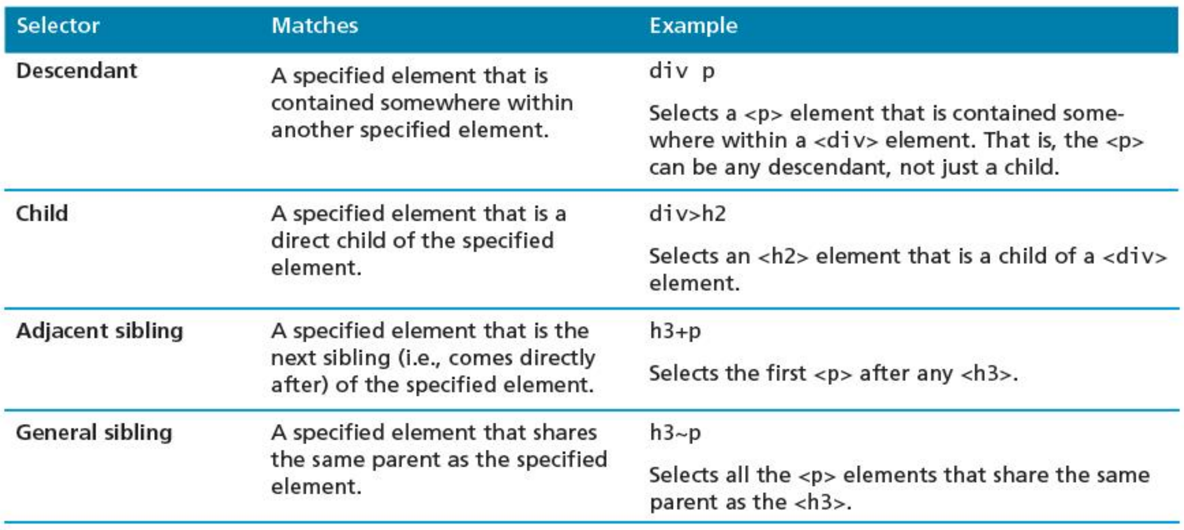
```
This is an introduction.

This is a paragraph with some text. A bit more text even.
```

7. **Contextual Selectors:** A **contextual selector** (in CSS3 also called **combinators**) allows selecting elements based on their *ancestors*, *descendants*, or *siblings*.
   That is, it selects elements based on their context or their relation to other elements in the document tree.

A **descendant selector** matches all elements that are contained within another element. The character used to indicate descendant selection is the space character.


context
selected element
div p { … }
Selects a <p> element somewhere within a <div> element
#main div p:first-child { … }
Selects the first <p> element somewhere within a <div> element that is somewhere within an element with an id="main"

Syntax of a descendant selection



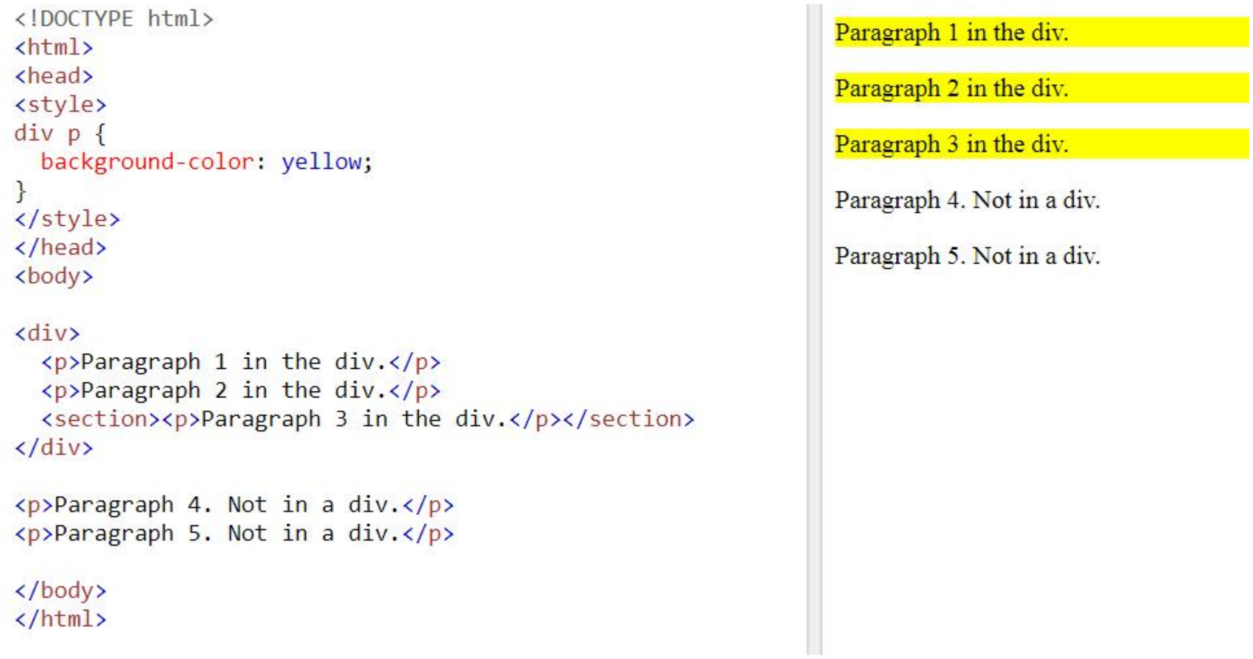
Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element.	`div p` Selects a <p> element that is contained some-where within a <div> element. That is, the <p> can be any descendant, not just a child.
Child	A specified element that is a direct 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	A specified element that shares the same parent as the specified element.	`h3~p` Selects all the <p> elements that share the same parent as the <h3>.

Contextual Selectors

A Combinator/Contextual Selector is something that explains the relationship between the selectors. A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator. There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

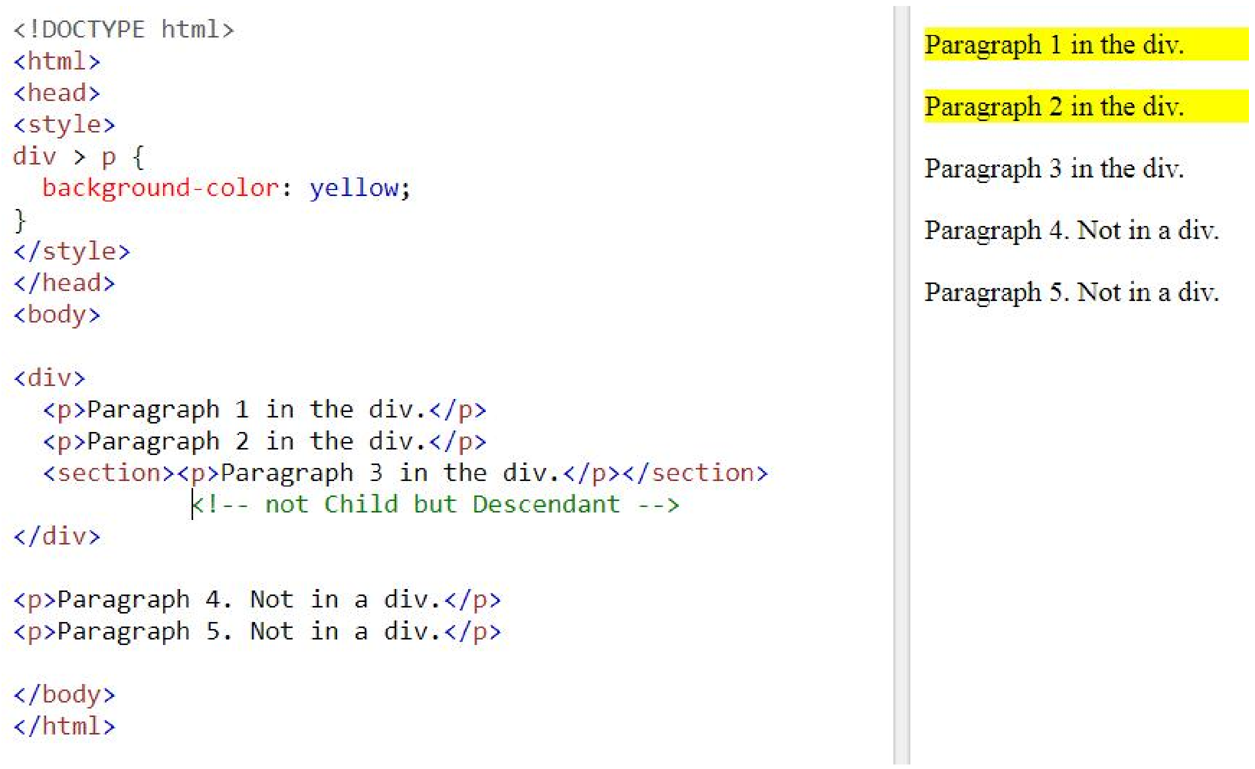
Descendant Selector : The descendant selector matches all elements that are descendants of a specified element. The following example selects all <p> elements inside <div> elements:



```
<!DOCTYPE html>
<html>
<head>
<style>
div p {
  background-color: yellow;
}
</style>
</head>
<body>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

```
Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.
```

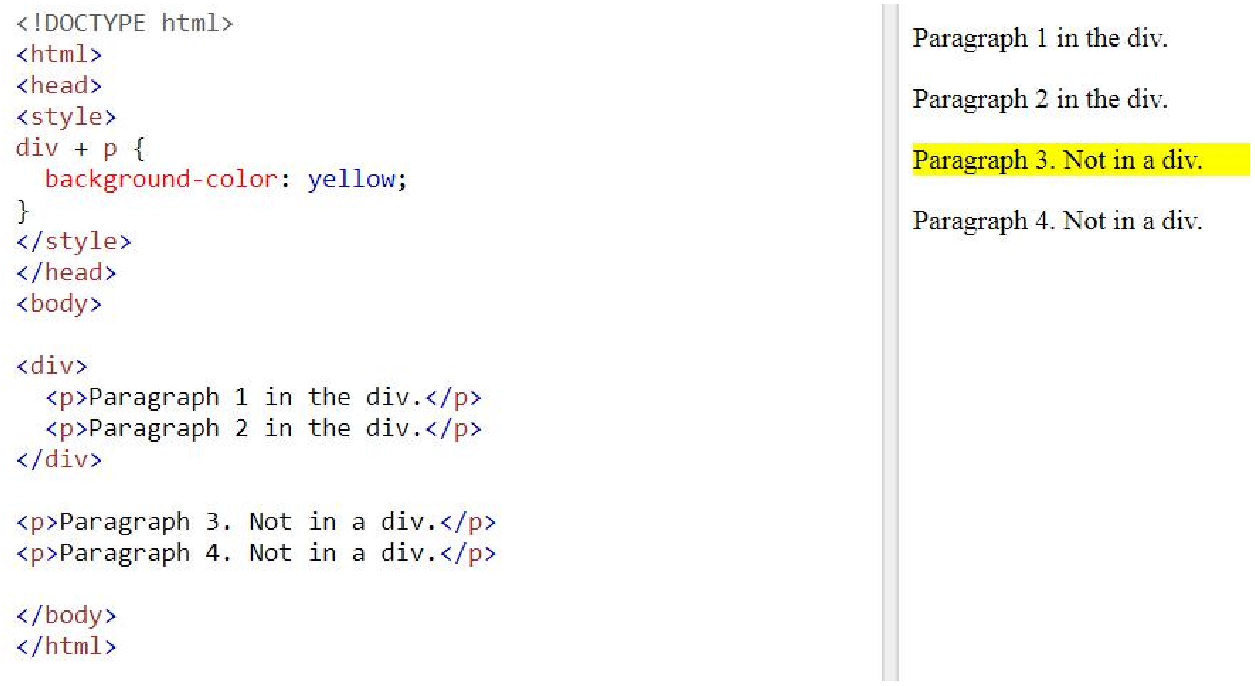
<u>Child Selector:</u> The child selector selects all elements that are the immediate children of a specified element. The following example selects all <p> elements that are immediate children of a <div> element.



```
<!DOCTYPE html>
<html>
<head>
<style>
div > p {
  background-color: yellow;
}
</style>
</head>
<body>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
            <!-- not Child but Descendant -->
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

```
Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.
```

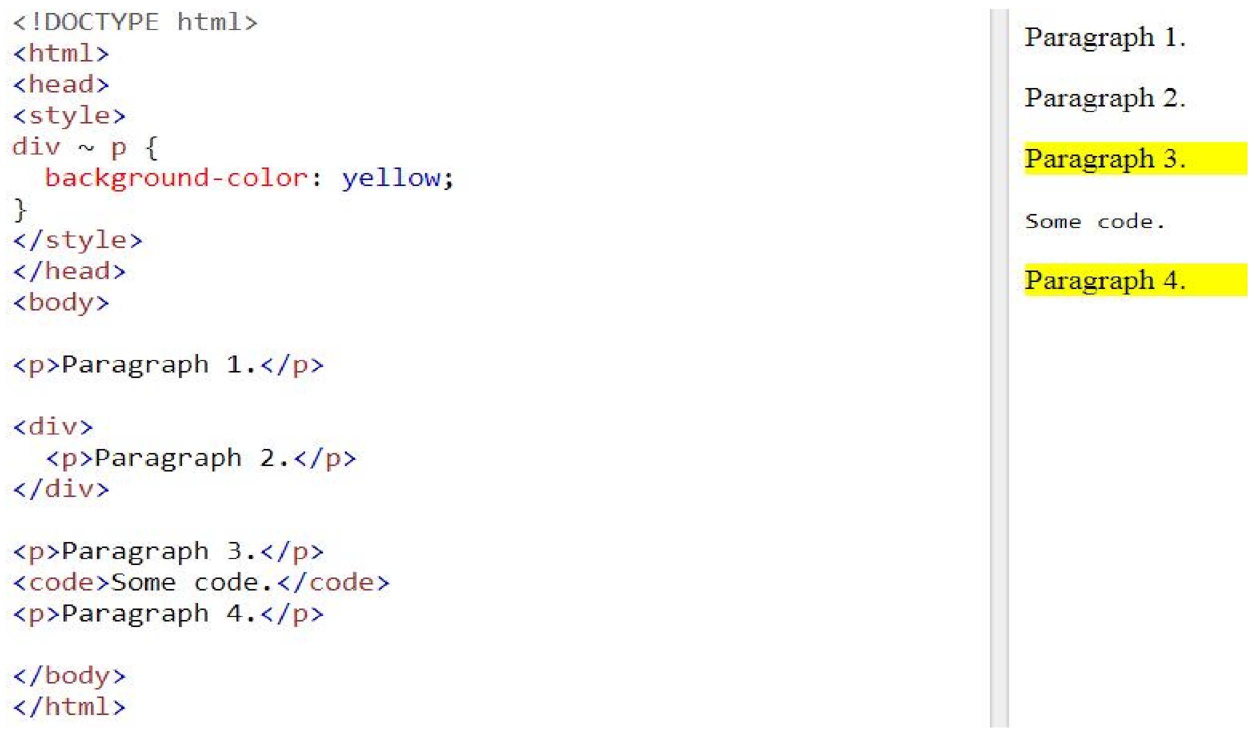
<u>Adjacent Sibling Selector:</u> The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element. Sibling elements must have the same parent element, and "adjacent" means "immediately following". The following example selects all <p> elements that are placed immediately after <div> elements:



```
<!DOCTYPE html>
<html>
<head>
<style>
div + p {
  background-color: yellow;
}
</style>
</head>
<body>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
</div>

<p>Paragraph 3. Not in a div.</p>
<p>Paragraph 4. Not in a div.</p>

</body>
</html>
```

```
Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3. Not in a div.

Paragraph 4. Not in a div.
```

<u>General Sibling Selector:</u> The general sibling selector selects all elements that are siblings of a specified element. The following example selects all <p> elements that are siblings of <div> elements.



```
<!DOCTYPE html>
<html>
<head>
<style>
div ~ p {
  background-color: yellow;
}
</style>
</head>
<body>

<p>Paragraph 1.</p>

<div>
  <p>Paragraph 2.</p>
</div>

<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>

</body>
</html>
```

```
Paragraph 1.

Paragraph 2.

Paragraph 3.

Some code.

Paragraph 4.
```

2.5 The Cascade: How Styles Interact

Why Conflict Happens?

