CSS: Introduction

PV219, spring 2022

Agenda

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 Selectors Level 3/4 – teaser
8. CSS Text Styling

What is CSS?

What is CSS?

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), within the **<head>** element, or, most commonly, in a separate text file that contains only CSS.

Benefits of CSS

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

- The degree of formatting control in CSS is significantly better than that provided in HTML.
- Web sites become significantly more maintainable because all formatting can be centralized into one, or a small handful, of CSS files.
- CSS-driven sites are more accessible.
- 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 markup.
- CSS can be used to adopt a page for different output mediums.

CSS Versions

Let's just say there's more than 1

- W3C published the CSS Level 1 Recommendation in 1996.
- 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.

Browser Adoption

While Microsoft’s Internet Explorer was an early champion of CSS, its later versions (especially IE5, IE6, and IE7) for Windows had uneven support for certain parts of CSS2.

In fact, all browsers have left certain parts of the *CSS2 Recommendation* unimplemented.

CSS has a reputation for being a somewhat frustrating language.

- this reputation is well deserved!

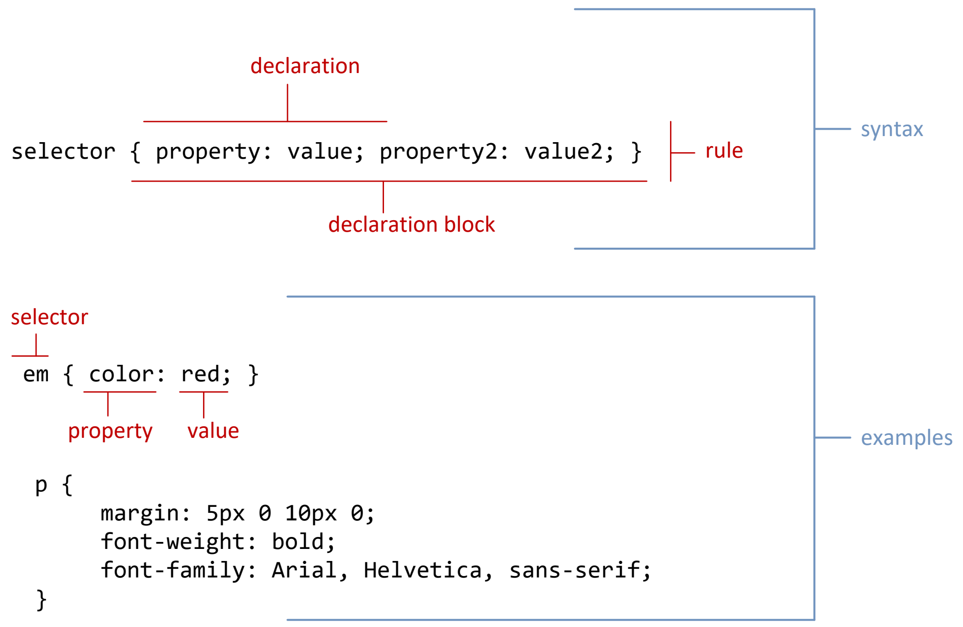
CSS Syntax

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, followed by a series of **property** and **value** pairs (each pair is also called a **declaration**).



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

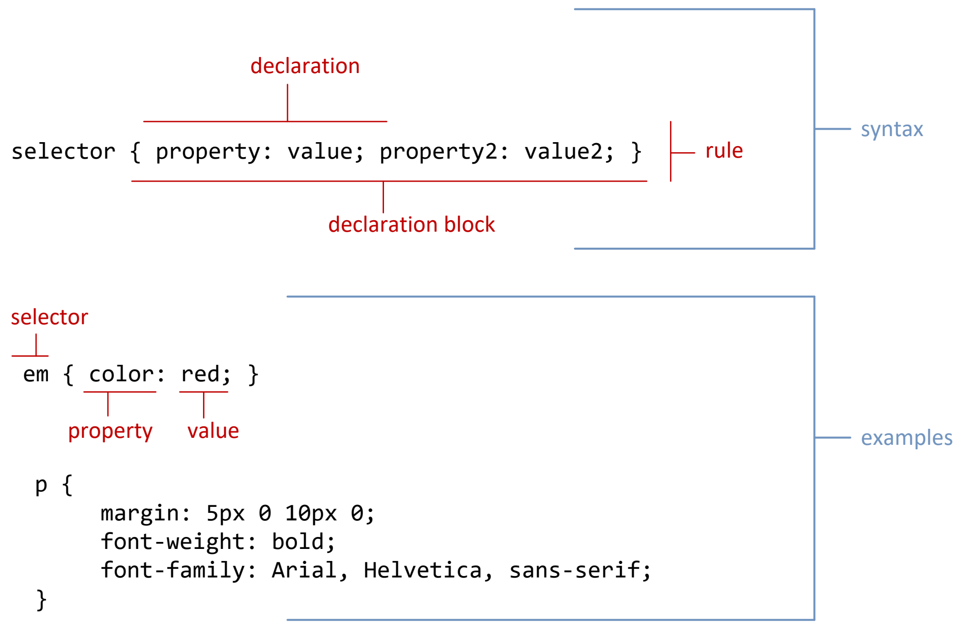
```
em { color: red; }
```

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

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.



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

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

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

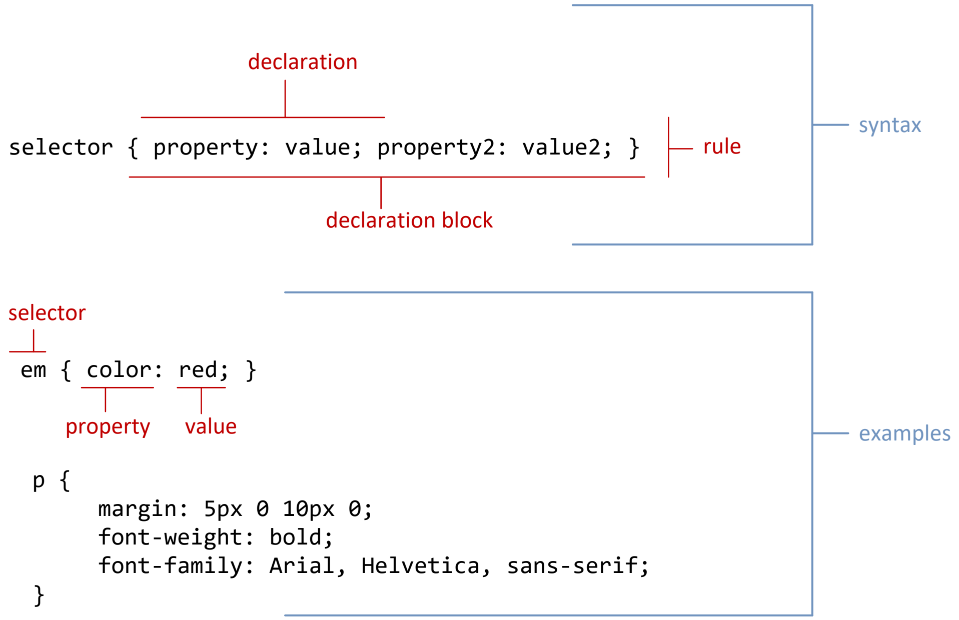
Selectors

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.

