

# CSS 1: Introduction

**Abdallah Karakra & Sobhi Ahmed**

## Chapter 3

# Objectives

**1**

What is **CSS**?

**2**

CSS **Syntax**

**3**

**Location** of Styles

**4**

**Selectors**

**5**

The **Cascade**: How  
Styles Interact

**6**

The **Box** Model

**7**

CSS **Text** Styling

Section 1 of 7

# WHAT IS CSS?

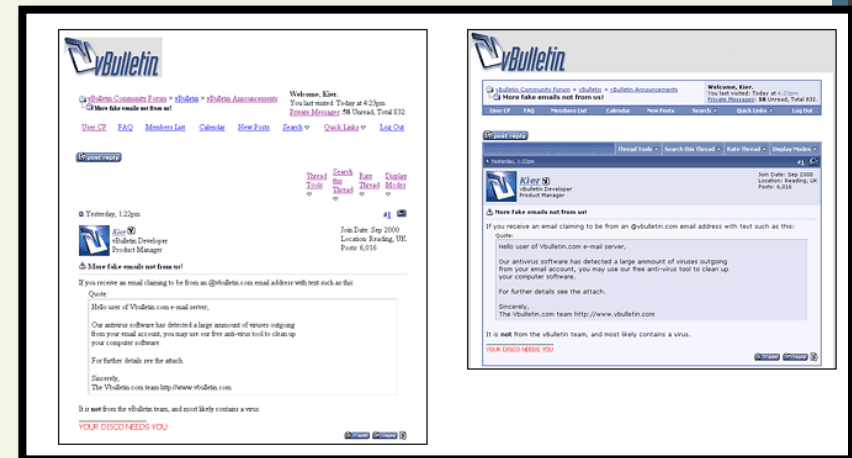
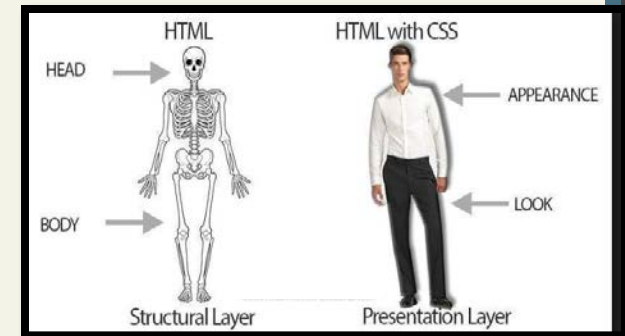
# What is CSS?

You be styling soon

CSS is a W3C standard for describing the **presentation (or appearance)** of HTML elements.

With CSS, we can assign

- font properties,
- colors,
- sizes,
- borders,
- background images,
- even the position of elements.



# What is CSS?

CSS is a language in that it has its own syntax rules.

CSS can be added :

- Directly to any HTML element (**via the style attribute**)

```
<h2 style="font-size: 24pt; font-weight: bold;"> Reviews</h2>
```

- within **the <head> element.**

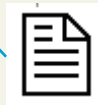
```
<head lang="en">
  <meta charset="utf-8">
  <title>Share Your Travels -- New York - Central Park</title>
  <style>
    h1 { font-size: 24pt; }
    h2 {
      font-size: 18pt;
      font-weight: bold;
    }
  </style>
</head>
```

# What is CSS?

Cont.

- Most commonly, in a separate **text file that contains only CSS**.

```
<head lang="en">
  <meta charset="utf-8">
  <title>Share Your Travels -- New York - Central Park</title>
  <link rel="stylesheet" href="styles.css" />
</head>
```



**Styles.css**

```
<link rel="stylesheet" href="styles.css" />
```

# Benefits of CSS

Why using CSS is a better way of describing presentation than HTML

- **Improved control over formatting.** 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
- **Improved accessibility.** By keeping presentation out of the HTML.
- **Improved output flexibility.** CSS can be used to adopt a page for different output media (**responsive design**).



Section 2 of 7

# CSS SYNTAX



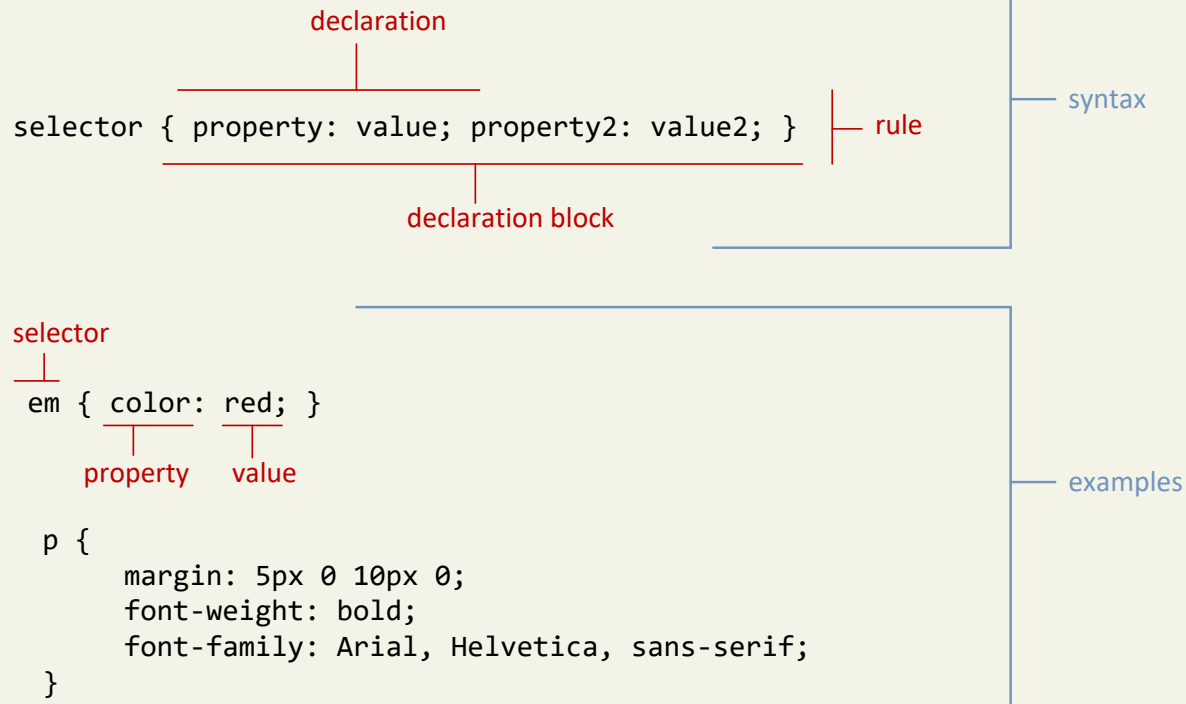
# CSS Syntax

Rules, properties, values, declarations

A CSS document consists of one or more [style rules](#).

A rule consists of

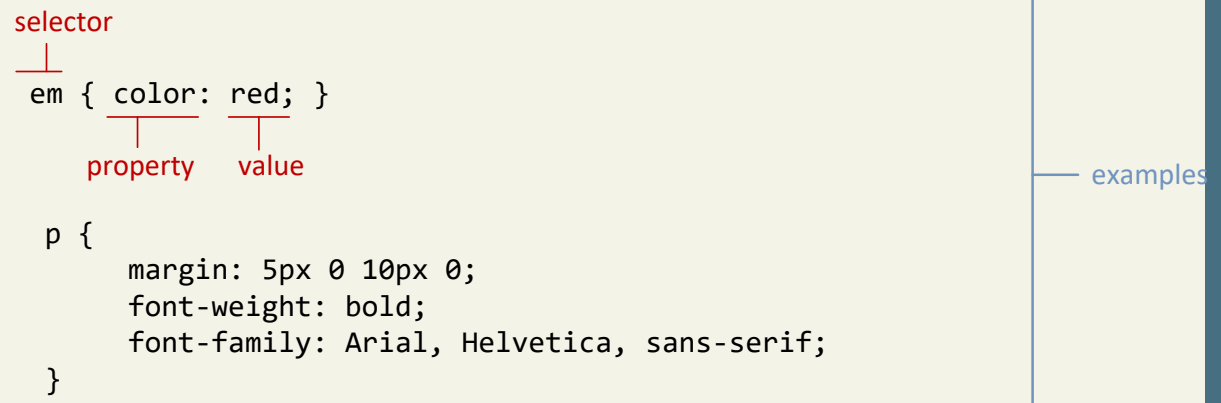
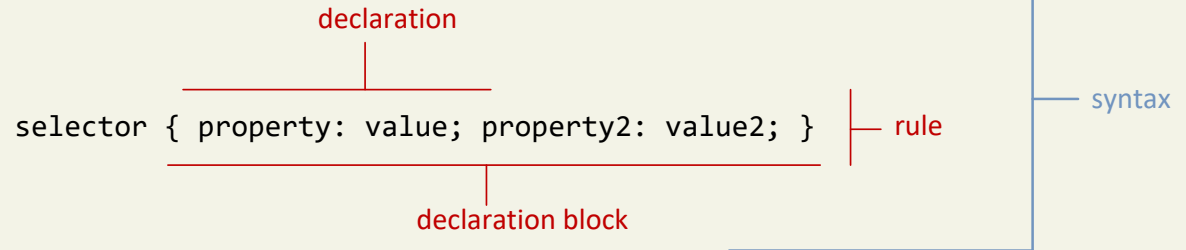
- a [selector](#) that identifies the HTML element or elements that will be affected.
- a series of [property](#) and [value](#) pairs (each pair is also called a [declaration](#)).