Because

- There are three different types of style sheets (author-created, user-defined, and the default browser style sheet).
- Author-created style sheets can define multiple rules for the same HTML element.

CSS has a system to help the browser determine how to display elements when different style rules conflict.

The **cascade** is an algorithm that defines how to combine property values originating from different sources. It lies at the core of CSS, as emphasized by the name: *Cascading* Style Sheets.

The "Cascade" in CSS refers to how conflicting rules are handled. The downward movement of water down a cascade is meant to be analogous to how a given style rule will continue to take precedence with child elements.

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

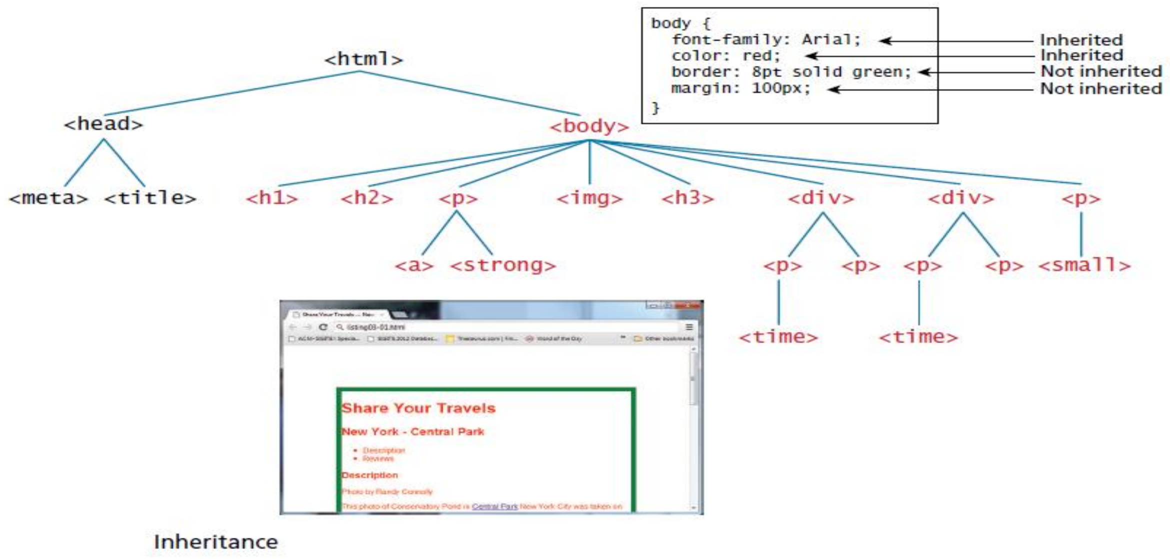
- Inheritance
- Specificity
- Location

Inheritance - Inheritance is the first of cascading principles. Many CSS properties affect not only themselves but their descendants as well. Font, color, list, and text properties are inheritable. Layout, sizing, border, background and spacing properties are not inheritable.

Figures illustrate CSS inheritance. In the first example, only some of the property rules are inherited for the <body> element. That is, only the body element will have a thick green border and the 100-px margin, however all the text in the other elements in the document will be in the Arial font and colored red.

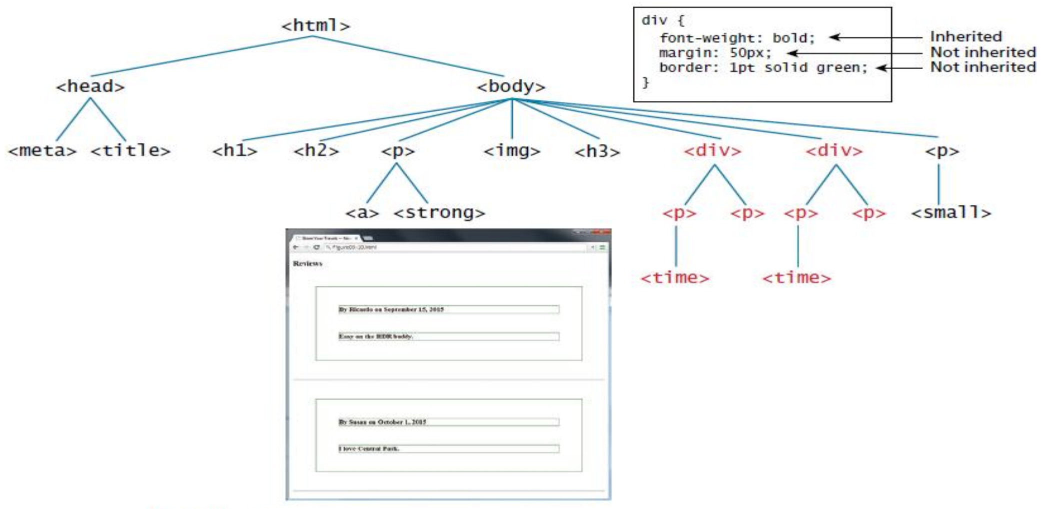
In the second example in Figure 2, it is assumed that there is no longer the body styling but instead will have a single style rule that styles *all* the <div> elements. The <p> and <time> elements within the <div> inherit the bold font-weight property but not the margin or border styles.

However, it is possible to tell elements to inherit properties that are normally not inheritable, as shown in 3rd Figure. In comparison to Figure 2nd, <p> elements nested within the <div> elements now inherit the border and margins of their parent.



```
body {
   font-family: Arial;
   color: red;
   border: 8pt solid green;
   margin: 100px;
}
```

Figure 1: Inhereitence



```
div {
  font-weight: bold;
  margin: 50px;
  border: 1pt solid green;
}
```

Reviews

By Ricardo on September 15, 2015

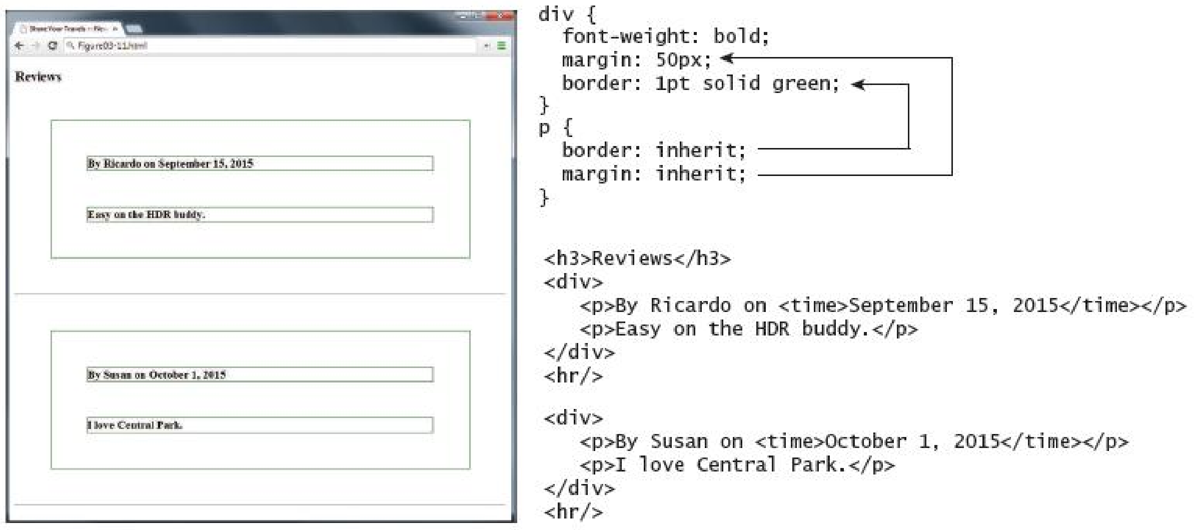
Easy on the HDR buddy.

By Susan on October 1, 2015

I love Central Park.

More Inheritance

Figure 2: More Inheritance



Reviews

By Ricardo on September 15, 2015

Easy on the HDR buddy.

By Susan on October 1, 2015

I love Central Park.

```
div {
  font-weight: bold;
  margin: 50px;
  border: 1pt solid green;
}
p {
  border: inherit;
  margin: inherit;
}
```

```
<h3>Reviews</h3>
<div>
   <p>By Ricardo on <time>September 15, 2015</time></p>
   <p>Easy on the HDR buddy.</p>
</div>
<hr/>

<div>
   <p>By Susan on <time>October 1, 2015</time></p>
   <p>I love Central Park.</p>
</div>
<hr/>
```

Using the `inherit` value

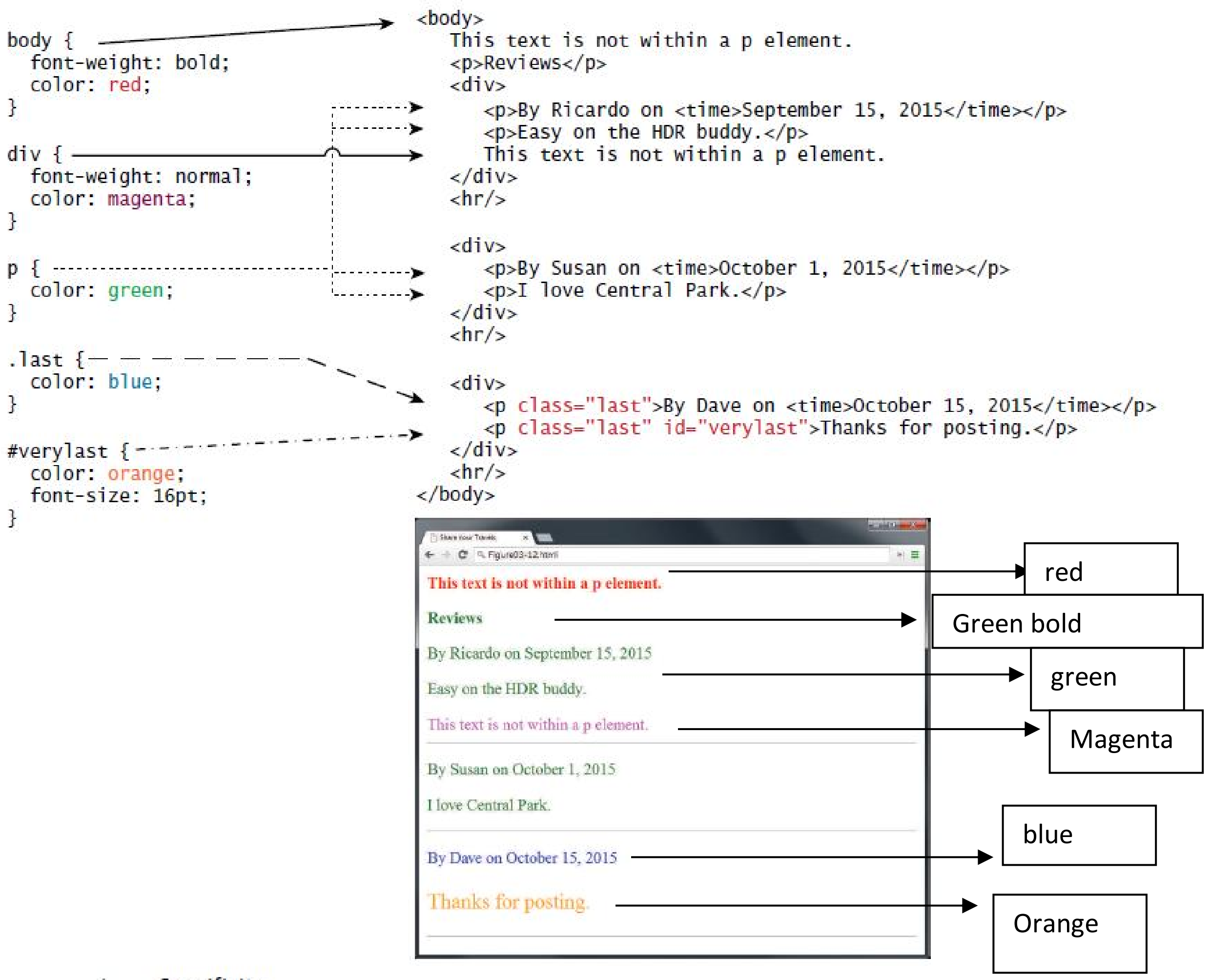
- **Specificity**

Specificity is how the browser determines which style rule takes precedence when more than one style rule could be applied to the same element. In CSS, the more specific the selector, the more it takes precedence (i.e., overrides the previous definition).

The way that specificity works in the browser is that the browser assigns a weight to each style rule, when several rules apply the one with the greatest weight takes precedence.

In the example shown in Figure below the color and font-weight properties defined in the <body> element are inheritable and thus potentially applicable to all the child elements contained within it. However, because the <div> and <p> elements also have the same properties set, they *override* the value defined for the <body> element because their selectors (<div> and <p>) are more specific. As a consequence, their font-weight is normal and their text is colored either green or magenta.

As in Figure below, class selectors take precedence over element selectors, and id selectors take precedence over class selectors. The precise algorithm the browser is supposed to use to determine specificity is quite complex.



```
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>September 15, 2015</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>October 1, 2015</time></p>
      <p>I love Central Park.</p>
   </div>
   <hr/>

   <div>
      <p class="last">By Dave on <time>October 15, 2015</time></p>
      <p class="last" id="verylast">Thanks for posting.</p>
   </div>
   <hr/>
</body>
```

This text is not within a p element.

Reviews

By Ricardo on September 15, 2015

Easy on the HDR buddy.

This text is not within a p element.

By Susan on October 1, 2015

I love Central Park.