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 the style.



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

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

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

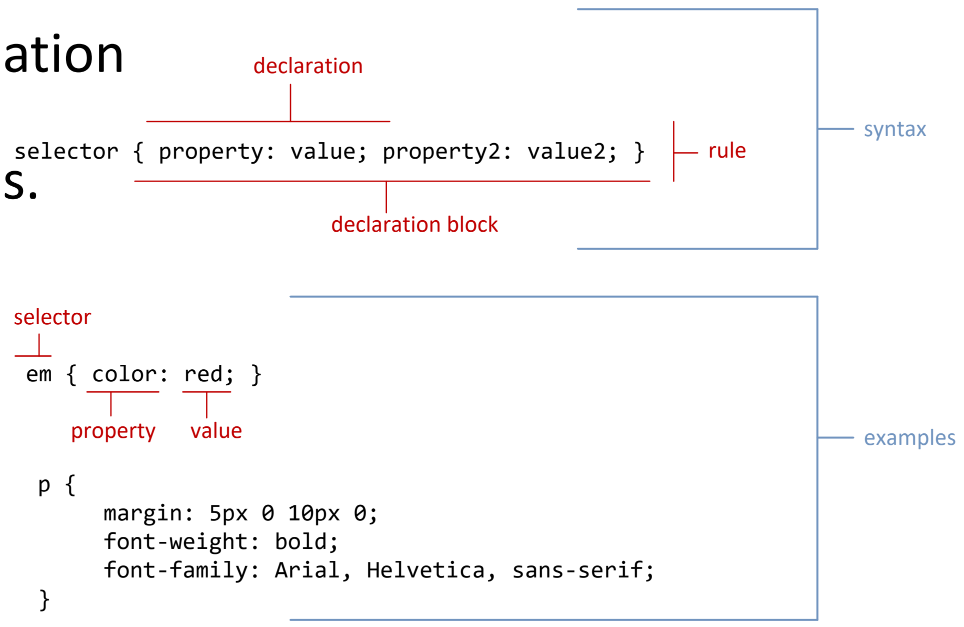
Properties

Which style properties of the selected elements

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

These property names are predefined by the CSS standard.

The CSS2.1 Recommendation defines over a hundred different property names.



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

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

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

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

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

Values

What style value for the properties

Each CSS declaration also contains a **value** for a property.

- 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.

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%);

Units of Measurement

There are multiple ways of specifying a unit of measurement 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 to **avoid using absolute units**.
>
> Pixels are perhaps the one popular exception (though as we shall see later there are also good reasons for avoiding the pixel unit).

Relative Units

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 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 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)

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

Comments in CSS

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

/* comment goes here */

Location of Styles

Actually there are three …

Different types of style sheet

Author-created style sheets (what we are learning in this presentation).

User style sheets allow the individual user to tell the browser to display pages using that individual's own custom style sheet. This option is available in a browser usually in its accessibility options area.

The **browser style sheet** defines the default styles the browser uses for each HTML element.

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

You can combine all 3.

Inline Styles

Style rules placed within an HTML element via the style attribute

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

LISTING 3.1 Internal styles example

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.

Embedded Style Sheet

Style rules placed within an HTML element via the style attribute

```
<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.

External 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>
   <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

Selectors

Selectors

Things that make your life easier

When defining CSS rules, you will need to first need to use a **selector** to tell the browser which elements will be affected.

CSS selectors allow you to select

- individual elements
- multiple HTML elements,
- elements that belong together in some way, or
- elements that are positioned in specific ways in the document hierarchy.

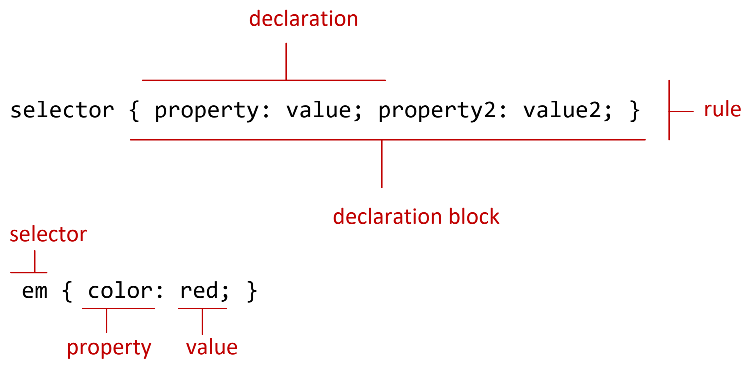
There are a number of different selector types.

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.



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

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

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

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;
}
```

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.

These 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.

Class Selectors

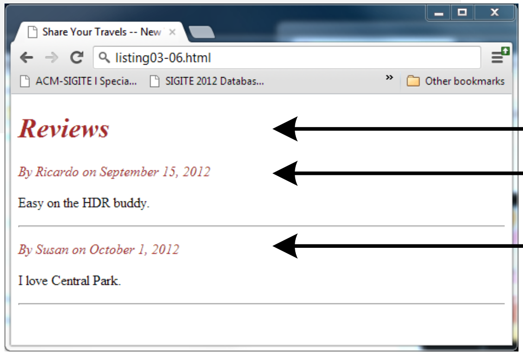
Simultaneously target different HTML elements

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.

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>
```



Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

By Susan on October 1, 2012

I love Central Park.

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

Class Selectors

- You can also specify that only specific HTML elements should be affected by a class. In the example below, only <p> elements with class="center" will be center-aligned:

```
p.center {text-align: center;}
```

- HTML elements can also refer to more than one class. In the example below, the <p> element will be styled according to class="center" and to class="large":

```
<p class="center large">This paragraph refers
to two classes.</p>
```

Id Selectors

Target a specific element by its id attribute

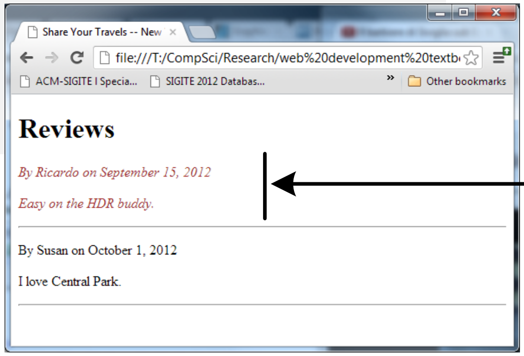
An **id selector** allows you to target a specific element by its id attribute regardless of its type or position.

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>
```


Share Your Travels -- New
file:///T:/CompSci/Research/web%20development%20textb
ACM-SIGITE | Specia...
SIGITE 2012 Databas...
Other bookmarks

Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

By Susan on October 1, 2012

I love Central Park.

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

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.

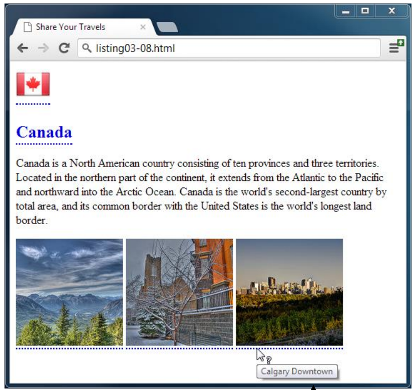
This can be a very powerful technique, but because of uneven historical support by some of the browsers, not all web authors have used them. All modern browsers have full support.