# Declaration Blocks

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
- Each declaration is terminated with a **semicolon**.



# Selectors

Which elements

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**.

```
h1,h2,h3,h4,h5,h6 {  
  color: green  
}
```

declaration

selector { property: value; property2: value2; } — rule

declaration block

syntax

selector

em { color: red; }

property value

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

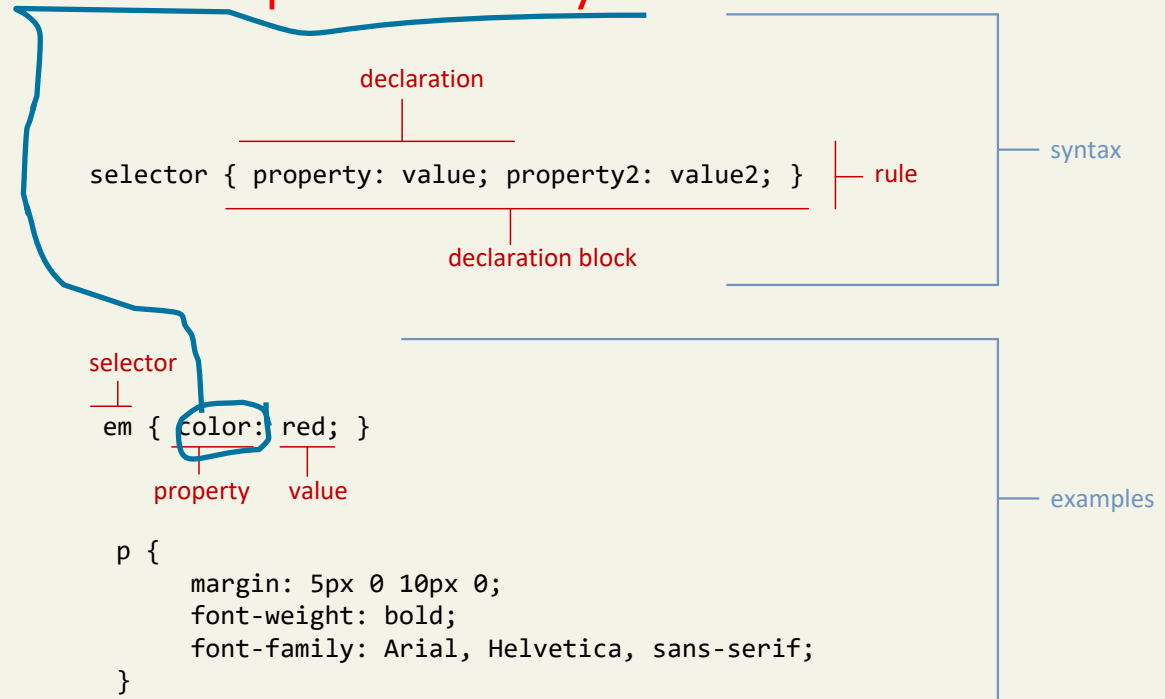
examples

# Properties

Which style properties of the selected elements

Each individual CSS declaration must contain a **property**.

These property names are predefined by the CSS standard.



# Properties

## Common CSS properties

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

These property names are predefined by the CSS standard.

Using the `style` attribute, can specify presentation style for a single HTML element

within tag, list sequence of property:value pairs

font-family:Courier,monospace

font-style:italic

font-weight:bold

font-size:12pt font-size:large font-size:larger

color:red color:#000080

background-color:white

text-decoration:underline

text-decoration:none

# Properties

Common CSS properties continued.

Property Type	Property
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

# Color Values

CSS supports a variety of different ways of describing color

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 for 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	Allows you to add an alpha, or transparency, value. This allows a background color or image to “show through” the color. Transparency 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

```
p { color: red; }  
h2 { color: rgb(128, 0, 196); }  
h4 { color: #FF8800; }
```

This paragraph uses the first style above.

This h2 uses the second style above.

This h4 uses the third style above.

## RGB Calculator



rgb(200, 10, 150)

#c80a96

hsl(316, 90%, 41%)



[https://www.w3schools.com/colors/colors\\_rgb.asp](https://www.w3schools.com/colors/colors_rgb.asp)

Each parameter (red, green, and blue) defines the intensity of the color as an integer between 0 and 255.



# Color Values

[https://www.w3schools.com/colors/colors\\_hsl.asp](https://www.w3schools.com/colors/colors_hsl.asp)

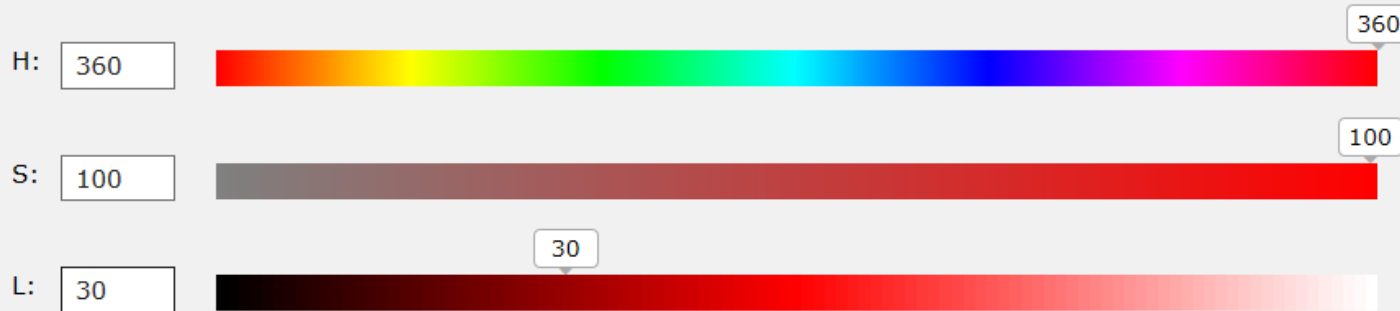
## HSL Calculator



hsl(0, 100%, 30%)

rgb(153, 0, 0)

#990000



## HSL Colors

HSL color values are supported in IE9+, Firefox, Chrome, Safari, and in Opera 10+.

HSL stands for hue, saturation, and lightness.

HSL color values are specified with: `hsl(hue, saturation, lightness)`.