By Dave on October 15, 2015

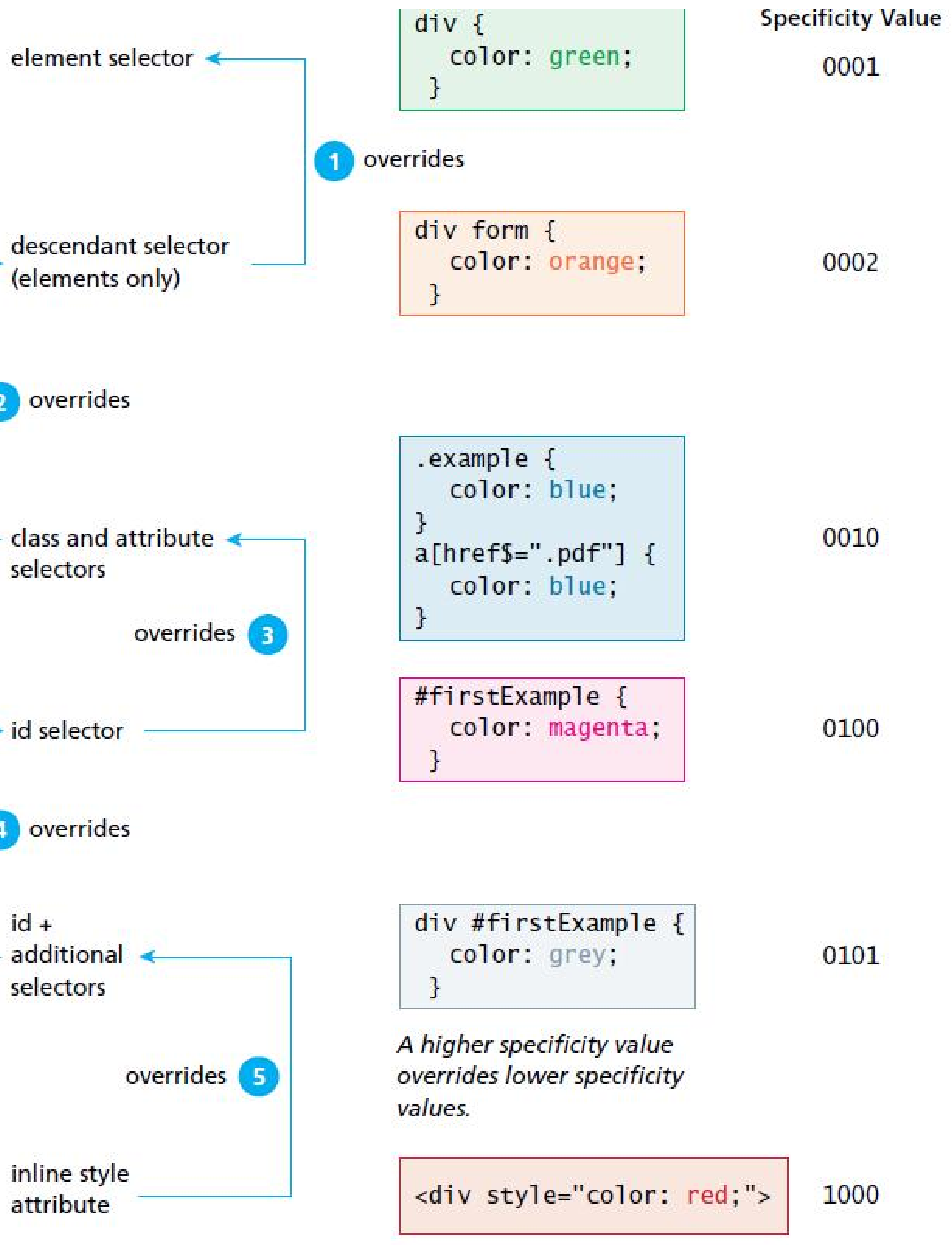
Thanks for posting.

Specificity

- **Location**

When inheritance and specificity cannot determine style precedence, the principle of **location** will be used. The principle of location is that 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.

Similarly, an embedded style will override an equally specific rule defined in an external author style sheet if it appears after the external sheet's <link> element. Styles defined in external author style sheet X will override styles in external author style sheet Y if X's <link> element is after Y's in the HTML document.


Specificity Value
element selector
1 overrides
descendant selector (elements only)
2 overrides
class and attribute selectors
overrides 3
id selector
4 overrides
id + additional selectors
overrides 5
inline style attribute
0001
0002
0010
0100
0101
1000
A higher specificity value overrides lower specificity values.

```
div {
  color: green;
 }
```

```
div form {
  color: orange;
 }
```

```
.example {
  color: blue;
}
a[href$=".pdf"] {
  color: blue;
}
```

```
#firstExample {
  color: magenta;
 }
```

```
div #firstExample {
  color: grey;
 }
```

```
<div style="color: red;">
```

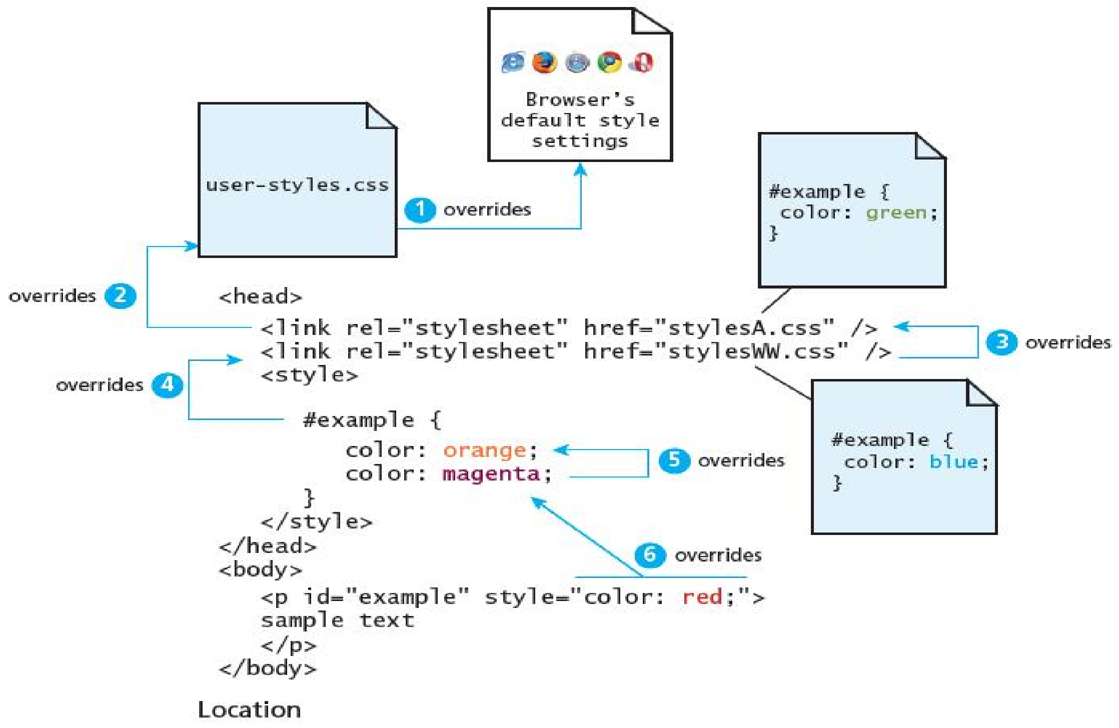
Specificity algorithm

Similarly, the algorithm that is used to determine specificity of any given element is defined by the W3C as follows.

■ First count 1 if the declaration is from a "style" attribute in the HTML, 0 otherwise (let that value = a).

■ Count the number of ID attributes in the selector (let that value = b).
■ Count the number of class selectors, attribute selectors, and pseudo-classes in the selector (let that value = c).
■ Count the number of element names and pseudo-elements in the selector (let that value = d).
■ finally, concatenate the four numbers a+b+c+d together to calculate the selector's specificity.
Knowing the specificity algorithm is useful to debug a CSS problem. When the same style property is defined multiple times within a single declaration block, the last one will take precedence.
Figure illustrates how location affects precedence.

What would be the color of the sample text if there wasn't an inline style definition?
It would be magenta.



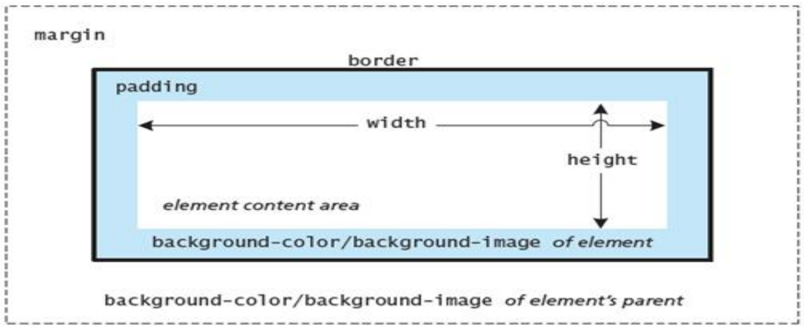
```
#example {
 color: green;
}
```

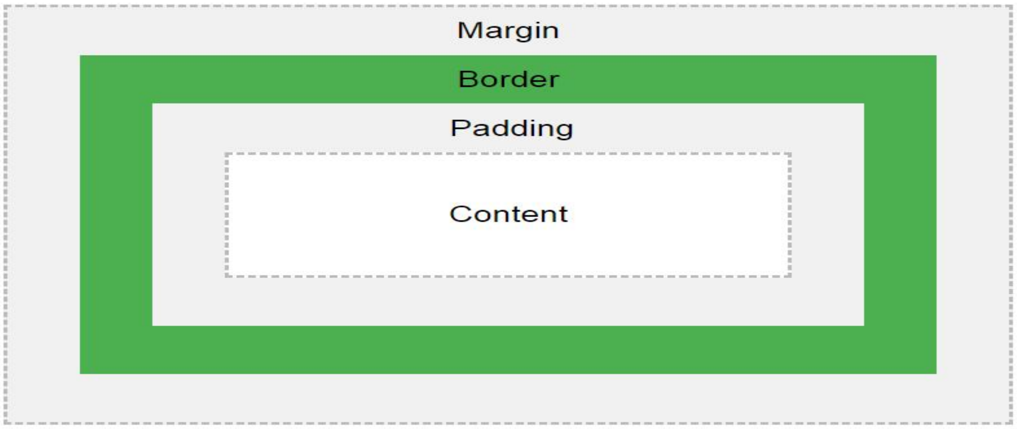
```
#example {
 color: blue;
}
```

```
<head>
   <link rel="stylesheet" href="stylesA.css" />
   <link rel="stylesheet" href="stylesWW.css" />
   <style>
      #example {
         color: orange;
         color: magenta;
      }
   </style>
</head>
<body>
   <p id="example" style="color: red;">
   sample text
   </p>
</body>
```

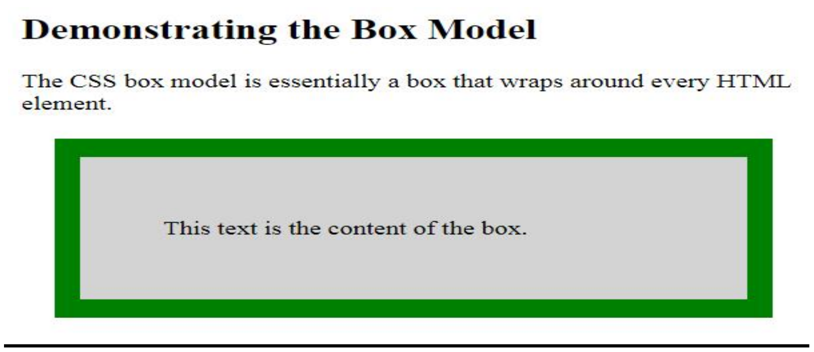
2.6 The Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" refers to design and layout. The CSS box model is essentially a box that wraps around every HTML element. It consists of: Borders, margins, borders, padding, and the actual content. The image below illustrates the box model:


margin
border
padding
width
height
element content area
background-color/background-image of element
background-color/background-image of element's parent



```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
</style>
</head>
<body>

<h2>Demonstrating the Box Model</h2>

<p>The CSS box model is essentially a box that wraps
around every HTML element. </p>

<div>This text is the content of the box. </div>

</body>
</html>
```



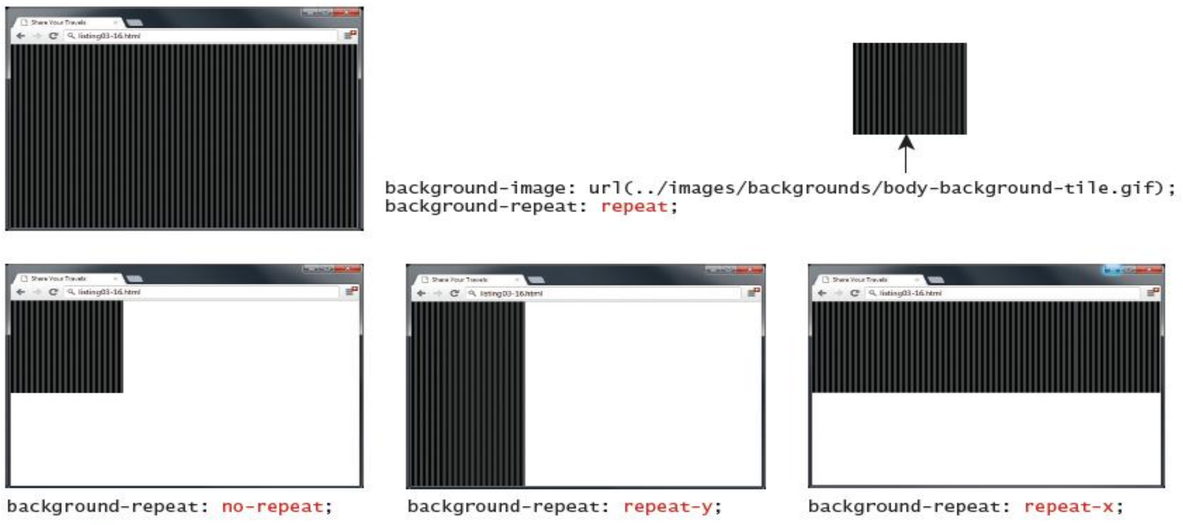
Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element.

This text is the content of the box.