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>
      <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 Sulpher Mountain">
          <img src="images/square/6592317633.jpg" title="Grace Presbyterian Church">
          <img src="images/square/6592914823.jpg" title="Calgary Downtown">
       </div>
   </div>
</body>
```

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



Canada

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.

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.

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

```
<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

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

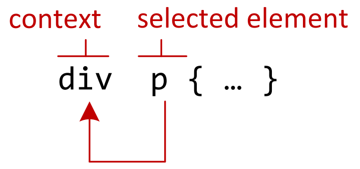
Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element	**div p** Selects a <p> element that is contained somewhere within a <div> element. That is, the <p> can be any descendant, not just a child.
Child	A specified element that is a direct child of the specified element	**div > h2** Selects an <h2> element that is a child of a <div> element.
Adjacent Sibling	A specified element that is the next sibling (i.e., comes directly after) of the specified element.	h3+p Selects the first <p> after any <h3>.
General Sibling	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>.

Descendant Selector

Selects all elements that are contained within another element

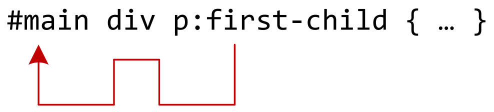
While some of these contextual selectors are used relatively infrequently, almost all web authors find themselves using descendant selectors.

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



```
div  p { … }
```

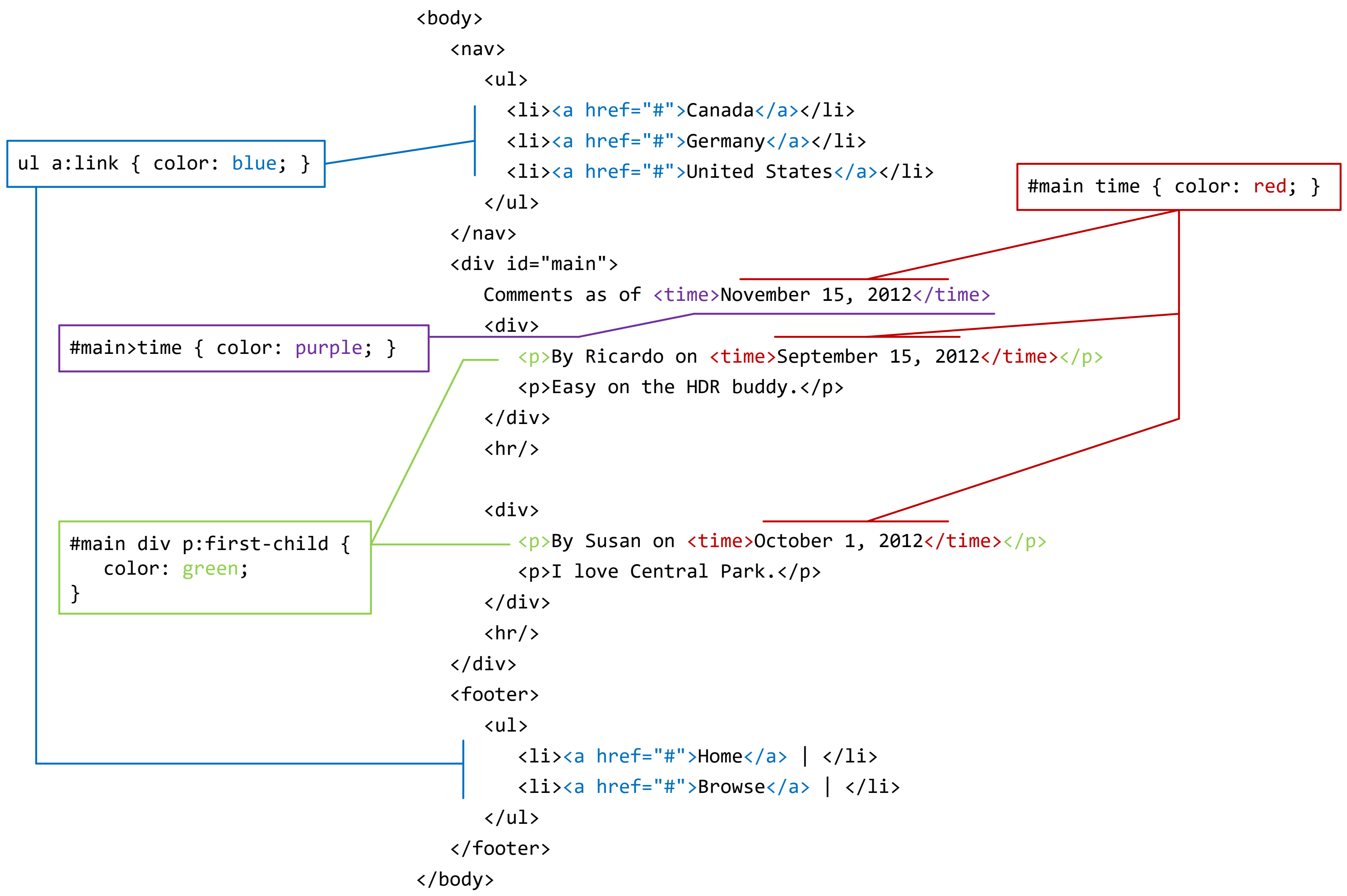
Selects a `<p>` element somewhere within a `<div>` element



```
#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"`

Contextual Selectors in Action



```
<body>
   <nav>
      <ul>
        <li><a href="#">Canada</a></li>
        <li><a href="#">Germany</a></li>
        <li><a href="#">United States</a></li>
      </ul>
   </nav>
   <div id="main">
      Comments as of <time>November 15, 2012</time>
      <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/>
   </div>
   <footer>
      <ul>
         <li><a href="#">Home</a> | </li>
         <li><a href="#">Browse</a> | </li>
      </ul>
   </footer>
</body>
```

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

```
#main time { color: red; }
```

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