# Color Values

CSS supports a variety of different ways of describing color

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

<h1 style="background-color:hsl(0, 100%, 50%);">Hello CS Students</h1>
<h1 style="background-color:hsl(300, 76%, 72%);">Hello CS Students</h1>
<h1 style="background-color:hsl(39, 100%, 50%);">Hello CS Students</h1>
<h1 style="background-color:Orange;">Hello CS Students</h1>
<h3 style="color:Tomato;">Hello World</h3>
<h1 style="border:5px solid Tomato;">Hello World</h1>

<p>In HTML, you can specify colors using HSL values.</p>

</body>
</html>
```

Hello CS Students

Hello CS Students

Hello CS Students

Hello CS Students

Hello World

Hello World

In HTML, you can specify colors using HSL values.

# Color Values

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

<p>Same as color name "Tomato":</p>
<h1 style="background-color:Tomato;">Tomato</h1>
<h1 style="background-color:rgb(255, 99, 71);">rgb(255, 99, 71)</h1>
<h1 style="background-color:#ff6347;">#ff6347</h1>
<h1 style="background-color:hsl(9, 100%, 64%;">hsl(9, 100%, 64%)</h1>

<p>Same as color name "Tomato", but 50% transparent:</p>
<h1 style="background-color:rgba(255, 99, 71, 0.5);">rgba(255, 99, 71, 0.5)</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">hsla(9, 100%, 64%,
0.5)</h1>

<h1 style="background-color:#FF69B4;">#ff6347</h1>

</body>
</html>
```

Same as color name "Tomato":

**Tomato**

**rgb(255, 99, 71)**

**#ff6347**

**hsl(9, 100%, 64%)**

Same as color name "Tomato", but 50% transparent:

**rgba(255, 99, 71, 0.5)**

**hsla(9, 100%, 64%, 0.5)**

**#ff6347**

# Relative Units

Unit	Description	Type
px	Pixel. In CSS2 this is a relative measure, while in CSS3 it is <b>absolute</b> (1/96 of an inch).	Relative (CSS2) Absolute (CSS3)
em	Equal to the <b>computed value of the font-size</b> property of the element on which it is used. When used for font sizes, the <b>em unit is in relation to the font size of the parent</b> .	Relative
%	A measure that is always relative to another value. The precise meaning of % varies depending upon which property 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 <b>font size of the root</b> element. Unlike <b>em</b> , which may be different for each element, the <b>rem</b> is constant throughout the document.	Relative (CSS3 only)
vw, vh	Stands for <b>viewport width and viewport height</b> . 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.  vw: 1% of the width of the viewport vh: 1% of the height of the viewport	Relative (CSS3 only)

# Absolute Units

Unit	Description	Type
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

# Relative Units

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  font-size: 16px;
  line-height: 2em;
}

div {
  font-size: 30px;
  border: 1px solid black;
}

span {
  font-size: 0.5em;
}
</style>
</head>
<body>
<p>These paragraphs have a calculated line-height of: 2x16px = 32px.</p>
<p>These paragraphs have a calculated line-height of: 2x16px = 32px.</p>
<p>These paragraphs have a calculated line-height of: 2x16px = 32px.</p>
<div>The font-size of the div element is set to 30px. <span>The span element
inside the div element has a font-size of 0.5em, which equals to 0.5x30 =
15px</span>.</div>
</body>
</html>
```

These paragraphs have a calculated line-height of: 2x16px = 32px.

These paragraphs have a calculated line-height of: 2x16px = 32px.

These paragraphs have a calculated line-height of: 2x16px = 32px.

The font-size of the div element is set to 30px. The span element inside the div element has a font-size of 0.5em, which equals to 0.5x30 = 15px.

## Example

# Comments in CSS

It is often helpful to add comments to your style sheets. Comments take the form:

*`/* comment goes here */`*

Section 3 of 7

# LOCATION OF STYLES



# Style Locations

Author Created CSS style rules can be located in three different locations

CSS style rules can be located in three different locations.

- Inline
- Embedded
- External

# Inline Styles

Style rules placed within an HTML element via the style attribute

```
<h2 style="font-size: 24pt; font-weight: bold;"> Reviews</h2>
```

```
<h2> Reviews</h2>
```

An inline style only affects the element it is defined within and will override any other style definitions for the properties used in the inline style.

Using inline styles is generally discouraged since they increase bandwidth and **decrease maintainability**.

Inline styles can however be handy for **quickly testing** out a style change.

# Inline Styles

Style rules placed within an HTML element via the style attribute

```
<!DOCTYPE HTML>
<HTML>
<HEAD>

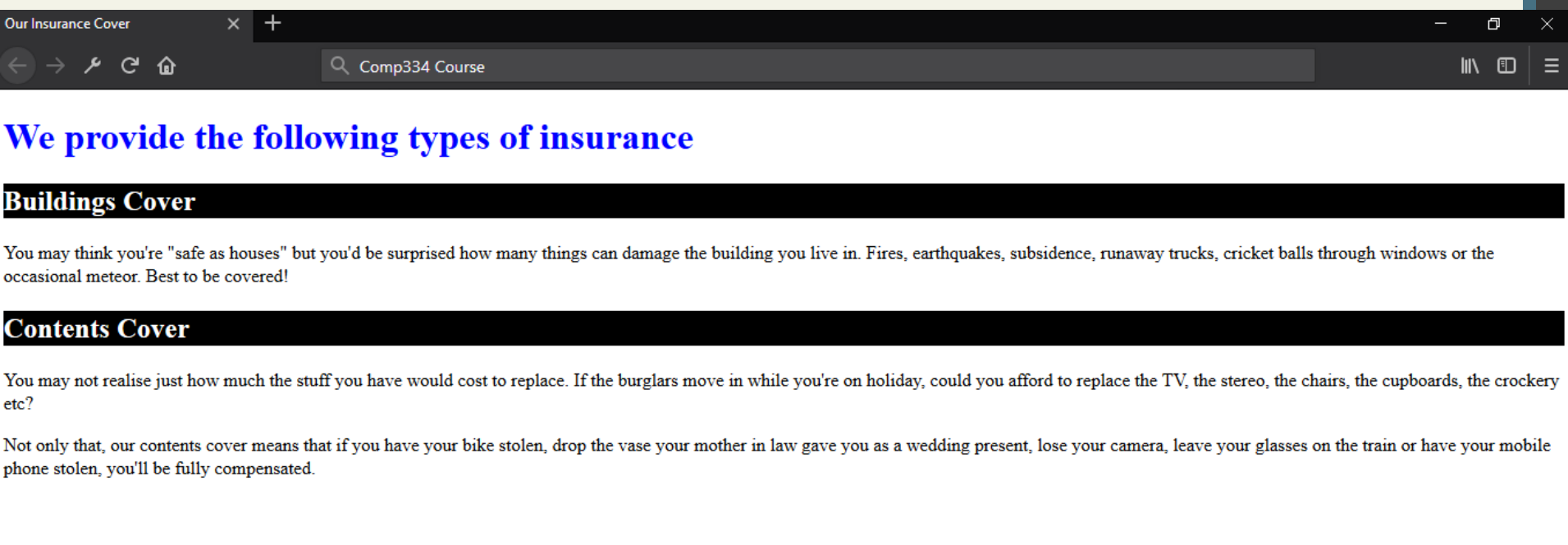
  <TITLE>Our Insurance Cover</TITLE>
</HEAD>
<BODY>
<!-- File: example4-1.htm -->
<H1 style="color: blue">We provide the following types of insurance</H1>
<H2 style="color: white; background-color: black">Buildings Cover</H2>
<P>You may think you're "safe as houses" but you'd be surprised how many things
can damage the building you live in. Fires, earthquakes, subsidence, runaway
trucks, cricket balls through windows or the occasional meteor. Best to be
covered!
</P>
<H2 style="color: rgb(255,255,255); background-color: rgb(0,0,0)">Contents
Cover</H2>
<P>You may not realise just how much the stuff you have would cost to replace. If
the burglars move in while you're on holiday, could you afford to replace the TV,
the stereo, the chairs, the cupboards, the crockery etc?
</P>
<P>Not only that, our contents cover means that if you have your bike stolen, drop
the vase your mother in law gave you as a wedding present, lose your camera, leave
your glasses on the train or have your mobile phone stolen, you'll be fully
compensated.
</P>
</BODY>
</HTML>
```

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

# Inline Styles

Style rules placed within an HTML element via the style attribute



Example

# Embedded Style Sheet

Style rules placed within the `<style>` element inside the `<head>` element