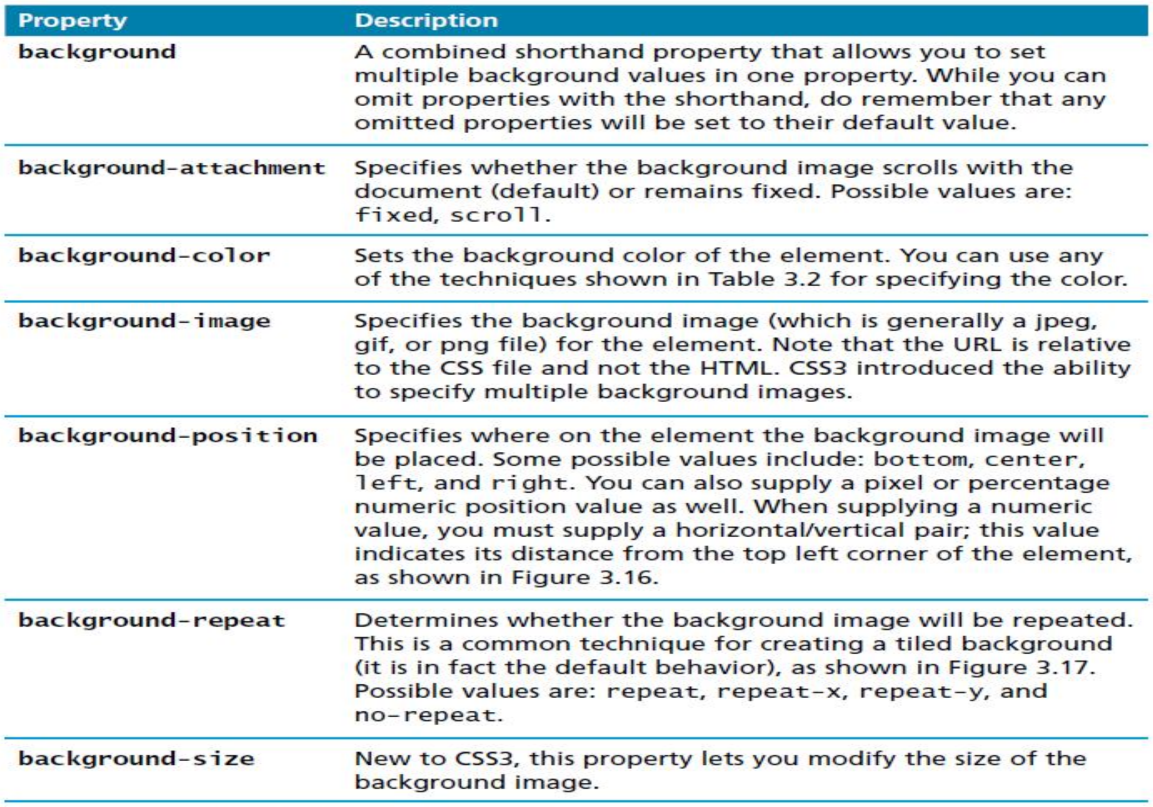
- **Background**

As can be seen in Figure above, the background color or image of an element fills an element out to its border. In contemporary web design, it has become extremely common to use CSS to display purely presentational images (such as background gradients and patterns, decorative images, etc.) rather than using the <img> element.


background-image: url(../images/backgrounds/body-background-tile.gif);
background-repeat: repeat;
background-repeat: no-repeat;
background-repeat: repeat-y;
background-repeat: repeat-x;

Background repeat

Table lists the most common background properties.

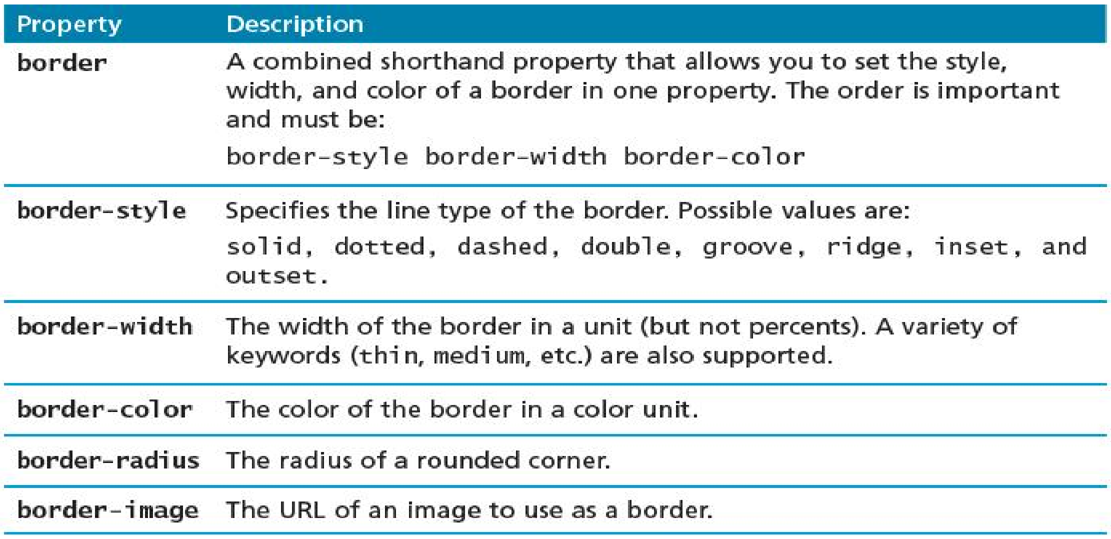


Property	Description
background	A combined shorthand property that allows you to set multiple background values in one property. While you can omit properties with the shorthand, do remember that any omitted properties will be set to their default value.
background-attachment	Specifies whether the background image scrolls with the document (default) or remains fixed. Possible values are: `fixed`, `scroll`.
background-color	Sets the background color of the element. You can use any of the techniques shown in Table 3.2 for specifying the color.
background-image	Specifies the background image (which is generally a jpeg, gif, or png file) for the element. Note that the URL is relative to the CSS file and not the HTML. CSS3 introduced the ability to specify multiple background images.
background-position	Specifies where on the element the background image will be placed. Some possible values include: `bottom`, `center`, `left`, and `right`. You can also supply a pixel or percentage numeric position value as well. When supplying a numeric value, you must supply a horizontal/vertical pair; this value indicates its distance from the top left corner of the element, as shown in Figure 3.16.
background-repeat	Determines whether the background image will be repeated. This is a common technique for creating a tiled background (it is in fact the default behavior), as shown in Figure 3.17. Possible values are: `repeat`, `repeat-x`, `repeat-y`, and `no-repeat`.
background-size	New to CSS3, this property lets you modify the size of the background image.

Common Background Properties

- **Borders**

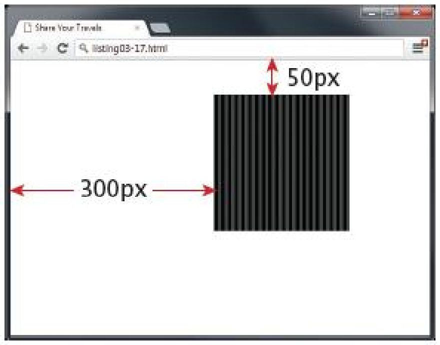
The border-style property specifies what kind of border to display. Borders provide a way to visually separate elements. Borders can be around all four sides of an element, or just one, two, or three of the sides. Table below lists the various border properties.



Property	Description
border	A combined shorthand property that allows you to set the style, width, and color of a border in one property. The order is important and must be: `border-style border-width border-color`
border-style	Specifies the line type of the border. Possible values are: `solid, dotted, dashed, double, groove, ridge, inset, and outset.`
border-width	The width of the border in a unit (but not percents). A variety of keywords (`thin`, `medium`, etc.) are also supported.
border-color	The color of the border in a color unit.
border-radius	The radius of a rounded corner.
border-image	The URL of an image to use as a border.

Border Properties

Border widths are perhaps the one exception to the general advice against using the pixel measure. Using em units or percentages for border widths can result in unpredictable widths as the different browsers use different algorithms (some round up, some round down) as the zoom level increases or decreases. For this reason, border widths are almost always set to pixel units.



```
body {
        background: white url(../images/backgrounds/body-background-tile.gif) no-repeat;
        background-position: 300px 50px;
}
```

Background position

Margins and Padding

Margins and padding are essential properties for adding white space to a web page, which can help differentiate one element from another. Figure illustrates how these two properties can be used to provide spacing and element differentia.

With border, margin, and padding properties, it is possible to set the properties for one or more sides of the element box in a single property, or to set them individually using separate properties.

For instance, the side properties can be set 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, all four sides can be set to a single value via:

border-color: red; /* *sets all four sides to red* ***/**

Or all four sides can be set to different values via:

border-color: red green orange blue;

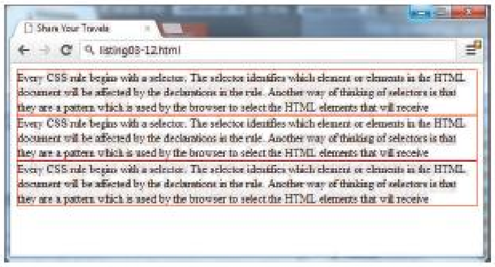
Margins add spacing around an element's content, while padding adds spacing within elements. Borders divide the margin area from the padding area.Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the larger of the two margins. This does not happen on horizontal (left and right) margins. Only vertical (top and bottom) margins.

The W3C specification defines this behavior as **collapsing margins**: *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.*

Margin collapsing occurs in three basic cases:

1. **Adjacent siblings** The margins of adjacent siblings are collapsed (except when the latter sibling needs to be cleared past floats).
2. **No content separating parent and descendants** If there is no border, padding, inline part, block formatting context created, or *clearance* to separate the margin-top of a block from the margin-top of one or more of its descendant blocks; or no border, padding, inline content, height, min-height, or max-height to separate the margin-bottom of a block from the margin-bottom of one or more of its descendant blocks, then those margins collapse. The collapsed margin ends up outside the parent.
3. **Empty blocks** If there is no border, padding, inline content, height, or min-height to separate a block's margin-top from its margin-bottom, then its top and bottom margins collapse.

Figure depicts Margin Collapsing:

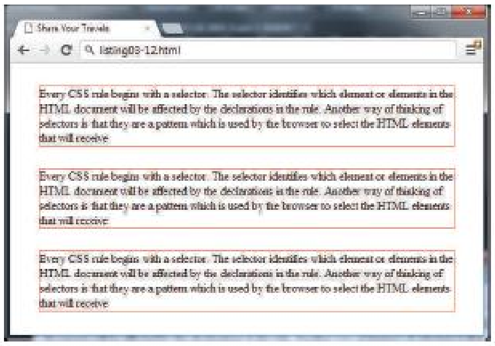


Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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

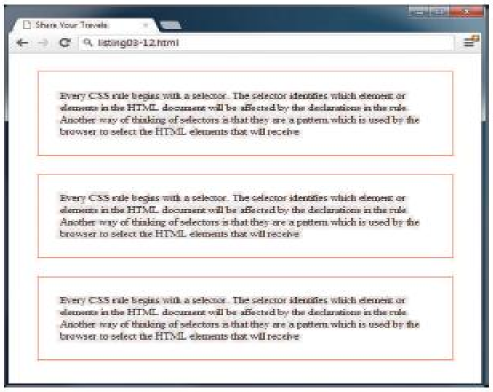


Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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



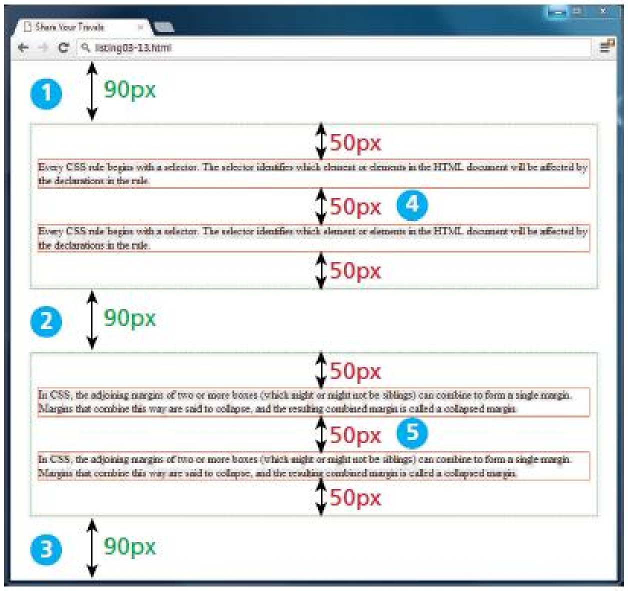
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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

Borders, margins, and padding provide element spacing and differentiation



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.

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.

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.

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.

```
<div>
  <p>Every CSS rule ...</p>
  <p>Every CSS rule ...</p>
</div>
<div>
  <p>In CSS, the adjoining ... </p>
  <p>In CSS, the adjoining ... </p>
</div>
```

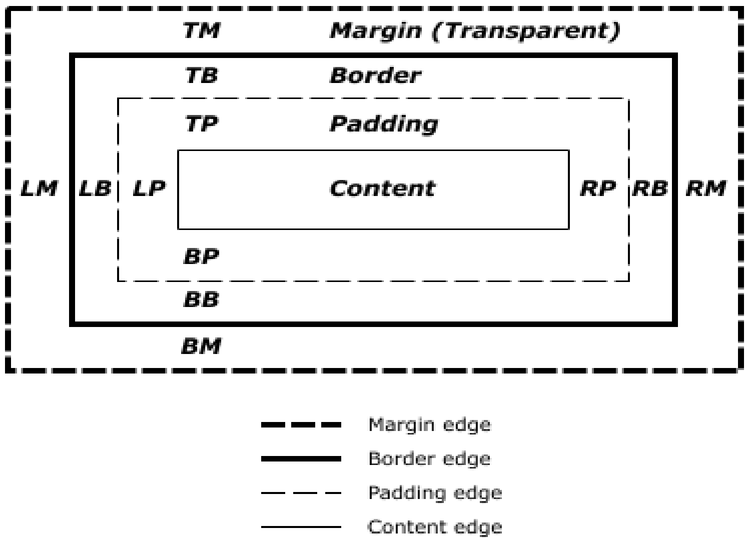
```
div {
   border: dotted 1pt green;
   padding: 0;
   margin: 90px 20px;
}
```

```
p {
   border: solid 1pt red;
   padding: 0;
   margin: 50px 20px;
}
```

Collapsing vertical margins

Box Dimensions

Each box has a content area (e.g., text, an image, etc.) and optional surrounding padding, border, and margin areas, the size of each area is specified by properties defined below:



i

The margin, border, and padding can be broken down into top, right, bottom, and left segments (e.g., in the diagram, "LM" for left margin, "RP" for right padding, "TB" for top border, etc.).