```
#main div p:first-child {
   color: green;
}
```

The Cascade: How Styles Interact

Why Conflict Happens

In CSS that is

Because

- there are three different types of style sheets (author-created, user-defined, and the default browser style sheet),
- author-created stylesheets 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.

Cascade

How conflicting rules are handled in CSS

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

The visual metaphor behind the term **cascade** is that of a mountain stream progressing downstream over rocks.

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.

Cascade Principles

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

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

Inheritance

Cascade Principle #1

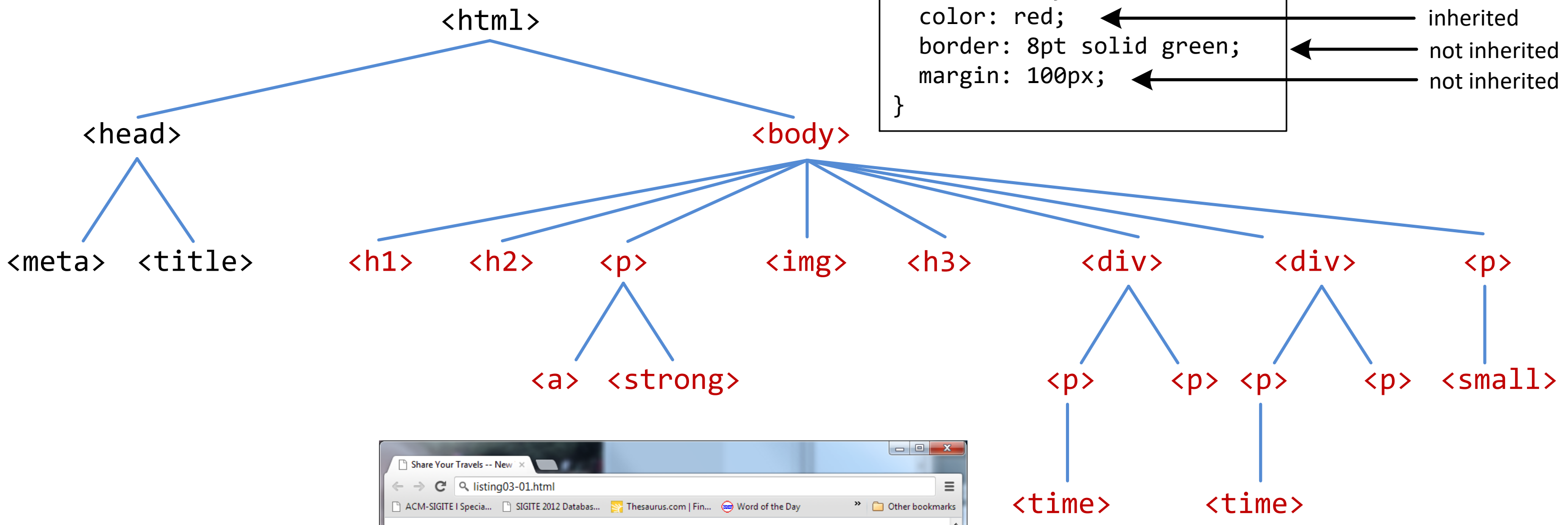
Many (but not all) CSS properties affect not only themselves but their descendants as well.

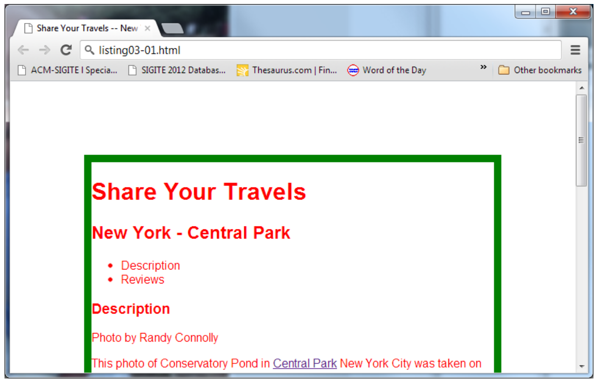
Font, color, list, and text properties are inheritable.

Layout, sizing, border, background and spacing properties are not.

Inheritance

```
body {
   font-family: Arial;       ← inherited
   color: red;               ← inherited
   border: 8pt solid green;  ← not inherited
   margin: 100px;            ← not inherited
}
```





Share Your Travels

New York - Central Park

- Description
- Reviews

Description

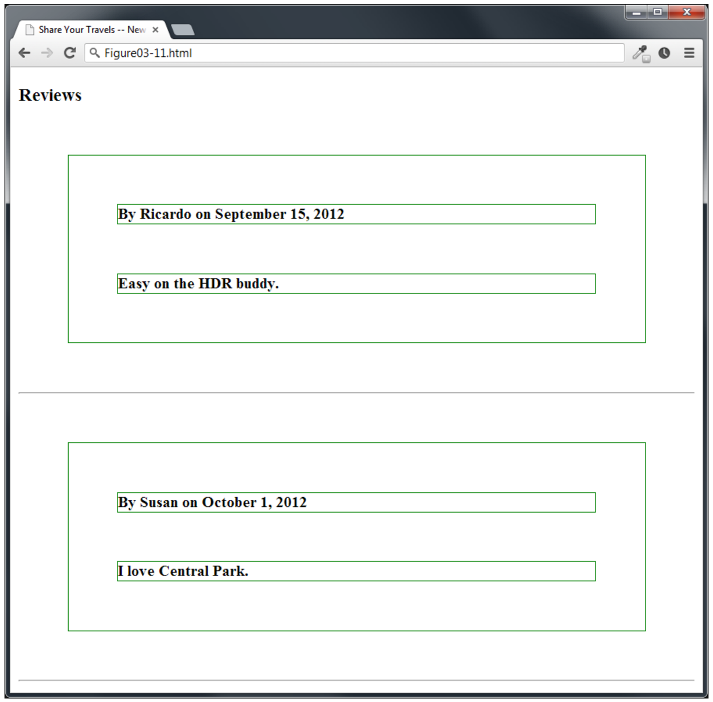
Photo by Randy Connolly

This photo of Conservatory Pond in Central Park New York City was taken on

Inheritance

How to force inheritance

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



Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

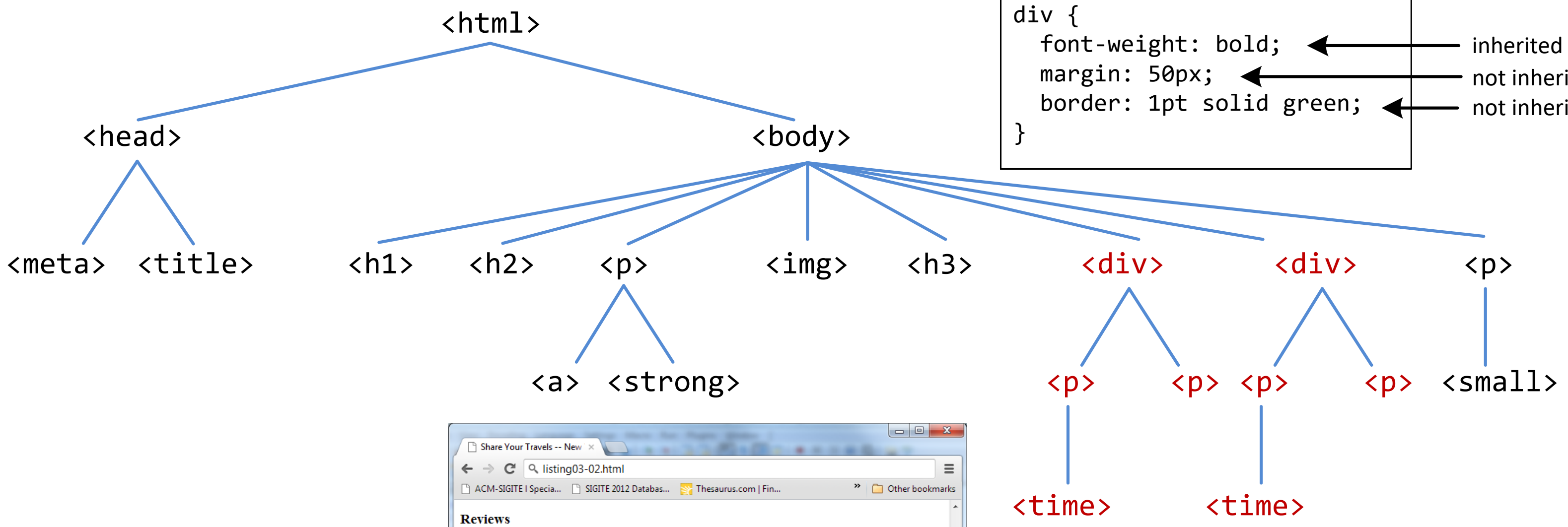
By Susan on October 1, 2012

I love Central Park.

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

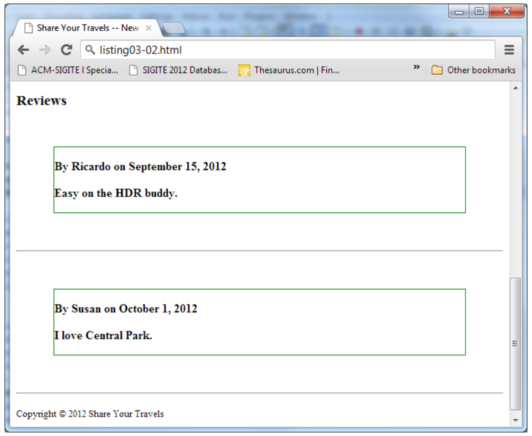
```
<h3>Reviews</h3>
<div>
   <p>By Ricardo on <time>September 15, 2012</time></p>
   <p>Easy on the HDR buddy.</p>
</div>
<hr/>