```
<head lang="en">
  <meta charset="utf-8">
  <title>Share Your Travels -- New York - Central Park</title>
  <style>
    h1 { font-size: 24pt; }
    h2 {
      font-size: 18pt;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <h1>Share Your Travels</h1>
  <h2>New York - Central Park</h2>
  ...

```

**LISTING 3.2** Embedded styles example

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

Since each HTML document has its own `<style>` element, it is more difficult to consistently style multiple documents when using embedded styles.

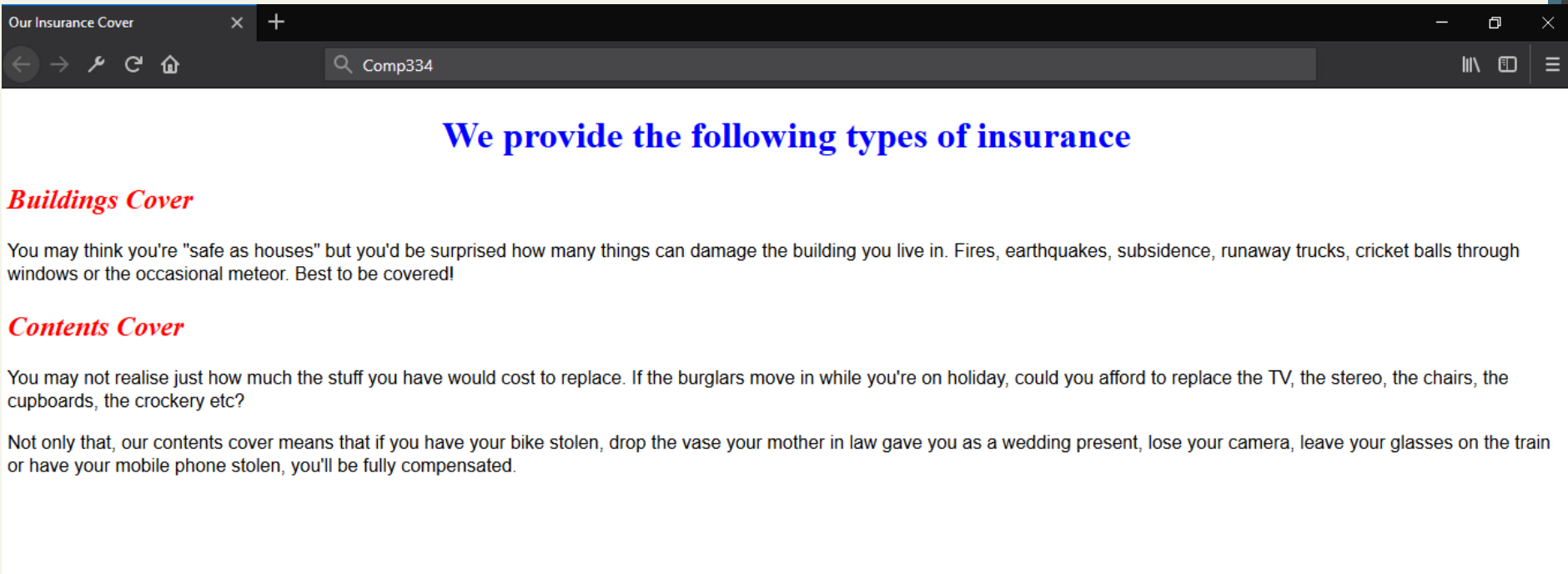
Example

# Embedded Style Sheet

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <STYLE>
      h1{color: blue; text-align: center}
      h2{color: red; font-style: italic}
      p{font-family: sans-serif}
    </STYLE>
    <TITLE>Our Insurance Cover</TITLE>
  </HEAD>
  <BODY>
    <!-- File: example4-2.htm -->
    <H1>We provide the following types of insurance</H1>
    <H2>Buildings Cover</H2>
    <P>You may think you're "safe as houses" but you'd be surprised how many things
    can damage the building you live in. Fires, earthquakes, subsidence, runaway
    trucks, cricket balls through windows or the occasional meteor. Best to be
    covered!
    </P>
    <H2>Contents Cover</H2>
    <P>You may not realise just how much the stuff you have would cost to replace. If
    the burglars move in while you're on holiday, could you afford to replace the TV,
    the stereo, the chairs, the cupboards, the crockery etc?
    </P>
    <P>Not only that, our contents cover means that if you have your bike stolen, drop
    the vase your mother in law gave you as a wedding present, lose your camera, leave
    your glasses on the train or have your mobile phone stolen, you'll be fully
    compensated.
    </P>
  </BODY>
</HTML>
```

# Embedded Style Sheet

Style rules placed within the `<style>` element inside the `<head>` element



# External Style Sheet

Style rules placed within a external text file with the .css extension

```
<head lang="en">
  <meta charset="utf-8">
  <title>Share Your Travels -- New York - Central Park</title>
  <link rel="stylesheet" href="styles.css" />
</head>
```

## LISTING 3.3 Referencing an external style sheet

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**



# External Style Sheet

```
<!DOCTYPE HTML>
```

```
<HTML>
```

```
<HEAD>
```

```
<LINK href="webhomecover.css" rel="stylesheet" />
```

```
<TITLE>Our Insurance Cover</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<!-- File: example4-3.htm -->
```

```
<H1>We provide the following types of insurance</H1>
```

```
<H2>Buildings Cover</H2>
```

```
<P>You may think you're "safe as houses" but you'd be surprised how many things can damage the building you live in. Fires, earthquakes, subsidence, runaway trucks, cricket balls through windows or the occasional meteor. Best to be covered!
```

```
</P>
```

```
<H2>Contents Cover</H2>
```

```
<P>You may not realise just how much the stuff you have would cost to replace. If the burglars move in while you're on holiday, could you afford to replace the TV, the stereo, the chairs, the cupboards, the crockery etc?
```

```
</P>
```

```
<P>Not only that, our contents cover means that if you have your bike stolen, drop the vase your mother in law gave you as a wedding present, lose your camera, leave your glasses on the train or have your mobile phone stolen, you'll be fully compensated.
```

```
</P>
```

```
</BODY>
```

```
</HTML>
```



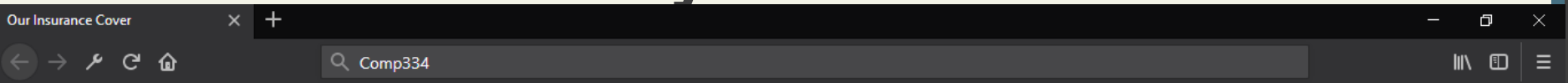
webhomecover.css

18/2/2013 11:12 PM

Cascading Style S...

1 KB

# External Style Sheet



## We provide the following types of insurance

### *Buildings Cover*

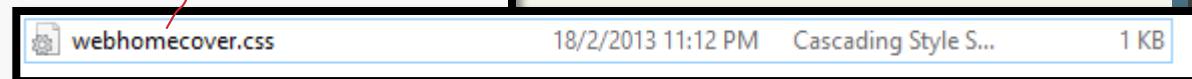
You may think you're "safe as houses" but you'd be surprised how many things can damage the building you live in. Fires, earthquakes, subsidence, runaway trucks, cricket balls through windows or the occasional meteor. Best to be covered!

### *Contents Cover*

You may not realise just how much the stuff you have would cost to replace. If the burglars move in while you're on holiday, could you afford to replace the TV, the stereo, the chairs, the cupboards, the crockery etc?

Not only that, our contents cover means that if you have your bike stolen, drop the vase your mother in law gave you as a wedding present, lose your camera, leave your glasses on the train or have your mobile phone stolen, you'll be fully compensated.

```
h1{color: blue}
h2{color: red; font-style: italic}
p{font-family: sans-serif}
.heading{text-align: center}
p.emphasis{font-weight: bold}
#footer
{
  font-family: sans-serif;
  font-weight:bold;
  font-style:italic;
  color:white;
  background-color:black;
  text-align:center
}
ul{list-style-type: square}
ol{list-style-type: lower-roman}
table{width: 50%; border: 3px black solid; margin: auto; border-collapse: collapse}
th{color: white; background-color: black}
td{border: 1px black solid}
```



- [Example 1](#)
- [Example 2](#)

Section 4 of 7

# SELECTORS

# Element Selectors

Selects all instances of a given HTML element

Uses the HTML element name.

You can select all elements by using the **universal element selector**, which is the \* (asterisk) character.

declaration

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

rule

declaration block

selector

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

property value

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

# Element Selectors

Selects all instances of a given HTML element

Uses the HTML element name.

You can select all elements by using the **universal element selector**, which is the **\*** (asterisk) character.

Example:

```
<head>
<style>
* {
    background-color: yellow;
    color:red;
    font-size:13pt;
}
</style>
</head>
```

# Element Selectors

Selects all instances of a given HTML element

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  background-color: yellow;
  color: red;
  font-size: 13pt;
}
</style>
</head>
<body>

<h1>Welcome to My Homepage</h1>

<div>
  <p>My name is Donald.</p>
  <p>I live in Duckburg.</p>
</div>

<p>My best friend is Mickey.</p>

</body>
</html>
```

Welcome to My Homepage

My name is Donald.

I live in Duckburg.

My best friend is Mickey.

# Grouped Selectors

Selecting multiple things

```
/* 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 3.4** Sample grouped selector

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

This is a sensible way to **reduce the size and complexity of your CSS files, by combining multiple identical rules into a single rule**.