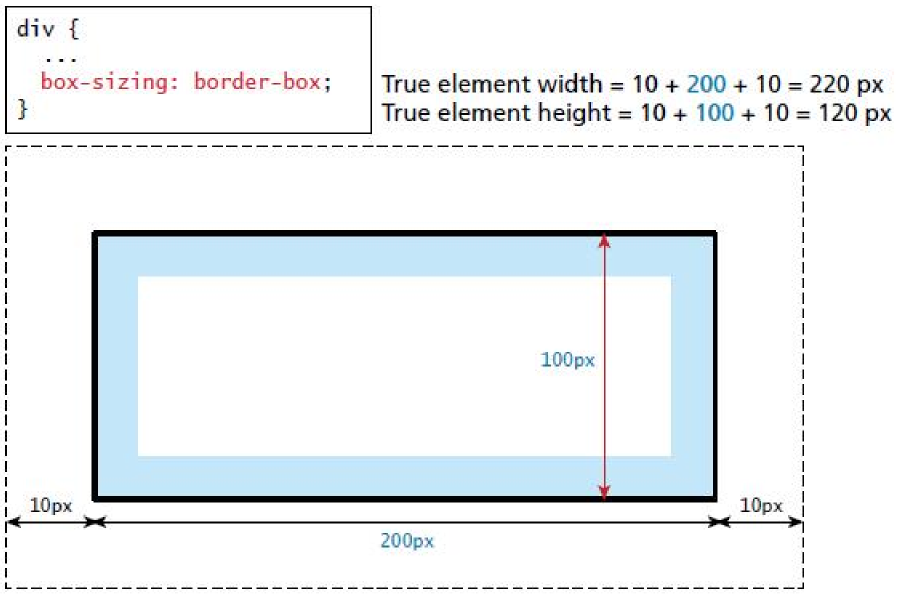
The perimeter of each of the four areas (content, padding, border, and margin) is called an "edge", so each box has four edges:

- **Content edge or inner edge:** The content edge surrounds the rectangle given by the width and height of the box, which often depend on the element's rendered content. The four content edges define the box's content box.
- **Padding edge:** The padding edge surrounds the box padding. If the padding has 0 width, the padding edge is the same as the content edge. The four padding edges define the box's padding box.
- **Border edge:** The border edge surrounds the box's border. If the border has 0 width, the border edge is the same as the padding edge. The four border edges define the box's border box.
- **Margin edge or Outer edge:** The margin edge surrounds the box margin. If the margin has 0 widths, the margin edge is the same as the border edge. The four margin edges define the box's margin box.

Each edge may be broken down into a top, right, bottom, and left edge.

The dimensions of the content area of a box — the content width and content height — depend on several factors: whether the element generating the box has the 'width' or 'height' property set, whether the box contains text or other boxes, whether the box is a table, etc.

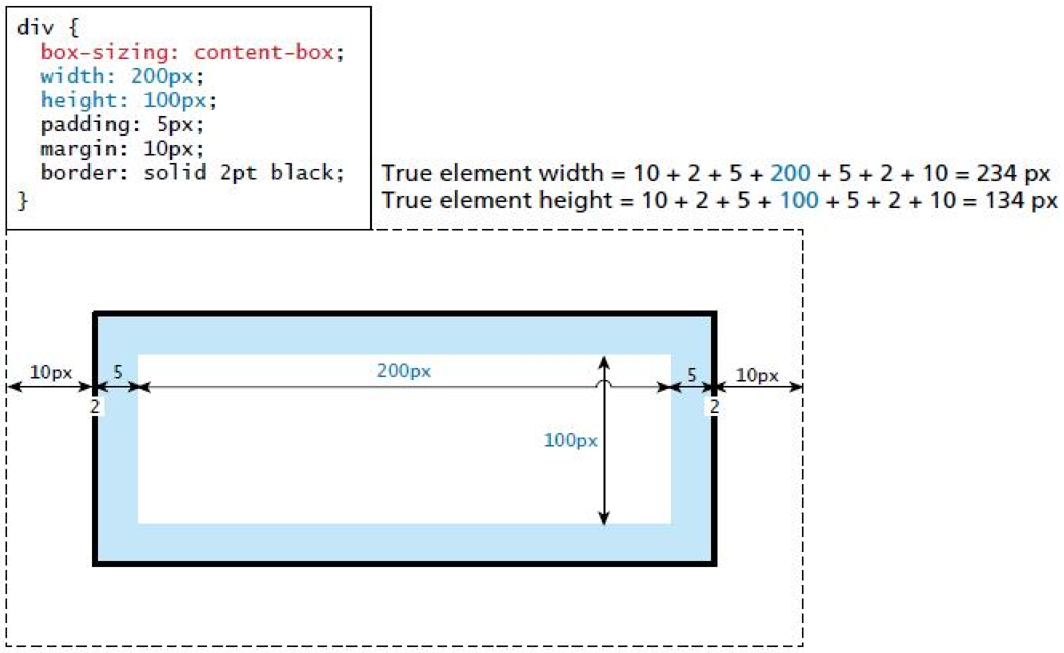
Padding+ Width + border = actual width of an element
Padding+ Height + padding + border = actual height of an element



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

True element width = 10 + 200 + 10 = 220 px
True element height = 10 + 100 + 10 = 120 px

Calculating an element's true size



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

True element width = 10 + 2 + 5 + 200 + 5 + 2 + 10 = 234 px
True element height = 10 + 2 + 5 + 100 + 5 + 2 + 10 = 134 px

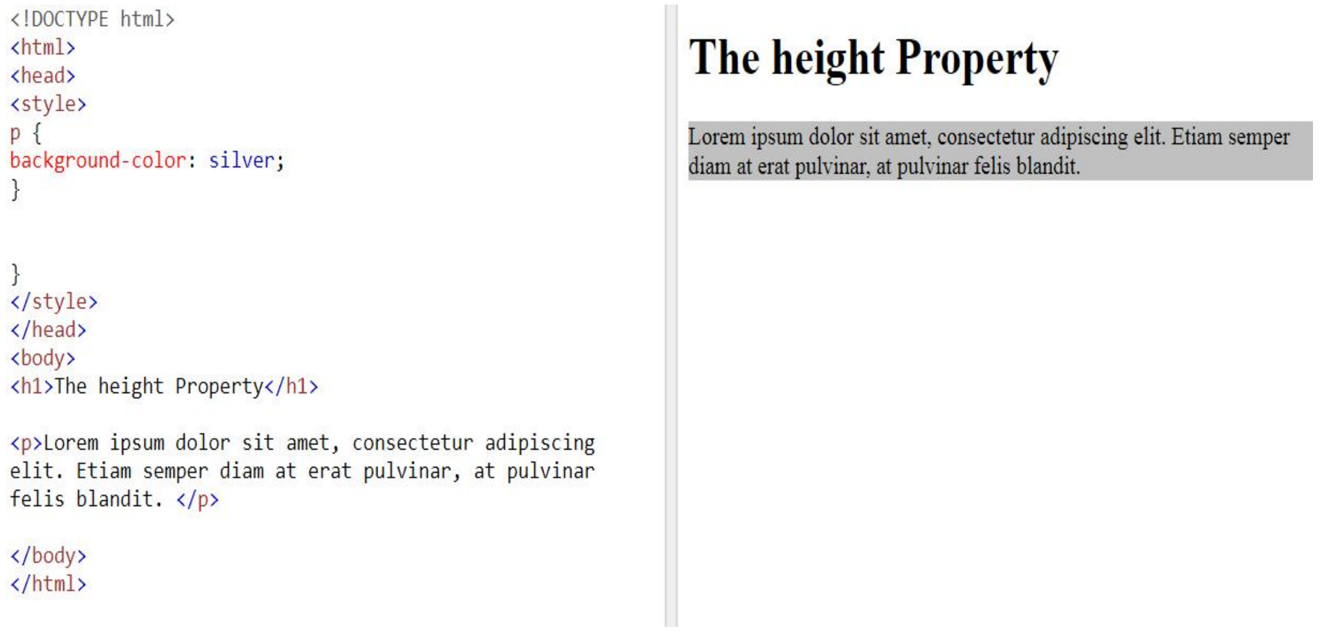
Height Property:

The height property sets the height of an element. The height of an element does not include padding, borders, or margins! If height: auto; the element will automatically adjust its height to allow its content to be displayed correctly. If height is set to a numeric value (like pixels, (r)em, percentages) then if the content does not fit within the specified height, it will overflow. How the container will handle the overflowing content is defined by the overflow property.

Syntax height: auto | length | initial | inherit;

Property Values

Value	Description
auto	The browser calculates the height. This is default
length	Defines the height in px, cm, etc. Read about length units
%	Defines the height in percent of the containing block

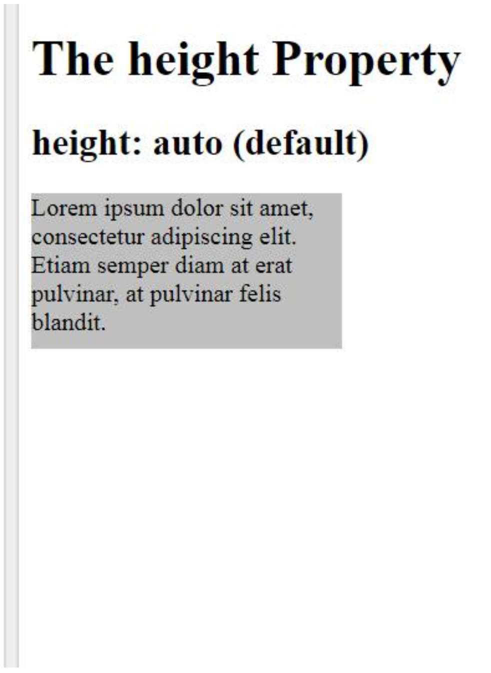


```
<!DOCTYPE html>
<html>
<head>
<style>
p {
background-color: silver;
}


}
</style>
</head>
<body>
<h1>The height Property</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Etiam semper diam at erat pulvinar, at pulvinar
felis blandit. </p>

</body>
</html>
```

The height Property

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
background-color: silver;
width: 200px;
height: 100px;
}


</style>
</head>
<body>
<h1>The height Property</h1>

<h2>height: auto (default)</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Etiam semper diam at erat pulvinar, at pulvinar
felis blandit. </p>

</body>
</html>
```

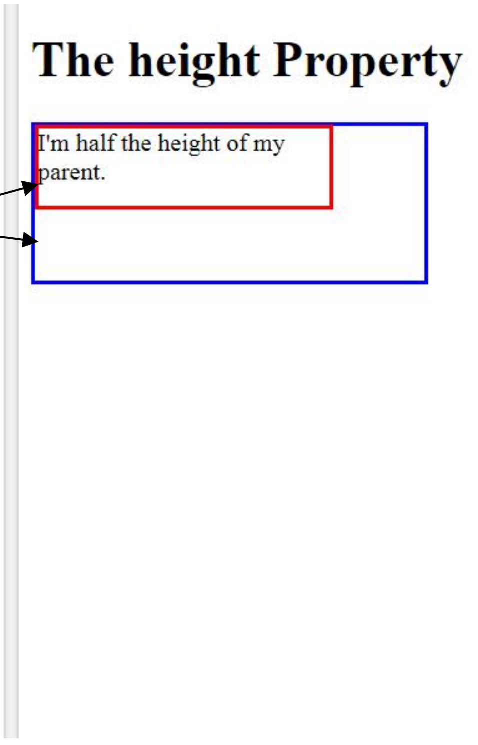


The height Property

height: auto (default)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.

```
<!DOCTYPE html>
<html>
<head>
<style>
#parent {
  height: 100px;
  width: 250px;
  border: 2px solid blue;
}

#child {
  height: 50%;
  width: 75%;
  border: 2px solid red;
}
</style>
</head>
<body>
<h1>The height Property</h1>

<div id="parent">
  <div id="child">I'm half the height of my parent.</div>
</div>

</body>
</html>
```



The height Property

I'm half the height of my parent.

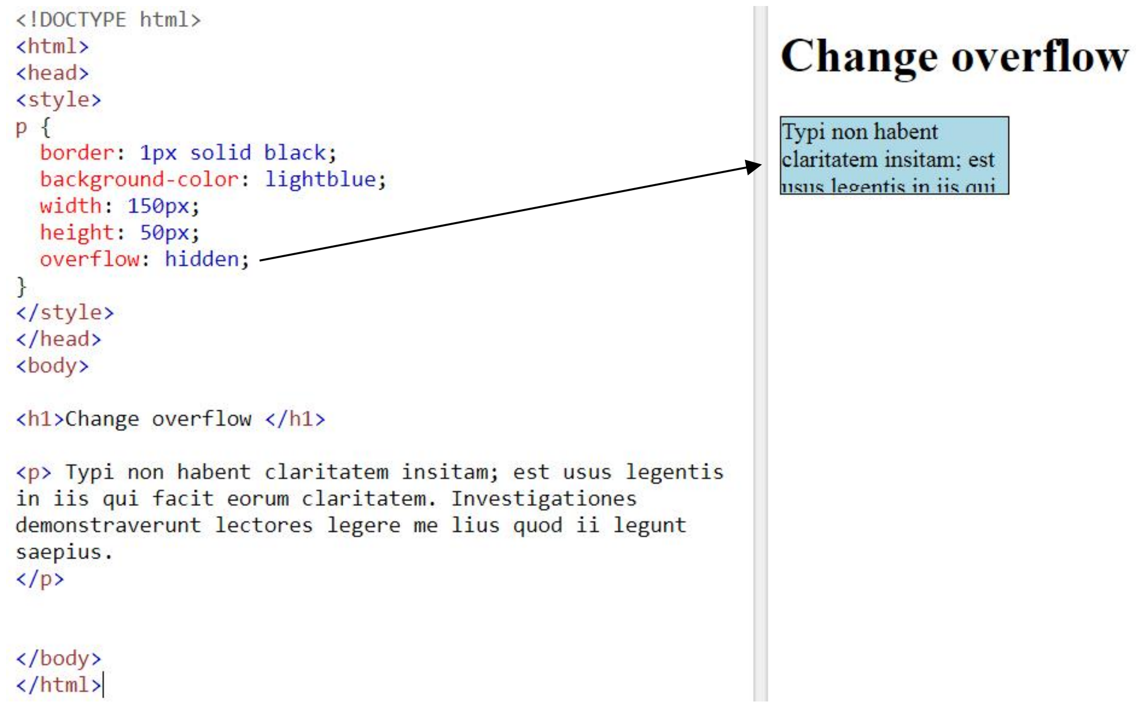
Overflow Property: The overflow property specifies what should happen if content overflow an element's box. This property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.

Note: The overflow property only works for block elements with a specified height.

Syntax overflow: visible | hidden | scroll |auto | initial |inherit;

Property Values

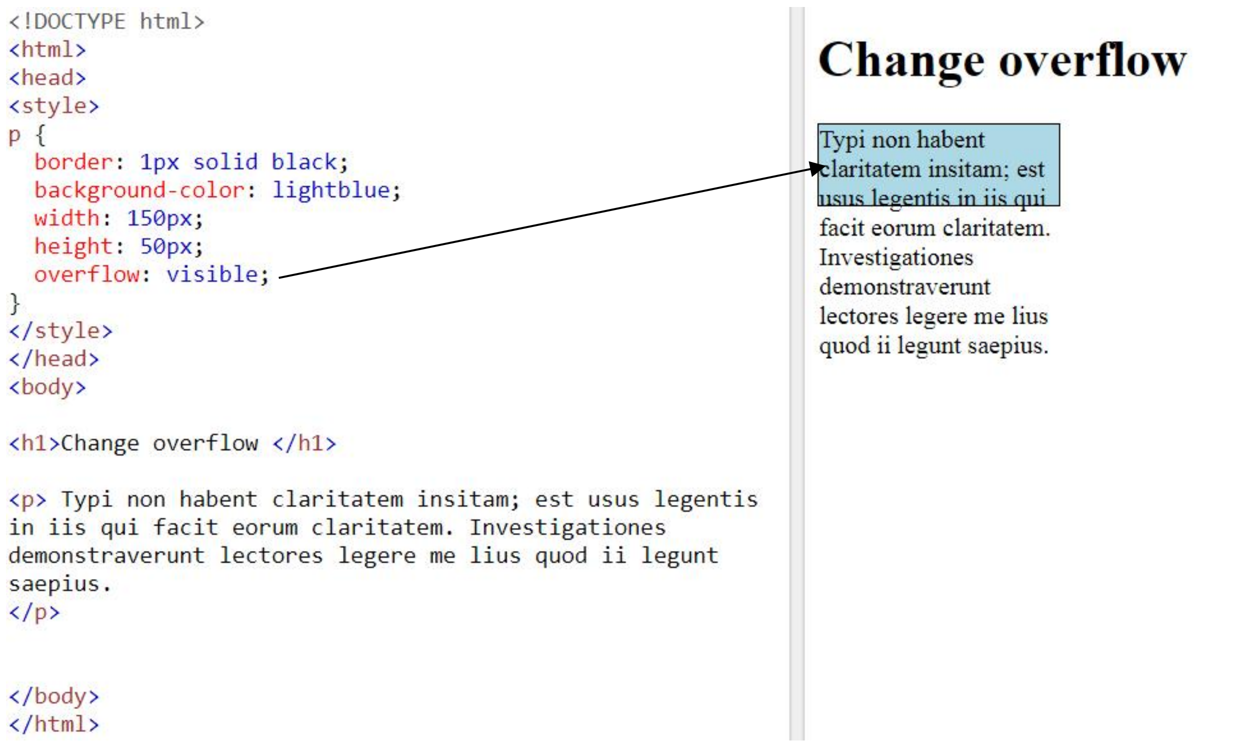
Value	**Description**
visible	The overflow is not clipped. It renders outside the element's box. This is default
hidden	The overflow is clipped, and the rest of the content will be invisible
scroll	The overflow is clipped, but a scroll-bar is added to see the rest of the content
auto	If overflow is clipped, a scroll-bar should be added to see the rest of the content