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

Inheritance



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

inherited
not inherited
not inherited



Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

By Susan on October 1, 2012

I love Central Park.

Copyright © 2012 Share Your Travels

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.

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

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.

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

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

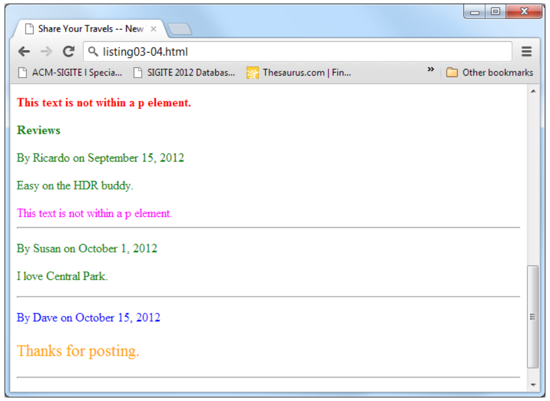
Class selectors are more specific than element selectors, and thus take precedence and override element selectors.

```
.last {
  color: blue;
}
```

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

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

```
This text is not within a p element.
<p>Reviews</p>
<div>
   <p>By Ricardo on <time>September 15, 2012</time
   <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, 2012</time></p>
   <p>I love Central Park.</p>
</div>
<hr/>

<div>
   <p class="last">By Dave on <time>October 15, 20
   <p class="last" id="verylast">Thanks for postin
</div>
<hr/>
```



This text is not within a p element.

Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

This text is not within a p element.

By Susan on October 1, 2012

I love Central Park.

By Dave on October 15, 2012

Thanks for posting.

Specificity Algorithm

The algorithm that is used to determine specificity is :

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

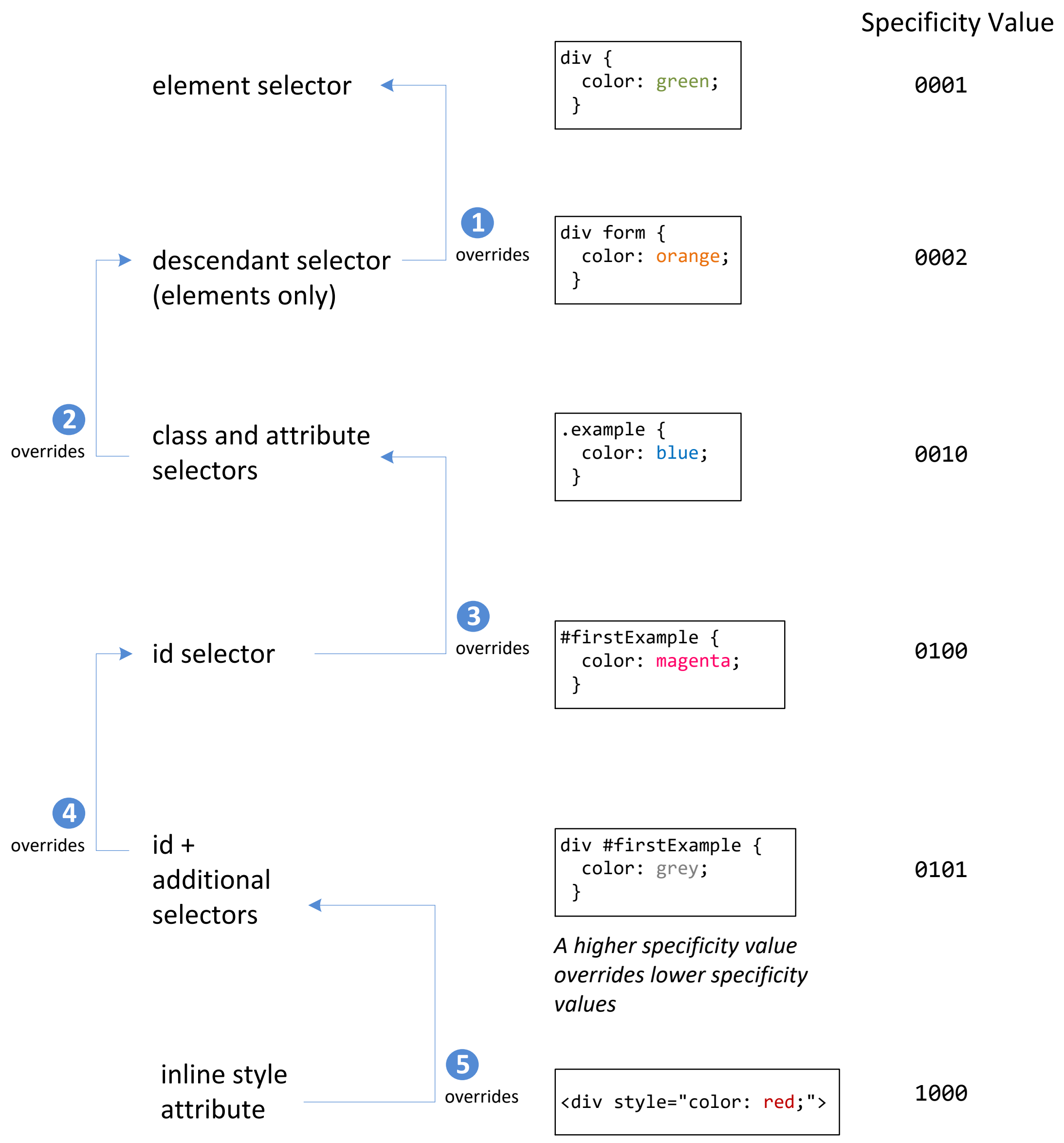
2. Count the number of ID attributes in the selector (let that value = b).