# Reset

```
html, body, div, span, h1, h2, h3, h4, h5, h6, p {  
  margin: 0;  
  padding: 0;  
  border: 0;  
  font-size: 100%;  
  vertical-align: baseline;  
}
```

			
<b>Heading</b> <a href="#">This is link</a> this is span this is code  this is blockquote	<b>Heading</b> <a href="#">This is link</a> this is span this is code  this is blockquote	<b>Heading</b> <a href="#">This is link</a> this is span this is code  this is blockquote	<b>Heading</b> <a href="#">This is link</a> this is span this is code  this is blockquote
<b>Heading</b> <a href="#">This is link</a> this is span this is code  this is blockquote	<b>Heading</b> <a href="#">This is link</a> this is span this is code  this is blockquote	<b>Heading</b> <a href="#">This is link</a> this is span this is code  this is blockquote	<b>Heading</b> <a href="#">This is link</a> this is span this is code  this is blockquote

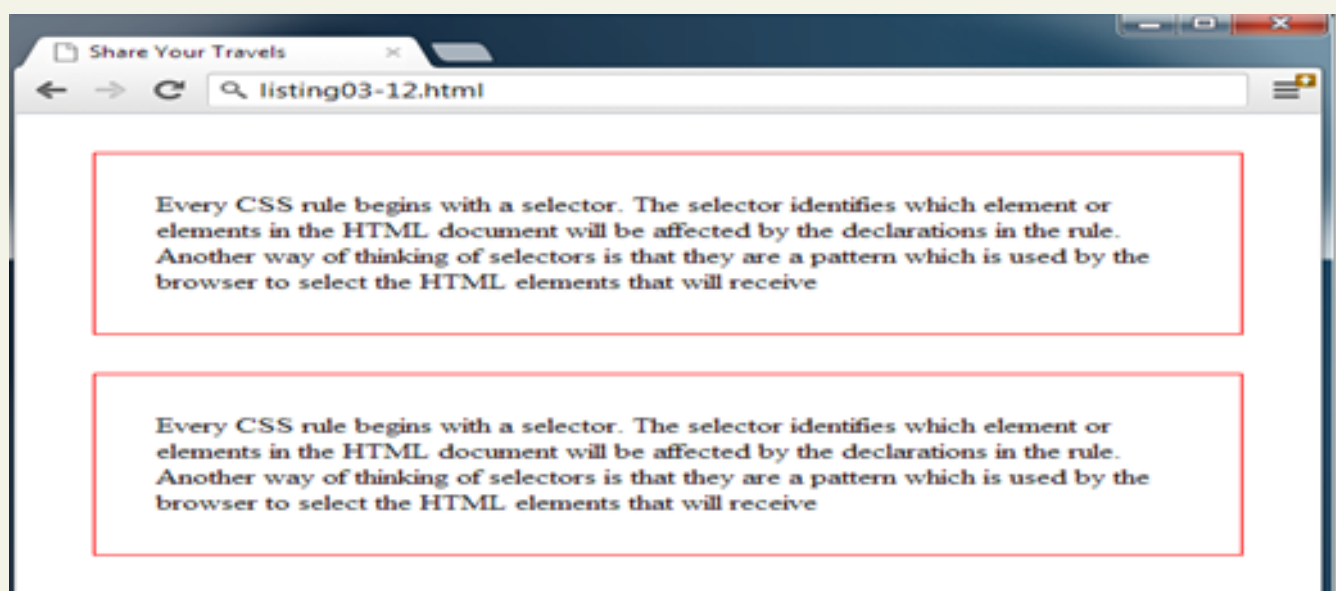
—reset→

Grouped selectors are often used as a way to quickly **reset** or remove browser defaults.

The goal of doing so is to reduce browser inconsistencies with things such as margins, line heights, and font sizes.



# Reset



reset styles can be placed in their own css file (**perhaps called reset.css**) and linked to the page **before** any other external styles sheets.

```
<head>
<link rel="stylesheet" type="text/css" href="reset.css">
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

1

2

# Class Selectors

Simultaneously target different HTML elements

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

If a series of HTML element have been labeled with ***the same class attribute value***, then you can target them for styling by using a class selector, which takes the form: **period (.) followed by the class name**.

`<h1>Reviews</h1>`

`<h1 class="first">Reviews</h1>`

`<p class="first">Hi</p>`

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

# Class Selectors

Simultaneously target different HTML elements

```
<h1>Reviews</h1>
```

```
<h1 class="first">Reviews</h1>
```

```
<p class="first">Hi</p></p>
```

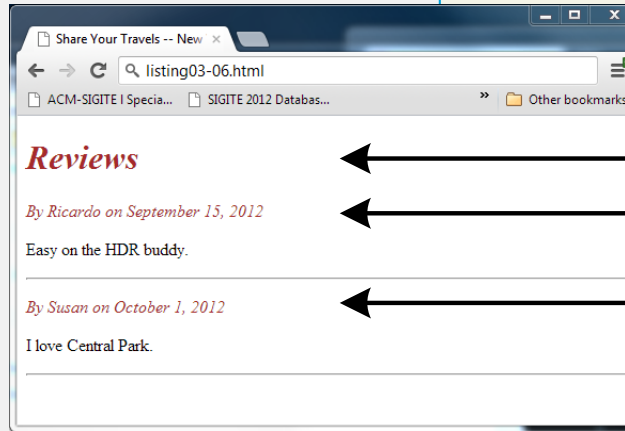
```
<p>Hi</p>
```

```
<div class="first"> ..... </div>
```

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

# Class Selectors

```
<head>
  <title>Share Your Travels </title>
  <style>
    .first {
      font-style: italic;
      color: brown;
    }
  </style>
</head>
<body>
  <h1 class="first">Reviews</h1>
  <div>
    <p class="first">By Ricardo on <time>September
15, 2012</time></p>
    <p>Easy on the HDR buddy.</p>
  </div>
  <hr/>
  <div>
    <p class="first">By Susan on <time>October 1,
2012</time></p>
    <p>I love Central Park.</p>
  </div>
  <hr/>
</body>
```



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

Example

# Id Selectors

Target a specific element by its id attribute

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

If an HTML element has been labeled with an id attribute, then you can target it for styling by using an id selector, which takes the form: pound/hash (#) **followed by the id name.**

Note: **You should only be using an id once per page**

# Id Selectors

```
<head lang="en">
  <meta charset="utf-8">
  <title>Share Your Travels -- New York - Central Park</title>
  <style>
    #latestComment {
      font-style: italic;
      color: brown;
    }
  </style>
</head>
<body>
  <h1>Reviews</h1>
  <div id="latestComment">
    <p>By Ricardo on <time>September 15, 2012</time></p>
    <p>Easy on the HDR buddy.</p>
  </div>
  <hr/>

  <div>
    <p>By Susan on <time>October 1, 2012</time></p>
    <p>I love Central Park.</p>
  </div>
  <hr/>
</body>
```



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

Example

# Id versus Class Selectors

How to decide

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 selectors should be used when (potentially) referencing **several related elements**.

# Attribute Selectors

Selecting via presence of element attribute or by the value of an attribute

An **attribute selector** provides a way to select HTML elements by either 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, not all web authors have used them**.

Attribute selectors can be a very helpful technique in the styling of hyperlinks and images.



# Attribute Selectors

```
<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>
    
    <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>
      
      
      