```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  border: 1px solid black;
  background-color: lightblue;
  width: 150px;
  height: 50px;
  overflow: hidden;
}
</style>
</head>
<body>

<h1>Change overflow </h1>

<p> Typi non habent claritatem insitam; est usus legentis
in iis qui facit eorum claritatem. Investigationes
demonstraverunt lectores legere me lius quod ii legunt
saepius.
</p>


</body>
</html>
```

Change overflow

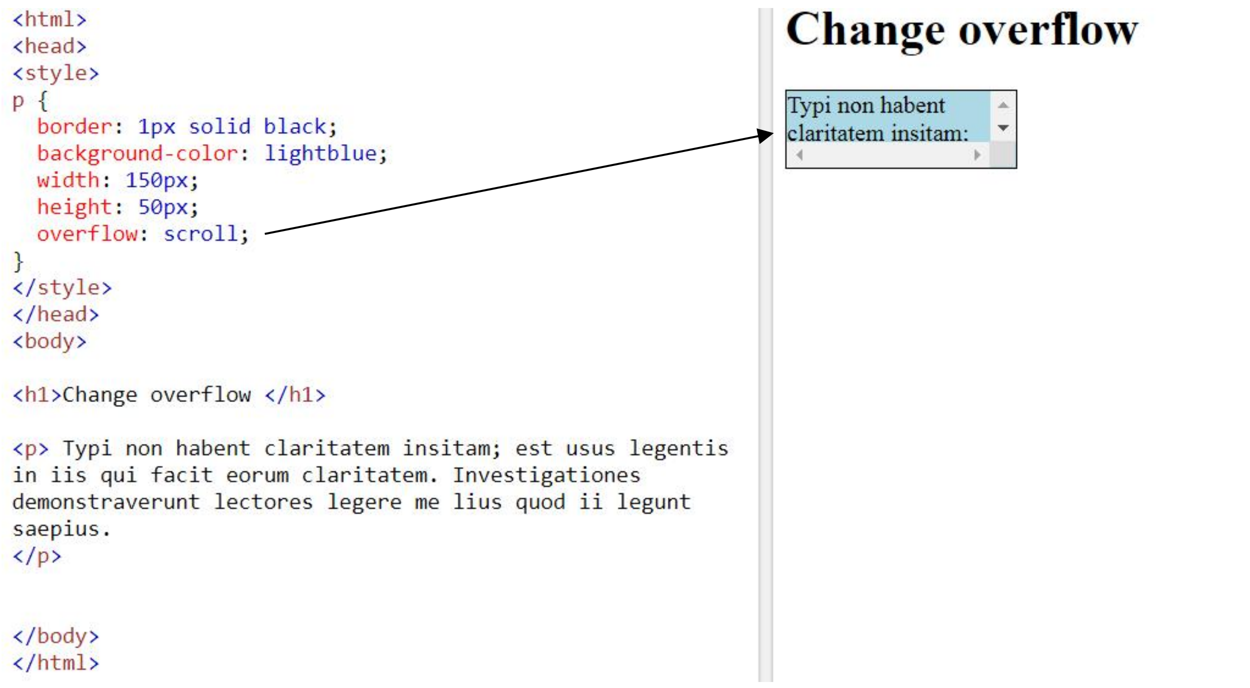
Typi non habent
claritatem insitam; est
usus legentis in iis qui



```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  border: 1px solid black;
  background-color: lightblue;
  width: 150px;
  height: 50px;
  overflow: visible;
}
</style>
</head>
<body>

<h1>Change overflow </h1>

<p> Typi non habent claritatem insitam; est usus legentis
in iis qui facit eorum claritatem. Investigationes
demonstraverunt lectores legere me lius quod ii legunt
saepius.
</p>


</body>
</html>
```

Change overflow

Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.



```
<html>
<head>
<style>
p {
  border: 1px solid black;
  background-color: lightblue;
  width: 150px;
  height: 50px;
  overflow: scroll;
}
</style>
</head>
<body>

<h1>Change overflow </h1>

<p> Typi non habent claritatem insitam; est usus legentis
in iis qui facit eorum claritatem. Investigationes
demonstraverunt lectores legere me lius quod ii legunt
saepius.
</p>


</body>
</html>
```

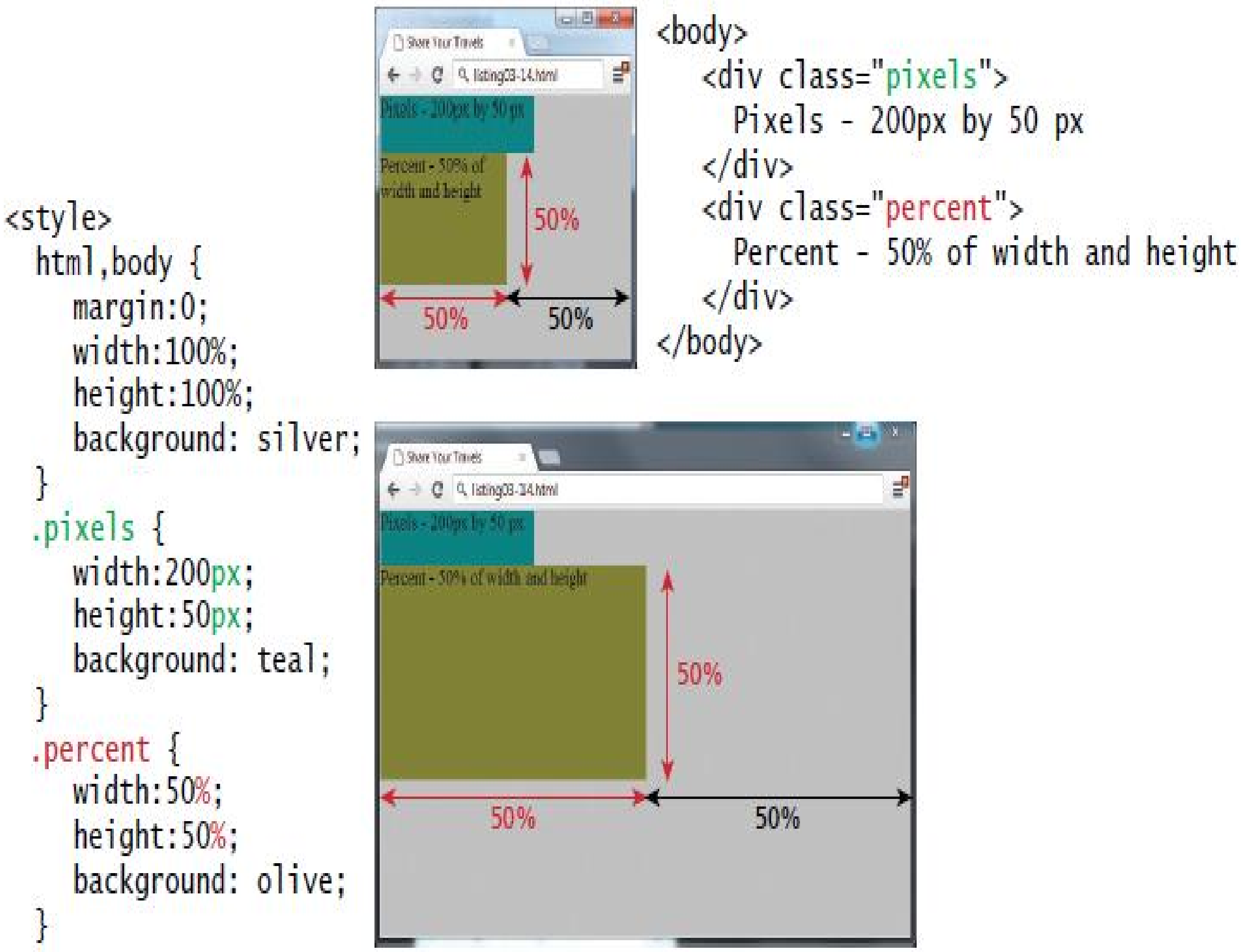
Change overflow

Typi non habent claritatem insitam;

Boxing with percentage:

One of the problems with using percentages as the unit for sizes is that as the browser window shrinks too small or expands too large (for instance on a widescreen monitor), elements might become too small or too large. You can put absolute pixel constraints on the minimum and maximum sizes via the min-width, min-height, max-width, and max-height properties.

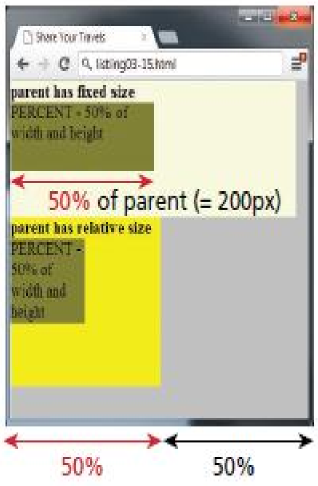
Developer tools in current browsers make it significantly easier to examine and troubleshoot CSS than was the case a decade ago.



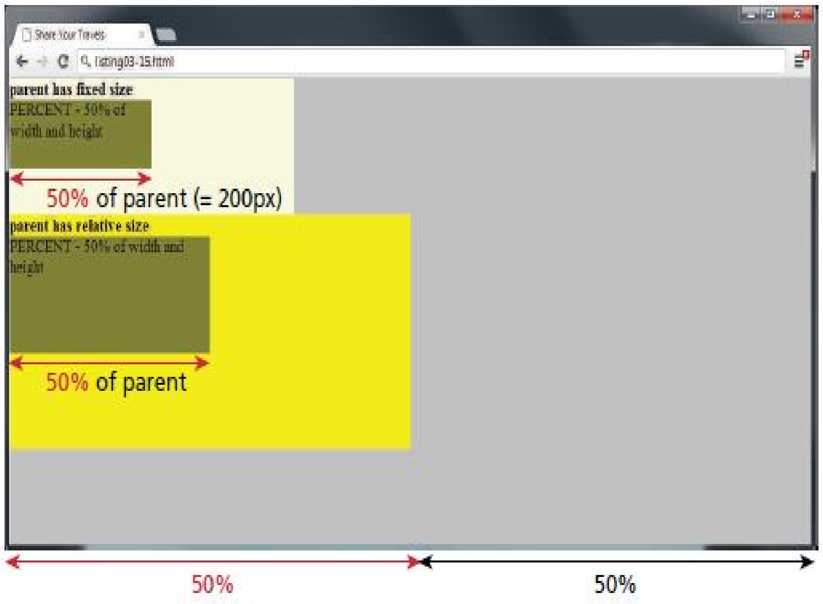
```
<style>
  html,body {
    margin:0;
    width:100%;
    height:100%;
    background: silver;
  }
  .pixels {
    width:200px;
    height:50px;
    background: teal;
  }
  .percent {
    width:50%;
    height:50%;
    background: olive;
  }
```

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

```
  .parentFixed {
     width:400px;
     height:150px;
     background: beige;
  }
  .parentRelative {
     width:50%;
     height:50%;
     background: yellow;
  }
</style>
```