3. Count the number of other attributes and pseudo-classes in the selector (let that value = c).

4. Count the number of element names and pseudo-elements in the selector (let that value = d).

5. Finally, concatenate the four numbers a + b + c + d together to calculate the selector’s specificity.

Specificity Algorithm



Specificity Value

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

0001

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

0002

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

0010

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

0100

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

0101

A higher specificity value overrides lower specificity values

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

1000

Specificity Algorithm

```
*               /* a=0 b=0 c=0 -> specificity =   0 */
LI              /* a=0 b=0 c=1 -> specificity =   1 */
UL LI           /* a=0 b=0 c=2 -> specificity =   2 */
UL OL+LI        /* a=0 b=0 c=3 -> specificity =   3 */
H1 + *[REL=up]  /* a=0 b=1 c=1 -> specificity =  11 */
UL OL LI.red    /* a=0 b=1 c=3 -> specificity =  13 */
LI.red.level    /* a=0 b=2 c=1 -> specificity =  21 */
#x34y           /* a=1 b=0 c=0 -> specificity = 100 */
#s12:not(FOO)   /* a=1 b=0 c=1 -> specificity = 101 */
```

Tool to visualize it.

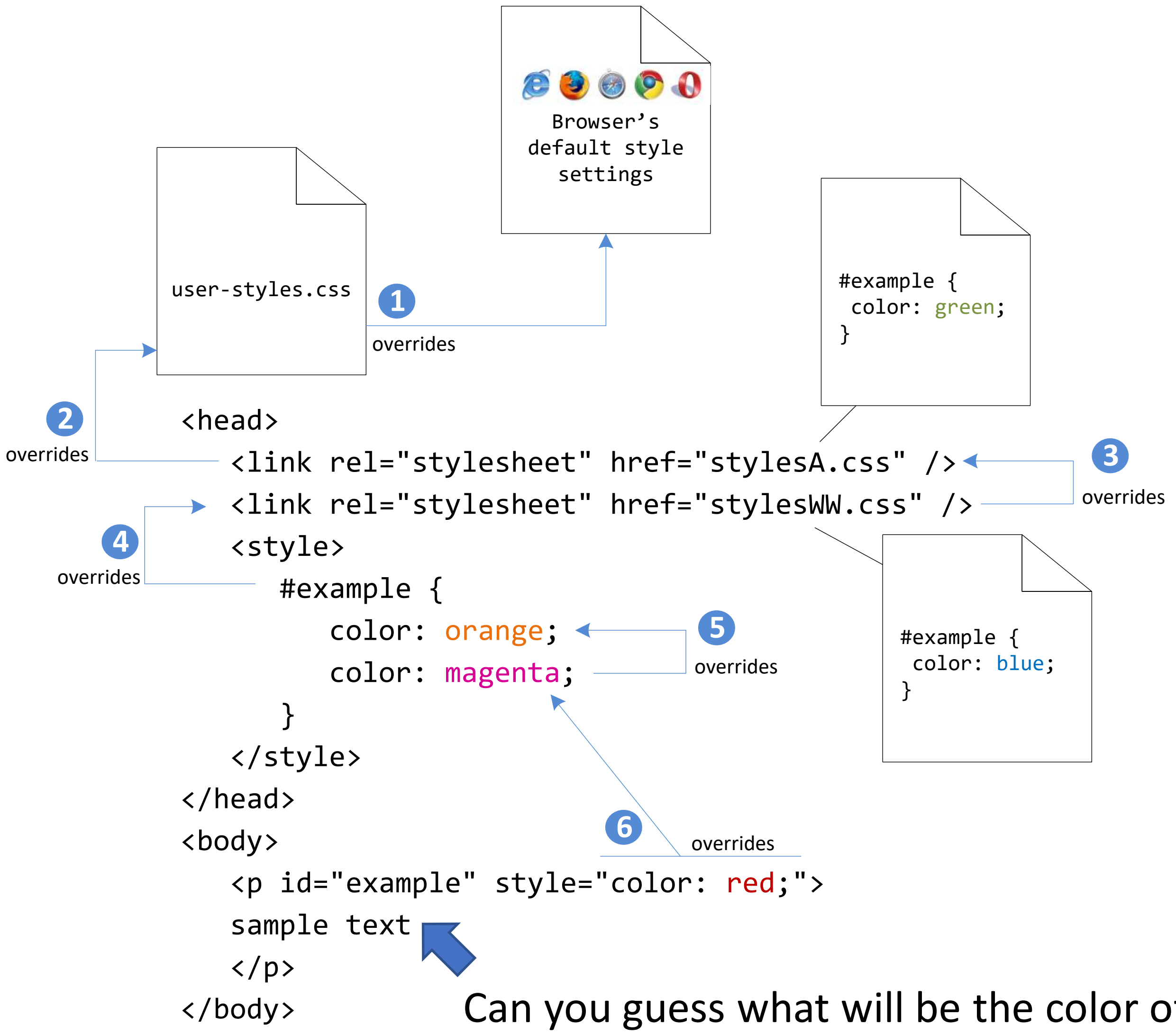
Location

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



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

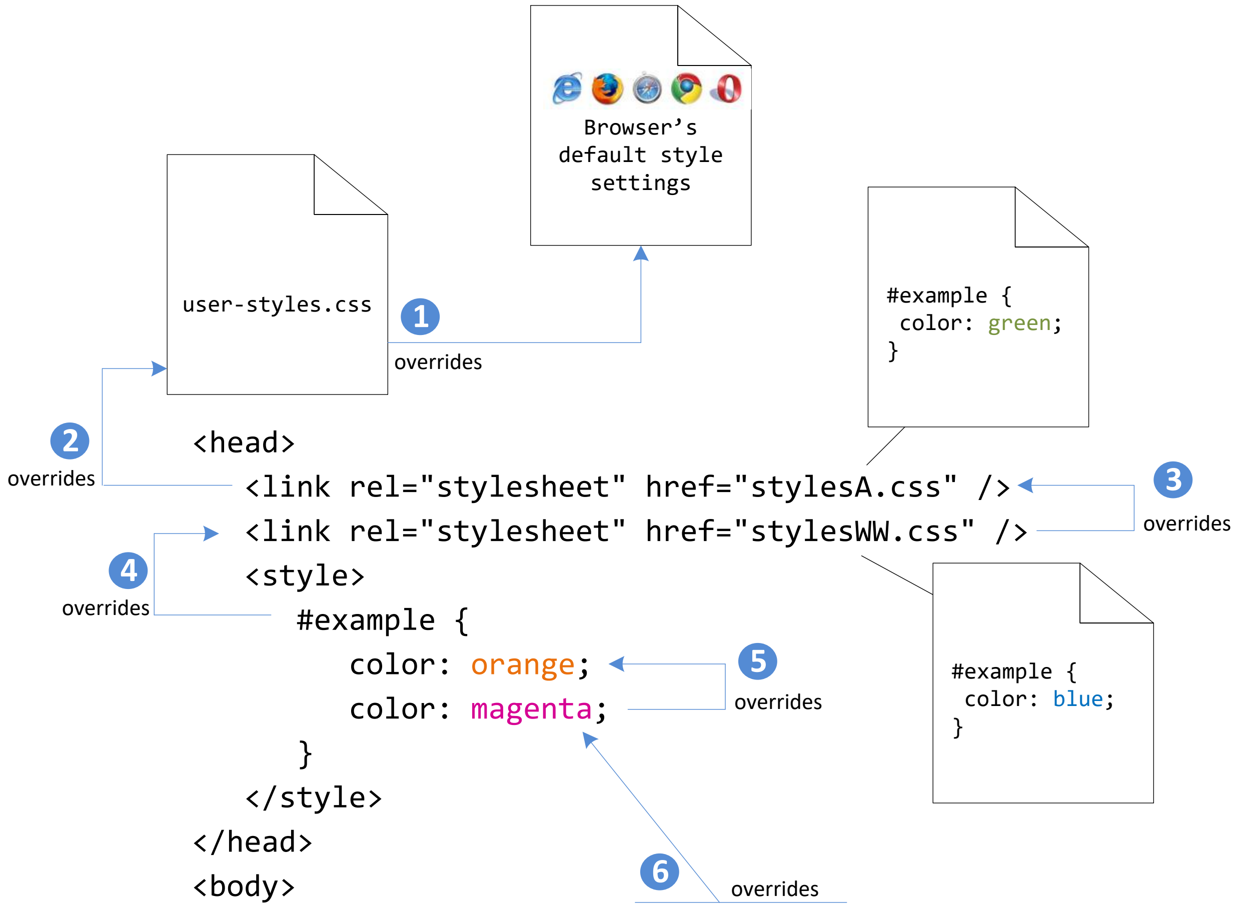
```
#example {
 color: blue;
}
```

```
<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>
```