    </div>
  </div>
</body>
```

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

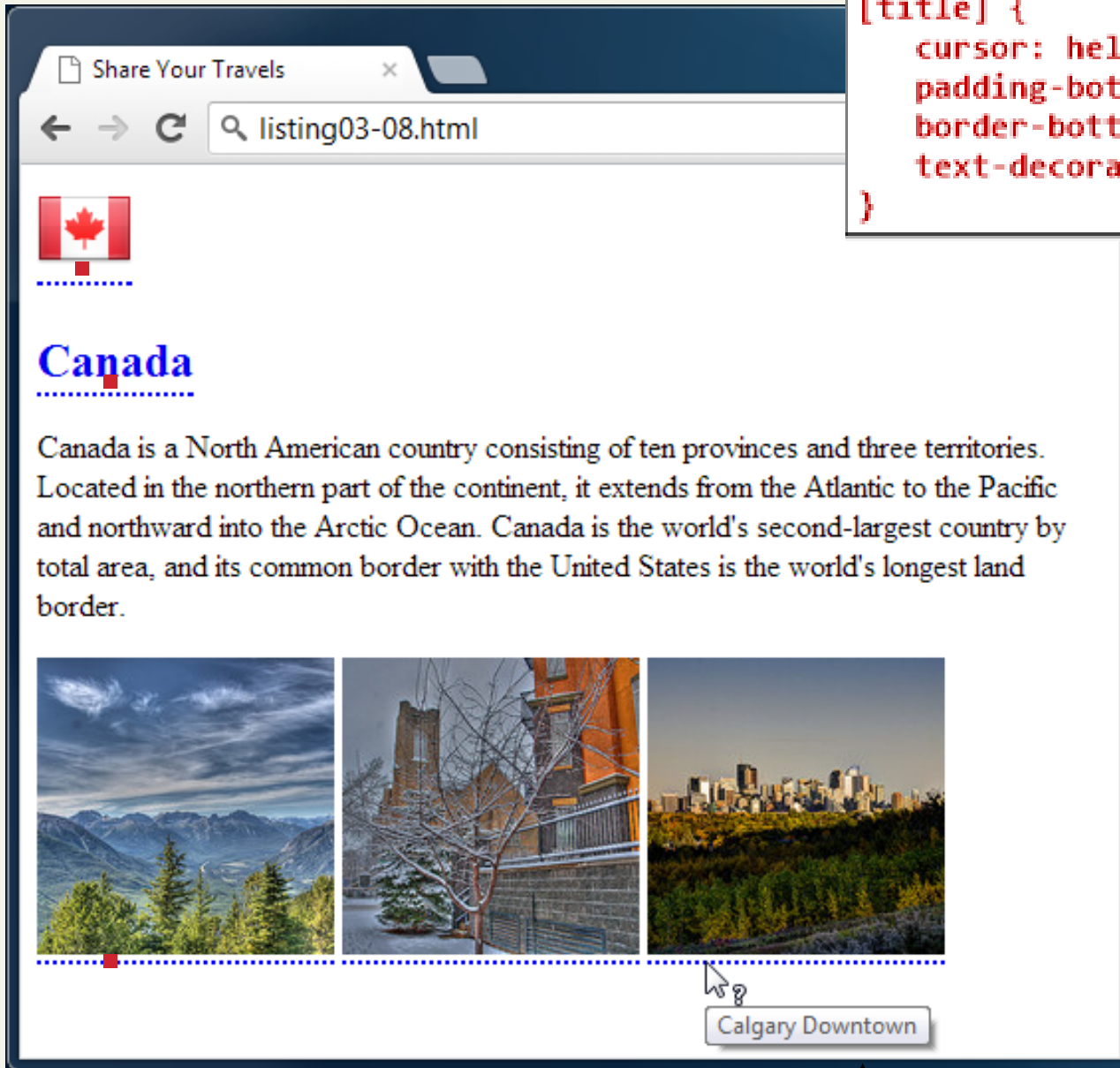


Example

# Attribute Selectors

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

- padding



# Selectors

## Attribute Selectors

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"

# Selectors

Example 1: Selector

```
<!DOCTYPE html>
<html>
<head>
<style>

  [title*="hello"]{
    color: brown;
  }

  a[href$=".pdf"]{
    color:red;
  }

</style>
</head>
<body>
```

**<b title="hello welcome ">** matches any element whose title attribute contains the keyword welcome.

```
</b>
<a href="example.pdf">click<a/>
<a href="example.html">click<a/>
</body>
</html>
```

matches any element whose title attribute contains the keyword welcome.  
[click](#) [click](#)

# Pseudo Selectors

Select something that does not exist explicitly as an element

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

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

A **pseudo-class selector** does apply to an HTML element, but targets either a particular state or, in CSS3, a variety of family relationships.

The most common use of this type of selectors is for targeting link states.

# Pseudo Selectors

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
    color: red;
}

/* visited link */
a:visited {
    color: green;
}

/* mouse over link */
a:hover {
    color: hotpink;
}

/* selected link */
a:active {
    color: blue;
}
</style>
</head>
<body>
```

Select something that does not exist explicitly as an element

```
<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p>
</body>
</html>
```

This is a link

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.

Note: a:active MUST come after a:hover in the CSS definition in order to be effective.

This is a link

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.

Note: a:active MUST come after a:hover in the CSS definition in order to be effective.

This is a link

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.

Note: a:active MUST come after a:hover in the CSS definition in order to be effective.

This is a link

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.

Note: a:active MUST come after a:hover in the CSS definition in order to be effective.

# Pseudo 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 3.8** Styling a link using pseudo-class selectors

# Pseudo Selectors

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



# Pseudo Selectors

```
input:focus {  
  background-color: yellow;  
}
```

First name:

First name:

```
p:first-letter {  
  font-size: 200%;  
  color: #8A2BE2;  
}
```

My name is

# Contextual Selectors

Select elements based on their ancestors, descendants, or siblings

A **contextual selector** (in CSS3 also called **combinators**) allows you to select 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.

# Contextual Selectors

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

# Descendant Selector


Selects all elements that are contained within another element

The descendant selector matches all elements that are descendants of a specified 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"

# Descendant Selector

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>
    <span><p>Paragraph 3 in the div.</p></span>
</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.

# Child Selector

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>
    <span><p>Paragraph 3 in the div.</p></span>
</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.

# Adjacent Sibling Selector

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:

# Adjacent Sibling Selector

```
<!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.



# General Sibling Selector

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:

# General Sibling Selector

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

<p>Paragraph 1.</p>

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

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

</body>
</html>
```

Paragraph 1.

Some code.

Paragraph 2.

Paragraph 3.

Some code.

Paragraph 4.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
div p:first-child {
```

```
    background-color: yellow;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<p>This paragraph is the first child of its parent (body).</p>
```

```
<h1>Welcome to My Homepage</h1>
```

```
<p>This paragraph is not the first child of its parent.</p>
```

```
<div>
```

```
<p>This paragraph is the first child of its parent (div).</p>
```

```
<p>This paragraph is not the first child of its parent.</p>
```

```
</div>
```

```
<p><b>Note:</b> For :first-child to work in IE8 and earlier, a DOCTYPE must be  
declared.</p>
```

```
</body>
```

```
</html>
```

This paragraph is the first child of its parent (body).

## Welcome to My Homepage

This paragraph is not the first child of its parent.

This paragraph is the first child of its parent (div).

This paragraph is not the first child of its parent.

**Note:** For :first-child to work in IE8 and earlier, a DOCTYPE must be declared.

# Exercise

```
<!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>
    <span><p>Paragraph 3 in the div.</p></span>
</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.

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.

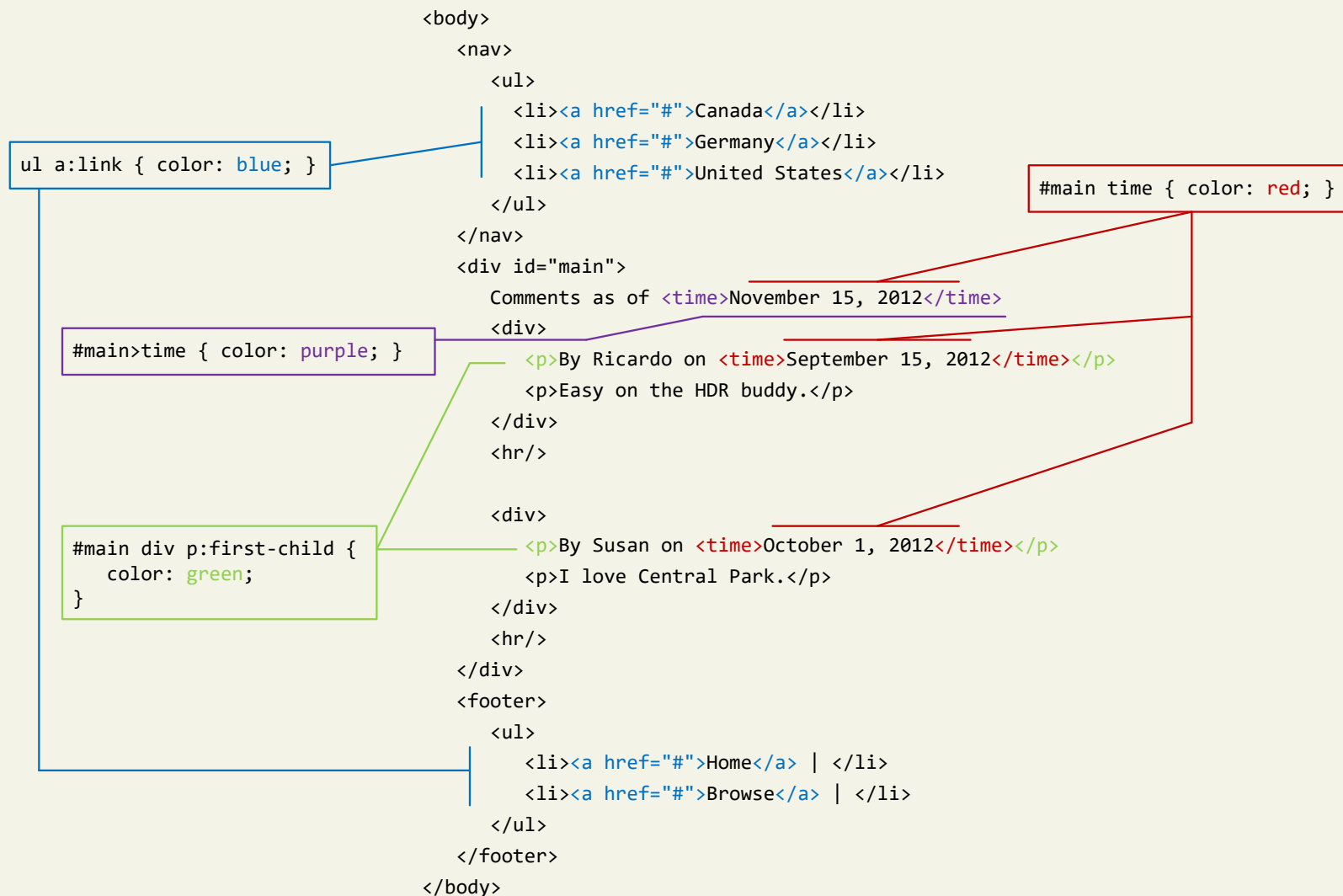
```
<!DOCTYPE html>
<html>
<head>
<style>
div > p {
    background-color: yellow;
}
</style>
</head>
<body>

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

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

</body>
</html>
```