```
<body>
<div class="parentFixed">
   <strong>parent has fixed size</strong>
   <div class="percent">
      PERCENT - 50% of width and height
   </div>
</div>
<div class="parentRelative">
   <strong>parent has relative size</strong>
   <div class="percent">
      PERCENT - 50% of width and height
   </div>
</div>
</body>
```



Box sizing via percents

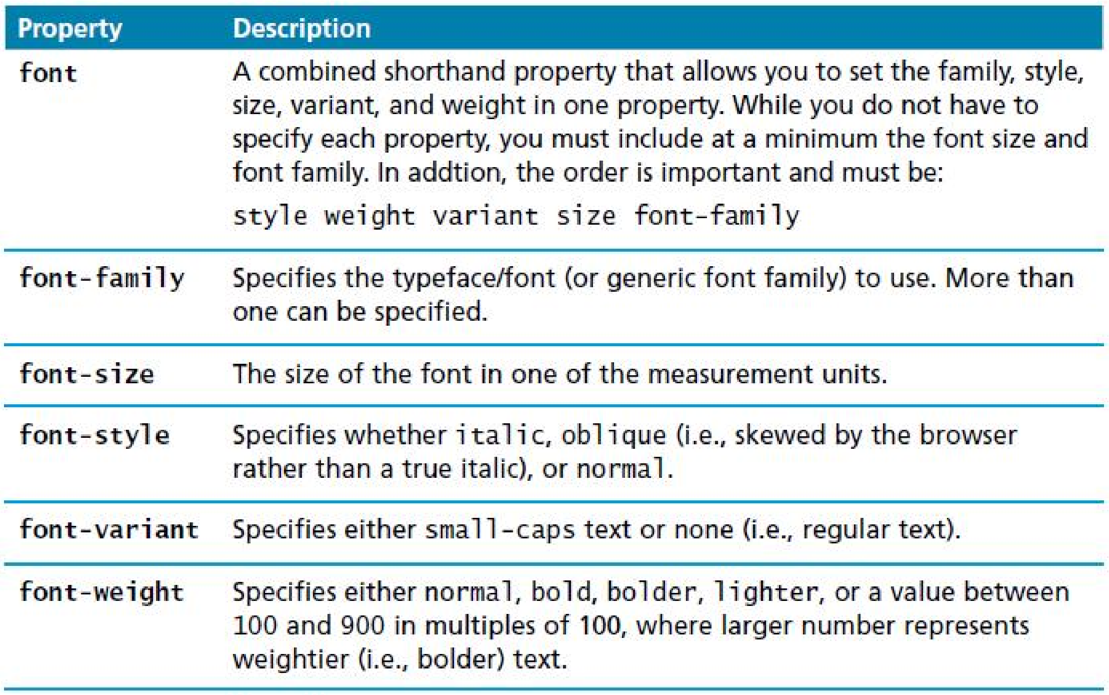
2.7 CSS Text Styling

CSS provides two types of properties that affect text.

- Font properties because they affect the font and its appearance.
- Paragraph properties since they affect the text in a similar way no matter which font is being used.

Font Family

Web font stack- A font stack is a list of fonts in the CSS font-family declaration. The fonts are listed in order of preference that would appear on the site in case of a problem like a font not loading.



Property	Description
`font`	A combined shorthand property that allows you to set the family, style, size, variant, and weight in one property. While you do not have to specify each property, you must include at a minimum the font size and font family. In addtion, the order is important and must be: `style weight variant size font-family`
`font-family`	Specifies the typeface/font (or generic font family) to use. More than one can be specified.
`font-size`	The size of the font in one of the measurement units.
`font-style`	Specifies whether `italic`, `oblique` (i.e., skewed by the browser rather than a true italic), or `normal`.
`font-variant`	Specifies either `small-caps` text or none (i.e., regular text).
`font-weight`	Specifies either `normal`, `bold`, `bolder`, `lighter`, or a value between 100 and 900 in multiples of 100, where larger number represents weightier (i.e., bolder) text.

Font Properties

In CSS, there are two types of font family names:

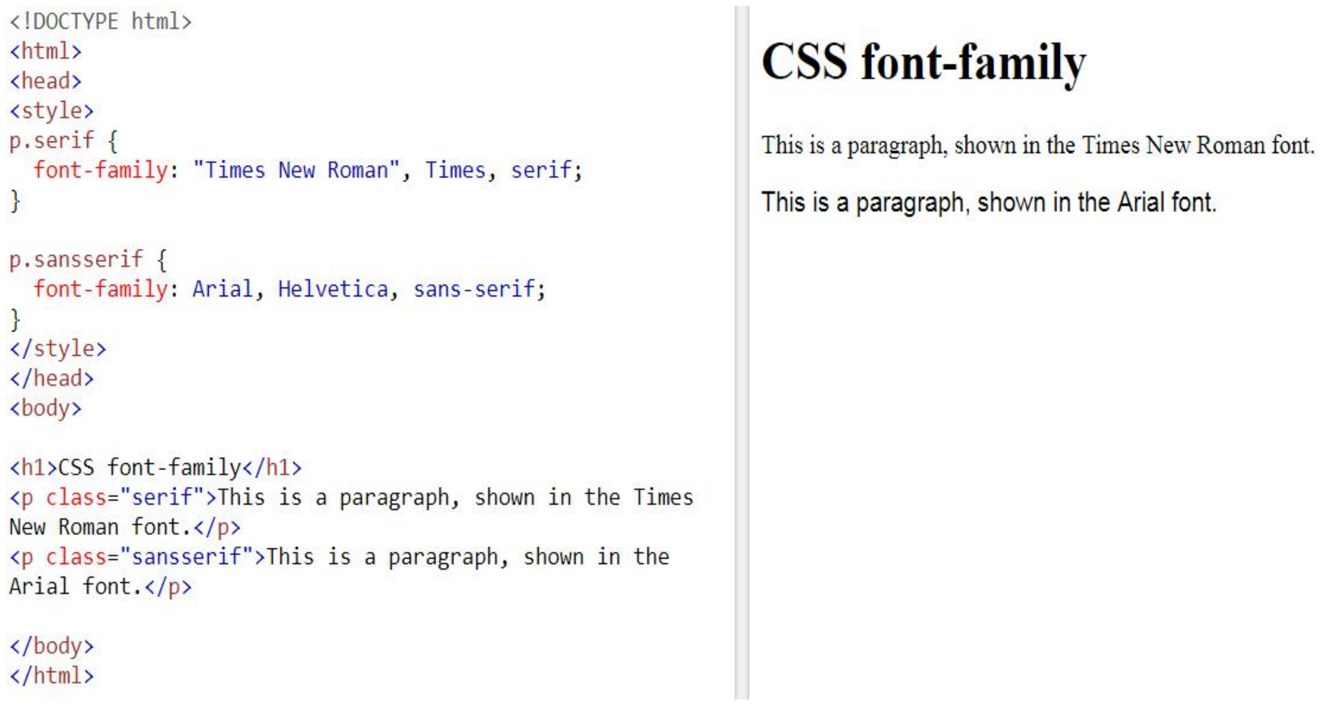
- generic family - a group of font families with a similar look (like "Serif" or "Monospace")
- font family - a specific font family (like "Times New Roman" or "Arial")

Times New Roman

Generic family	Font family	Description
Serif	Times New Roman	Serif fonts have small lines at the ends on some characters
	Georgia	
Sans-serif	Arial	"Sans" means without - these fonts do not have the lines at the ends of characters
	Verdana	
Monospace	Courier New	All monospace characters have the same width
	Lucida	
	Console	

<u>Font Family</u>: The font family of a text is set with the font-family property. The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on. Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

More than one font family is specified in a comma-separated list:



```
<!DOCTYPE html>
<html>
<head>
<style>
p.serif {
  font-family: "Times New Roman", Times, serif;
}

p.sansserif {
  font-family: Arial, Helvetica, sans-serif;
}
</style>
</head>
<body>

<h1>CSS font-family</h1>
<p class="serif">This is a paragraph, shown in the Times
New Roman font.</p>
<p class="sansserif">This is a paragraph, shown in the
Arial font.</p>

</body>
</html>
```

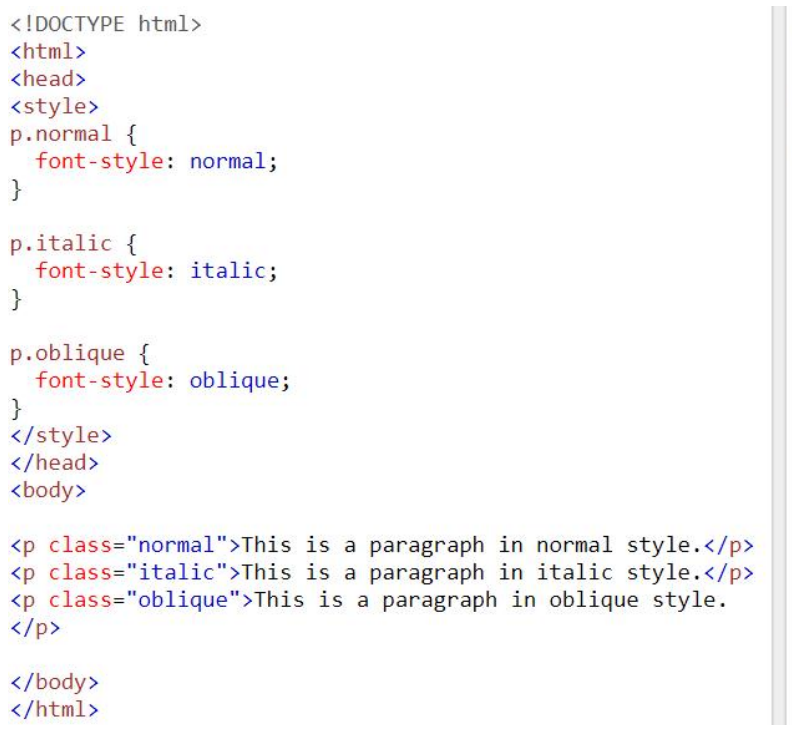
CSS font-family

This is a paragraph, shown in the Times New Roman font.

This is a paragraph, shown in the Arial font.

<u>Font Style:</u> The font-style property is mostly used to specify italic text.
This property has three values:

normal - The text is shown normally
italic - The text is shown in italics
oblique - The text is "leaning" (oblique is very similar to italic, but less supported)



```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  font-style: normal;
}

p.italic {
  font-style: italic;
}

p.oblique {
  font-style: oblique;
}
</style>
</head>
<body>

<p class="normal">This is a paragraph in normal style.</p>
<p class="italic">This is a paragraph in italic style.</p>
<p class="oblique">This is a paragraph in oblique style.
</p>

</body>
</html>
```

This is a paragraph in normal style.

This is a paragraph in italic style.

This is a paragraph in oblique style.

Paragraph Properties:

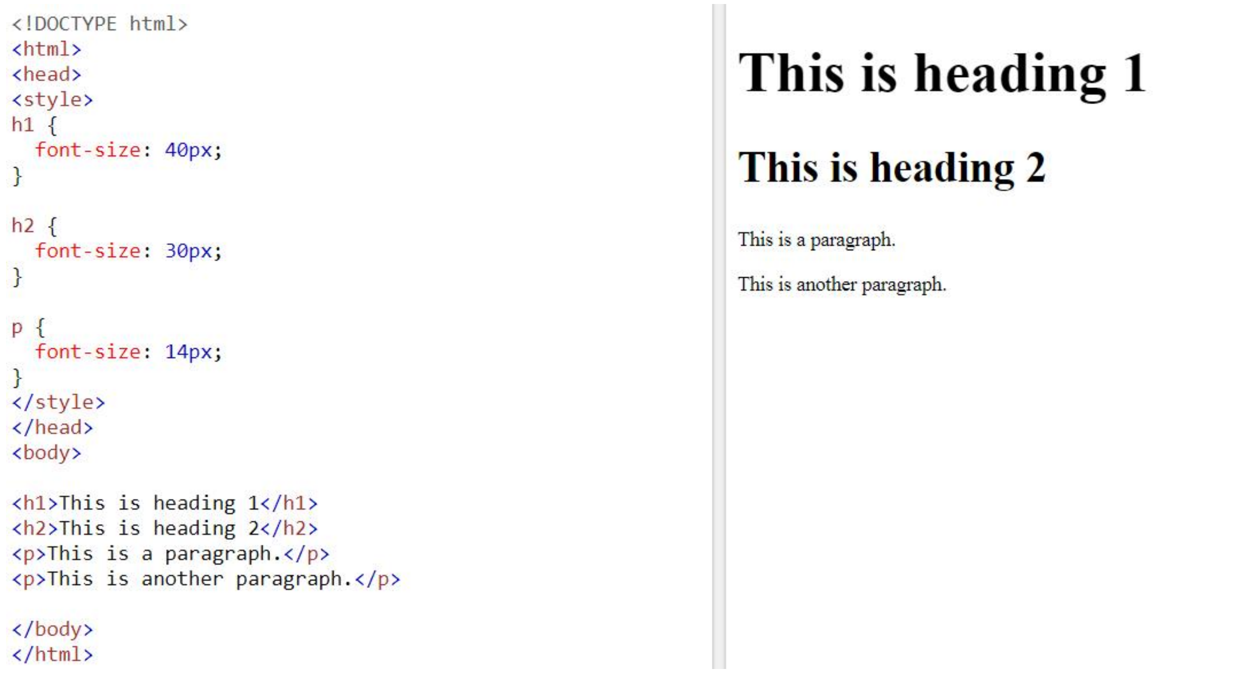
Font Size: The font-size property sets the size of the text. Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs. Sizing with pixels provides precise control to create web layouts that work well on different devices. Relative units such as **em units** or **percentages** for font sizes must be given.

Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs. The font-size value can be an absolute or relative size.

- **Absolute size:** Sets the text to a specified size Does not allow a user to change the text size in all browsers (bad for accessibility reasons). Absolute size is useful when the physical size of the output is known
- **Relative size:** Sets the size relative to surrounding elements. Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

Set Font Size With Pixels: Setting the text size with pixels gives you full control over the text size:



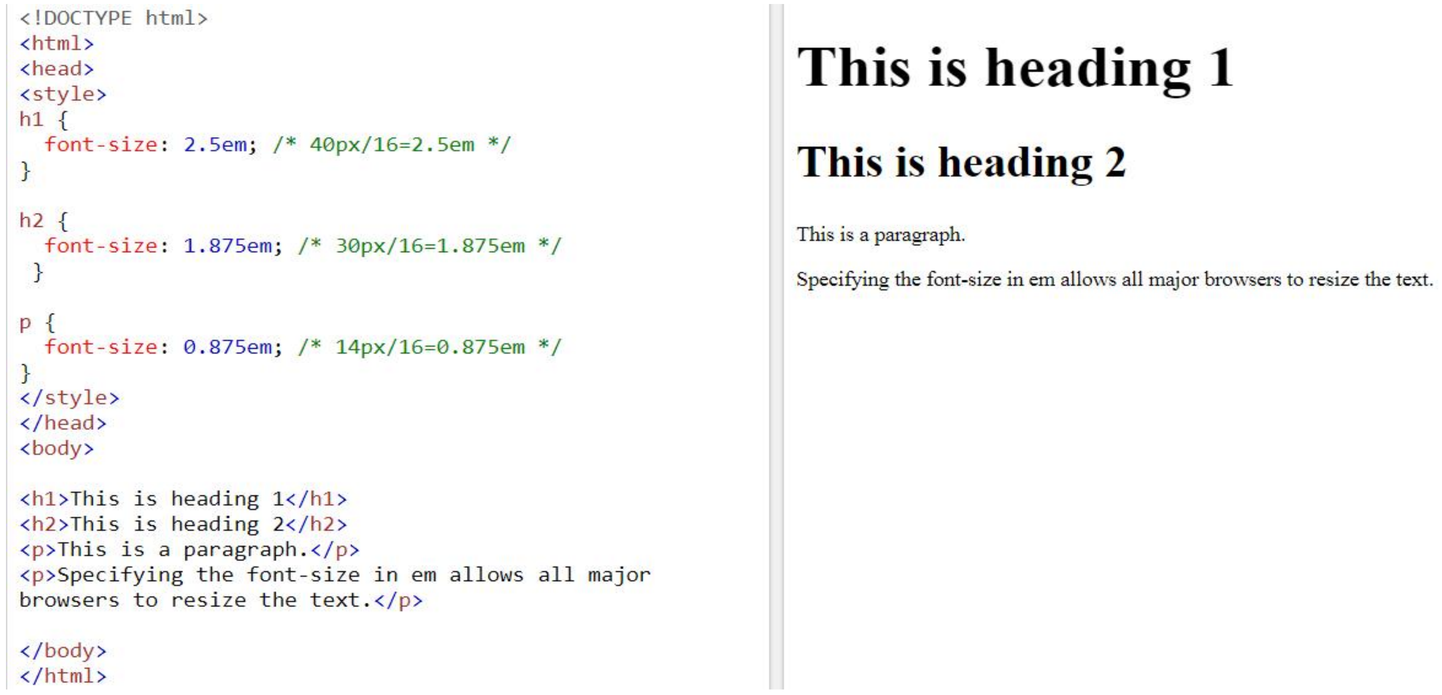
```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  font-size: 40px;
}

h2 {
  font-size: 30px;
}

p {
  font-size: 14px;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

```
This is heading 1

This is heading 2

This is a paragraph.

This is another paragraph.
```

Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels. The em size unit is recommended by the W3C. 1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: pixels/16=em



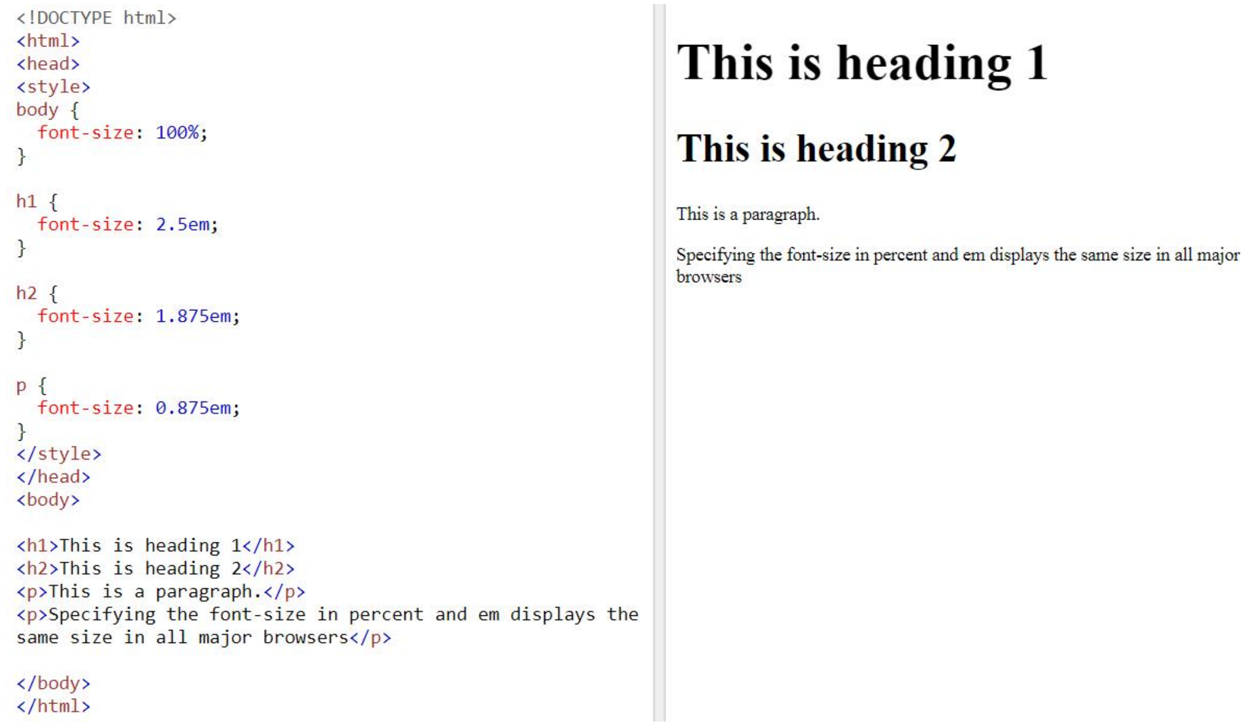
```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  font-size: 2.5em; /* 40px/16=2.5em */
}

h2 {
  font-size: 1.875em; /* 30px/16=1.875em */
 }

p {
  font-size: 0.875em; /* 14px/16=0.875em */
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>Specifying the font-size in em allows all major
browsers to resize the text.</p>

</body>
</html>
```

```
This is heading 1

This is heading 2

This is a paragraph.

Specifying the font-size in em allows all major browsers to resize the text.
```

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers. Unfortunately, there is still a problem with older versions of IE. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

Use a Combination of Percent and Em:

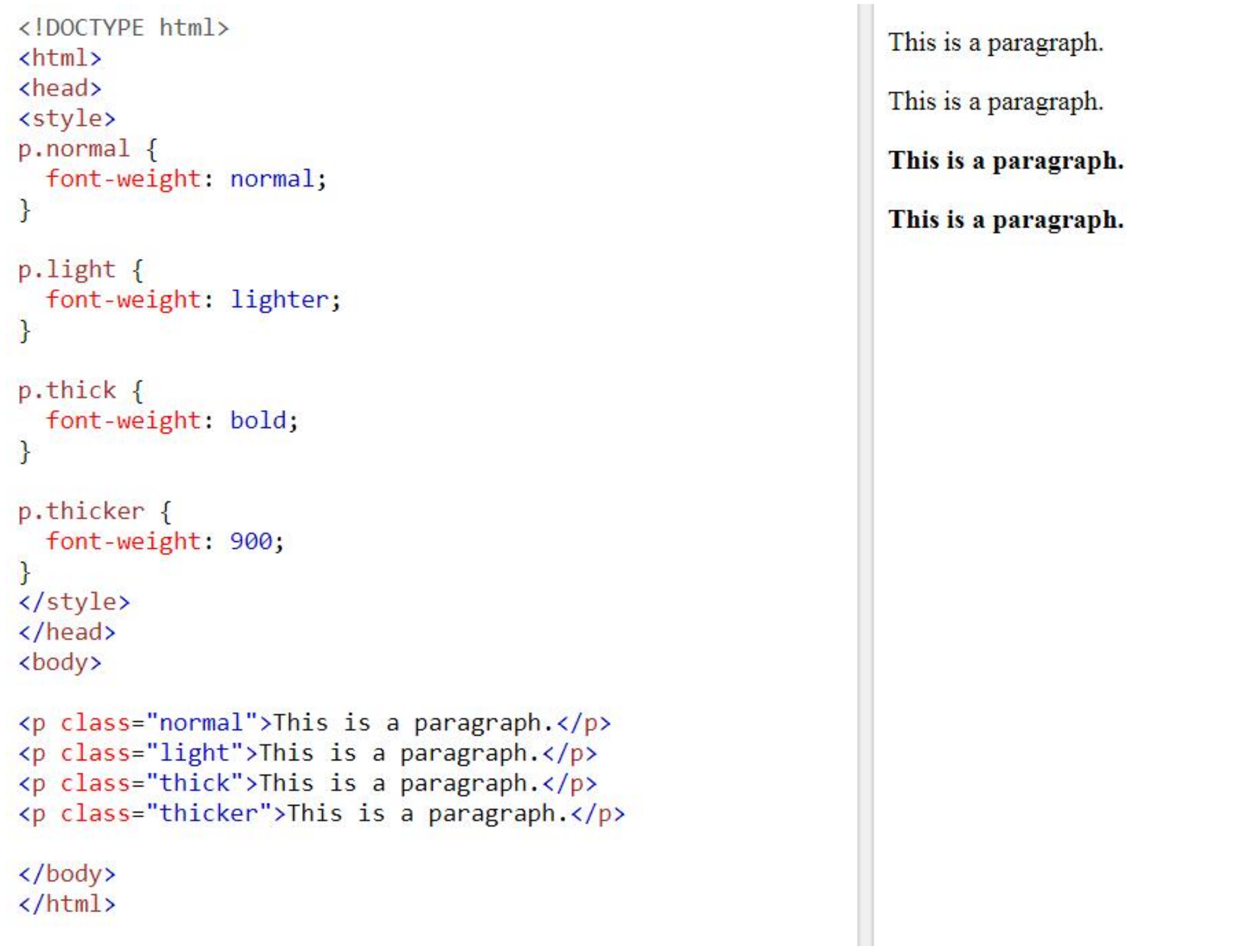
The solution that works in all browsers is to set a default font-size in percent for the <body> element:



```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-size: 100%;
}

h1 {
  font-size: 2.5em;
}

h2 {
  font-size: 1.875em;
}

p {
  font-size: 0.875em;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>Specifying the font-size in percent and em displays the
same size in all major browsers</p>

</body>
</html>
```

```
This is heading 1

This is heading 2

This is a paragraph.

Specifying the font-size in percent and em displays the same size in all major
browsers
```

Font Weight: The font-weight property specifies the weight of a font:



```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  font-weight: normal;
}

p.light {
  font-weight: lighter;
}

p.thick {
  font-weight: bold;
}

p.thicker {
  font-weight: 900;
}
</style>
</head>
<body>

<p class="normal">This is a paragraph.</p>
<p class="light">This is a paragraph.</p>
<p class="thick">This is a paragraph.</p>
<p class="thicker">This is a paragraph.</p>

</body>
</html>
```

This is a paragraph.

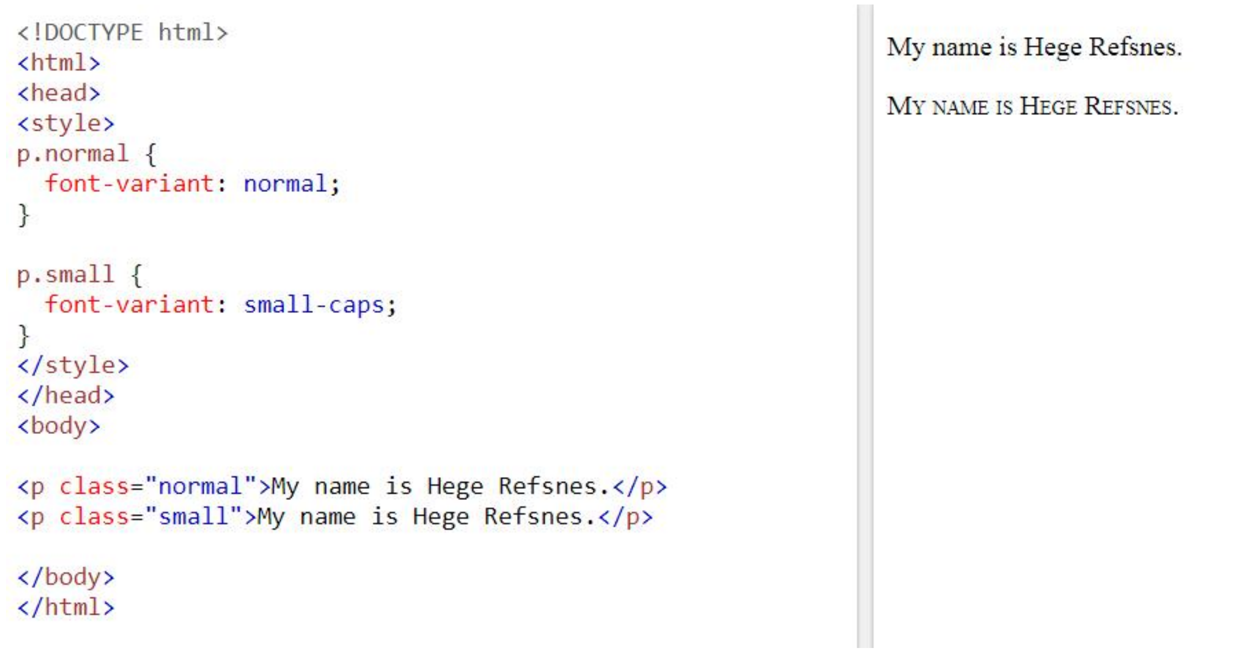
This is a paragraph.

This is a paragraph.

This is a paragraph.

Font Variant

The font-variant property specifies whether or not a text should be displayed in a small-caps font. In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.



```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  font-variant: normal;
}

p.small {
  font-variant: small-caps;
}
</style>
</head>
<body>

<p class="normal">My name is Hege Refsnes.</p>
<p class="small">My name is Hege Refsnes.</p>

</body>
</html>
```

```
My name is Hege Refsnes.

MY NAME IS HEGE REFSNES.
```

Property	Description
`letter-spacing`	Adjusts the space between letters. Can be the value `normal` or a length unit.
`line-height`	Specifies the space between baselines (equivalent to leading in a desktop publishing program). The default value is `normal`, but can be set to any length unit. Can also be set via the shorthand `font` property.
`list-style-image`	Specifies the URL of an image to use as the marker for unordered lists.
`list-style-type`	Selects the marker type to use for ordered and unordered lists. Often set to none to remove markers when the list is a navigational menu or a input form.
`text-align`	Aligns the text horizontally in a container element in a similar way as a word processor. Possible values are `left`, `right`, `center`, and justify.
`text-decoration`	Specifies whether the text will have lines below, through, or over it. Possible values are: none, `underline`, `overline`, `line-through`, and `blink`. Hyperlinks by default have this property set to `underline`.
`text-direction`	Specifies the direction of the text, left-to-right (`ltr`) or right-to-left (`rtl`).
`text-indent`	Indents the first line of a paragraph by a specific amount.
`text-shadow`	A new CSS3 property that can be used to add a drop shadow to a text. Not yet supported in IE9.
`text-transform`	Changes the capitalization of text. Possible values are none, `capitalize`, `lowercase`, and `uppercase`.
`vertical-align`	Aligns the text vertically in a container element. Most common values are: `top`, `bottom`, and `middle`.
`word-spacing`	Adjusts the space between words. Can be the value `normal` or a length unit.

Text Properties

Question Bank

1. What is the difference between XHTML and HTML5?
2. Why was the XHTML 2.0 standard eventually abandoned?
3. What role do HTML validators play in web development?
4. What are the main syntax rules for XML?
5. What are HTML elements? What are HTML attributes?
6. What is semantic markup? Why is it important?
7. Why is removing presentation-oriented markup from one's HTML documents considered to be a best practice?
8. What is the difference between standards mode and quirks mode? What role does the doctype play with these modes?
9. What is the difference between the <p> and the <div> element? In what contexts should one use the one over the other?
10. Describe the difference between a relative and an absolute reference. When should each be used?
11. What are the advantages of using the new HTML5 semantic elements? Disadvantages?
12. Are you allowed to use more than one <heading> element in a web page? Why or why not?
13. What are the main benefits of using CSS?
14. Compare the approach the W3C has used with CSS3 in comparison to CSS2.1.
15. What are the different parts of a CSS style rule?
16. What is the difference between a relative and an absolute measure unit in CSS? Why are relative units preferred over absolute units in CSS?
17. What are an element selector and a grouped element selector? Provide an example of each.
18. What are class selectors? What are id selectors? Briefly discuss why you would use one over the other.
19. What are contextual selectors? Identify the four different contextual selectors.
20. What are pseudo-class selectors? What are they commonly used for?
21. What does cascade in CSS refer to?
22. What are the three cascade principles used by browsers when style rules conflict? Briefly describe each.
23. Illustrate the CSS box model. Be sure to label each of the components of the box.
24. What is a web font stack? Why are they necessary?