Can you guess what will be the color of the sample text?

Location

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



```
#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>
```

Location

There's always an exception

There is one exception to the principle of location.

If a property is marked with !important in an author-created style rule, then it will override any other author-created style regardless of its location.

The only exception is a style marked with !important in an user style sheet; such a rule will override all others.

The Box Model

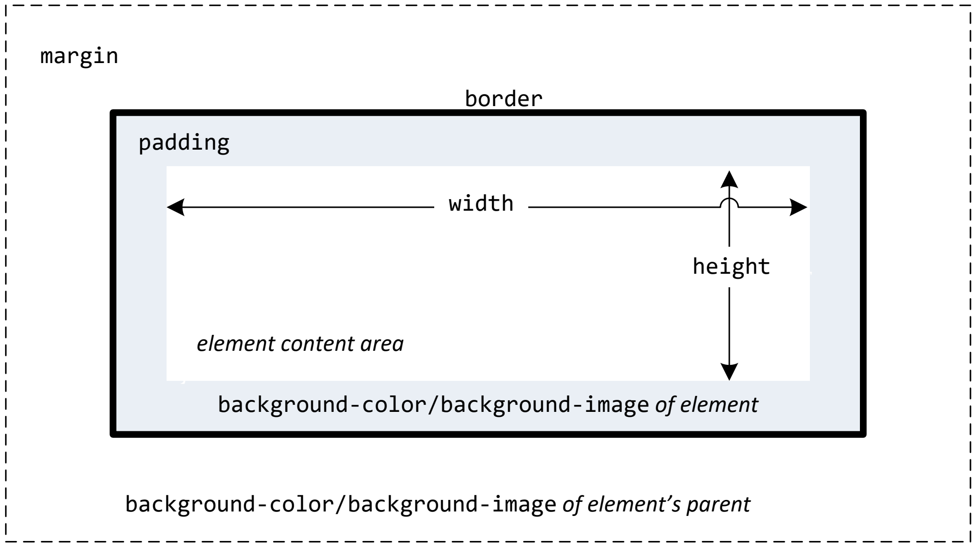
The Box Model

Time to think inside the box

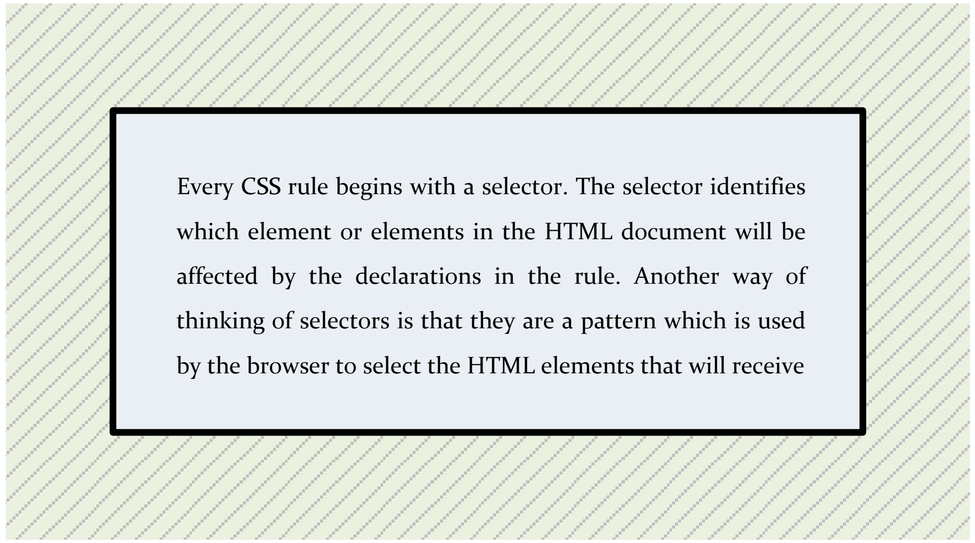
In CSS, all HTML elements exist within an **element box**.

It is absolutely essential that you familiarize yourself with the terminology and relationship of the CSS properties within the element box.

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*



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

Background

Box Model Property #1

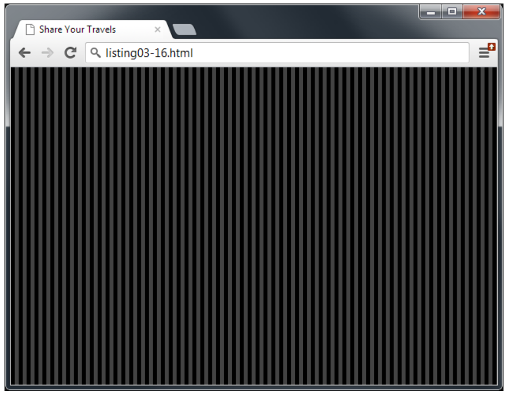
The background color or image of an element fills an element out to its border (if it has one that is).

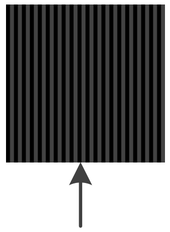
In contemporary web design, it has become extremely common too use CSS to display purely presentational images (such as background gradients and patterns, decorative images, etc) rather than using the <img /> element.

Background Properties

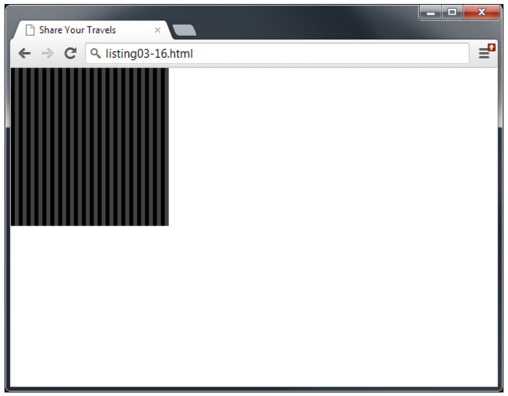
Property	Description
background	A combined short-hand property that allows you to set the background values in one property. While you can omit properties with the short-hand, do remember that any omitted properties will be set to their default value.
background-attachment	Specifies whether the background image scrolls with the document (default) or remains fixed. Possible values are: fixed, scroll.
background-color	Sets the background color of the element.
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.
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). 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.