# Contextual Selectors in Action



Section 5 of 7

# THE CASCADE: HOW STYLES INTERACT

# Cascade Principles

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

- **inheritance,**
- **specificity,**
- **location**

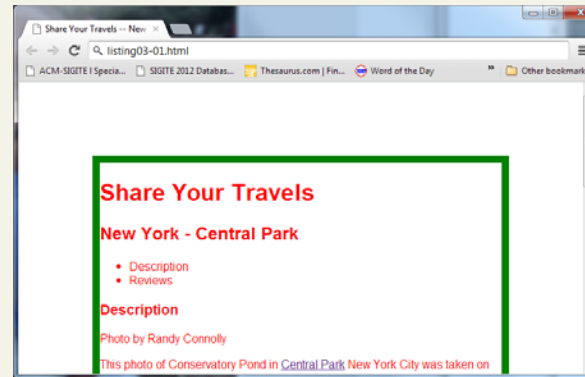
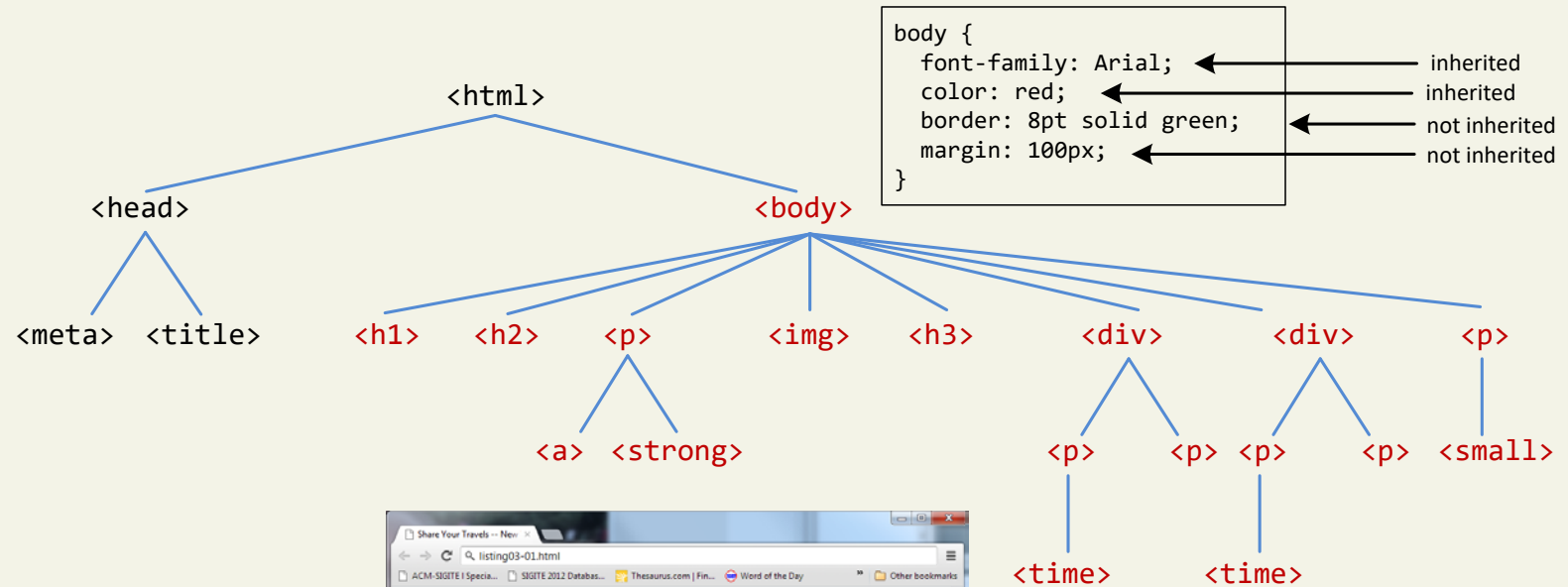
# Inheritance

## Cascade Principle #1

- Font, color, list, and text properties are inheritable.
- Layout, sizing, border, background and spacing properties are not.



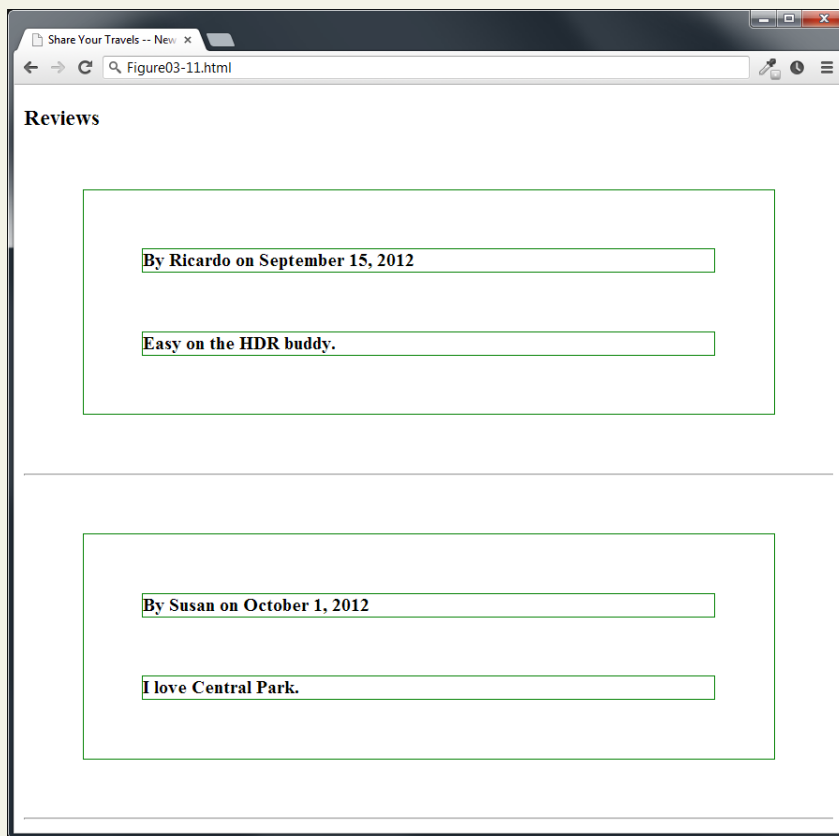
# Inheritance



- Font, color, list, and text properties are inheritable.
- Layout, sizing, border, background and spacing properties are not.

# Inheritance

How to force inheritance

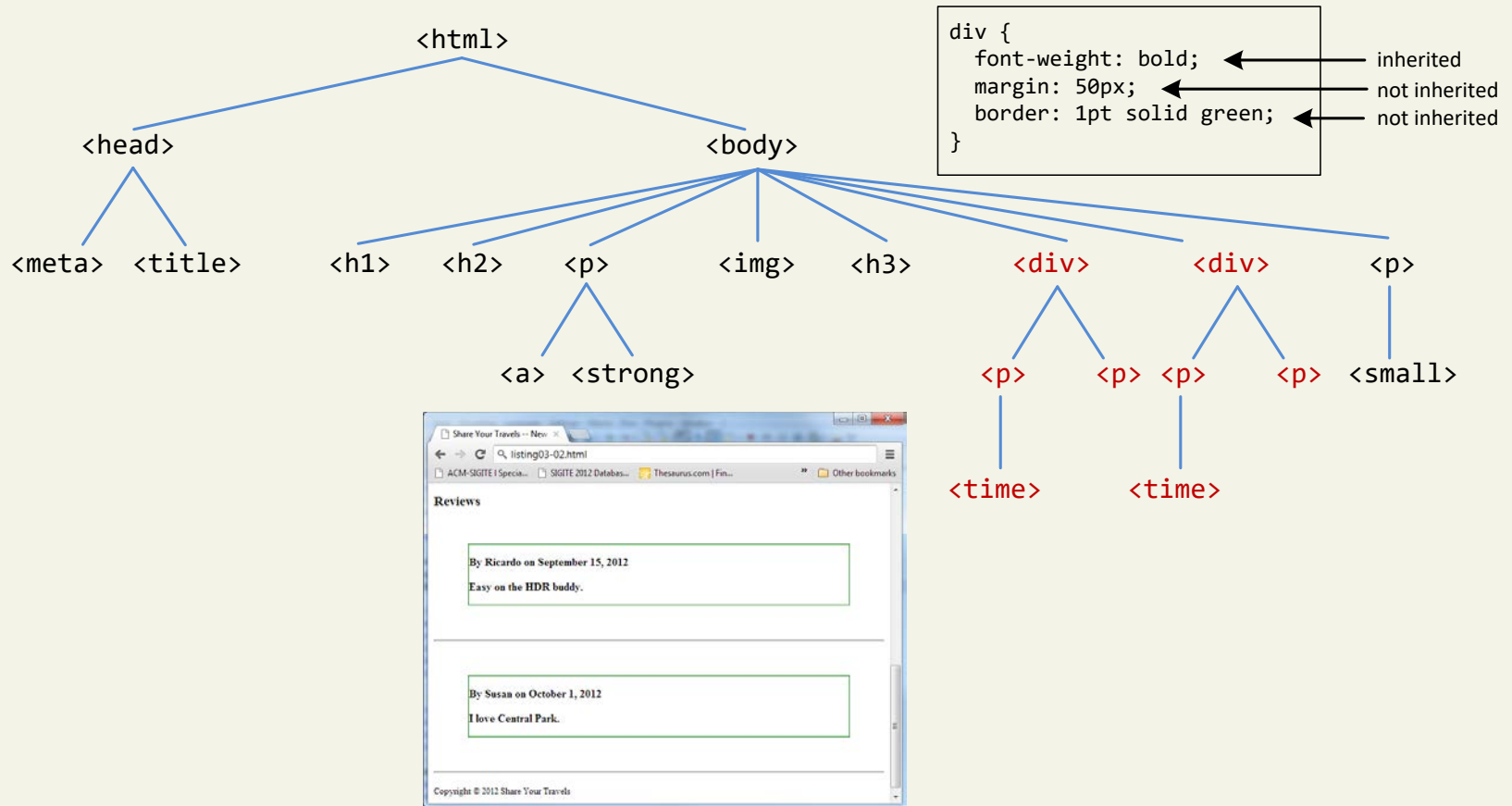


```
div {  
  font-weight: bold;  
  margin: 50px;  
  border: 1pt solid green;  
}  
p {  
  border: inherit;  
  margin: inherit;  
}
```

```
<h3>Reviews</h3>  
<div>  
  <p>By Ricardo on <time>September 15, 2012</time></p>  
  <p>Easy on the HDR buddy.</p>  
</div>  
<hr/>  
  
<div>  
  <p>By Susan on <time>October 1, 2012</time></p>  
  <p>I love Central Park.</p>  
</div>  
<hr/>
```

It is possible to tell elements to inherit properties that are normally not inheritable.

# Inheritance



# Specificity

## Cascade Principle #2

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

The more *specific* the selector, the more it takes precedence (i.e., overrides the previous definition).

# Specificity

How it works

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.

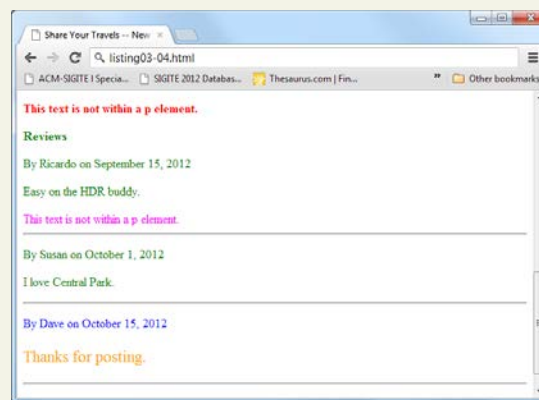
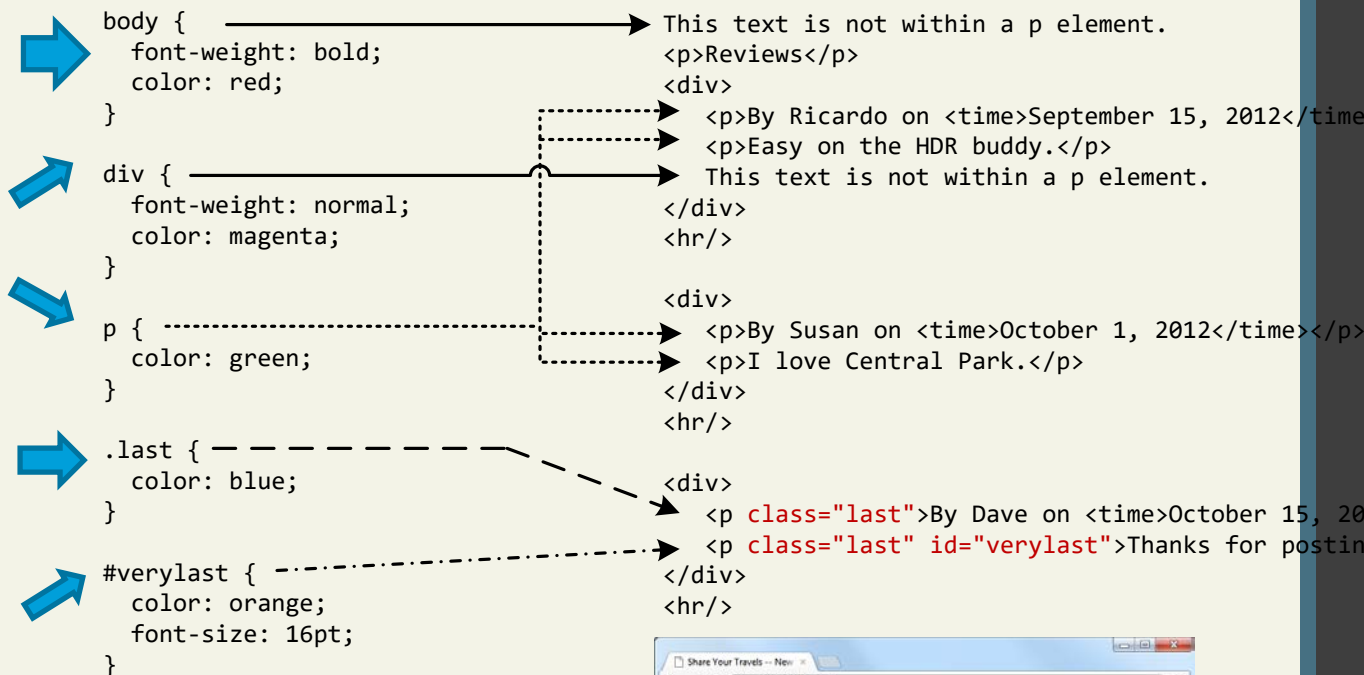
# Specificity

These color and font-weight properties are inheritable and thus potentially applicable to all the child elements contained within the body.

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**.

**Class selectors** are **more specific** than element selectors, and thus take precedence and override element selectors.

**Id selectors** are **more specific** than class selectors, and thus take precedence and override class selectors.



# Location

## Cascade Principle #3

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.**

# Location

```
— <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>
```



Section 6 of 7

# THE BOX MODEL