Background Repeat

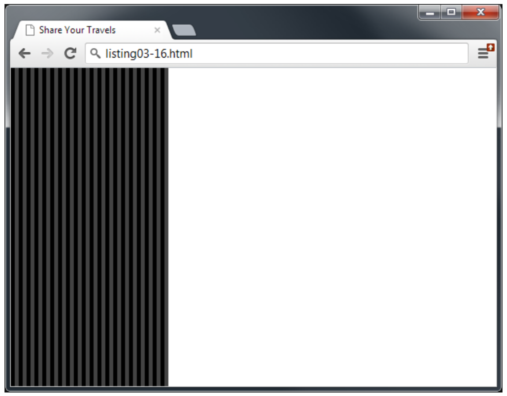




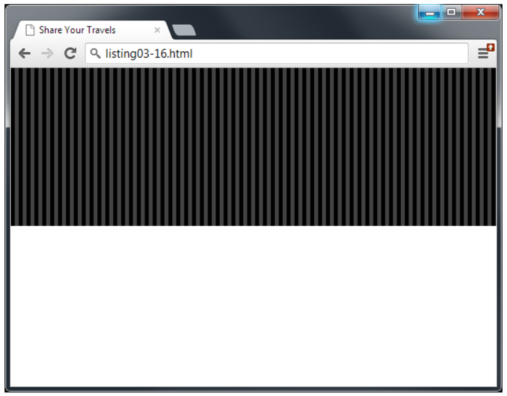
```
background-image: url(../images/backgrounds/body-background-tile.gif);
background-repeat: repeat;
```



```
background-repeat: no-repeat;
```

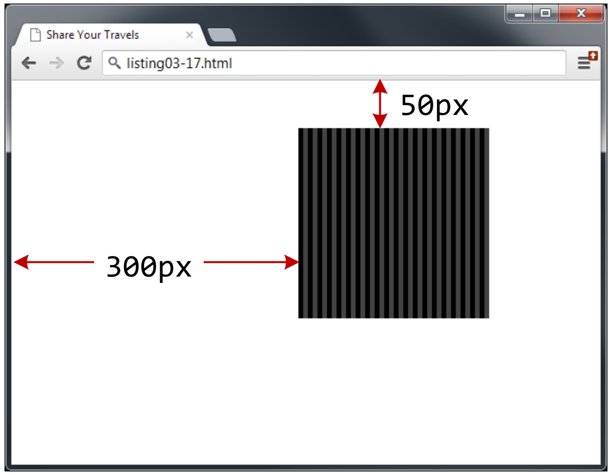


```
background-repeat: repeat-y;
```



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

Background Position



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

Borders

Box Model Property #2

Borders provide a way to visually separate elements.

You can put borders around all four sides of an element, or just one, two, or three of the sides.

Borders

Property	Description
border	A combined short-hand 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.

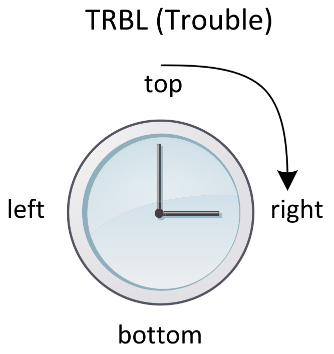
Shortcut notation

TRBL

With border, margin, and padding properties, there are long-form and shortcut methods to set the 4 sides

```
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 */

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

border-color: red green orange blue;     /* sets all four sides differently */
```

When using this multiple values shortcut, they are applied in clockwise order starting at the top.
Thus the order is: **top right bottom left.**

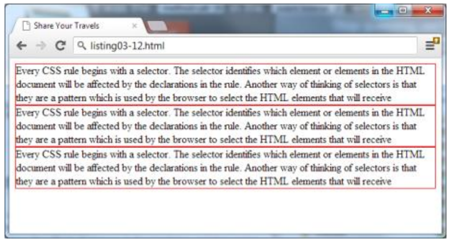


```
border-color: top right bottom left;
```

```
border-color: red green orange blue;
```

Margins and Padding

Box Model Properties #3 and #4

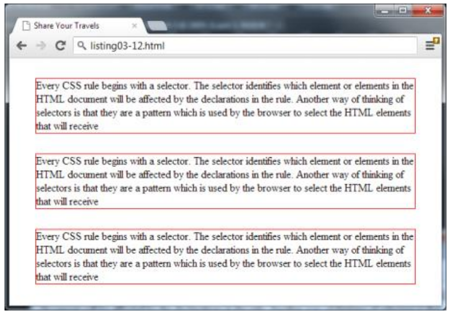


Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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;
}
```

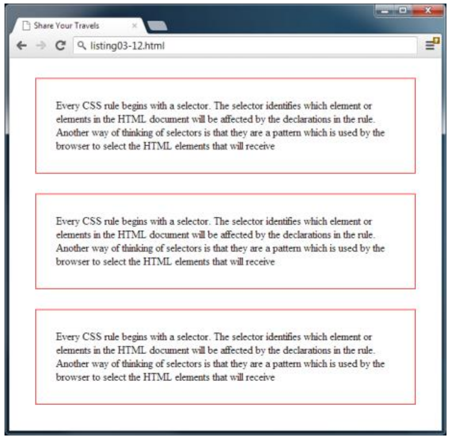


Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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

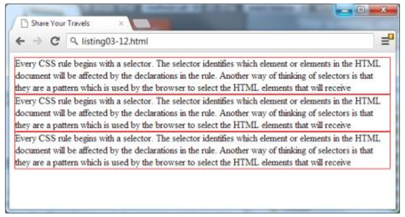
```
p {
   border: solid 1pt red;
   margin: 30px;
   padding: 30px;
}
```

Margins

Why they will cause you trouble.

Did you notice that the space between paragraphs one and two and between two and three is the same as the space before paragraph one and after paragraph three?

This is due to the fact that adjoining vertical margins collapse.


Share Your Travels

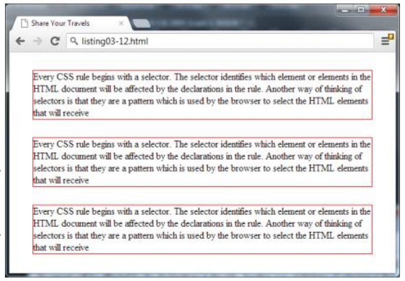
listing03-12.html

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;
}
```


Share Your Travels

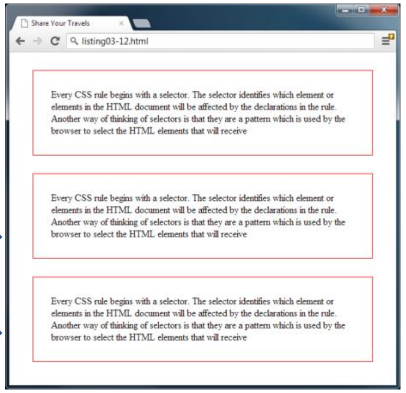
listing03-12.html

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;
}
```


Share Your Travels

listing03-12.html

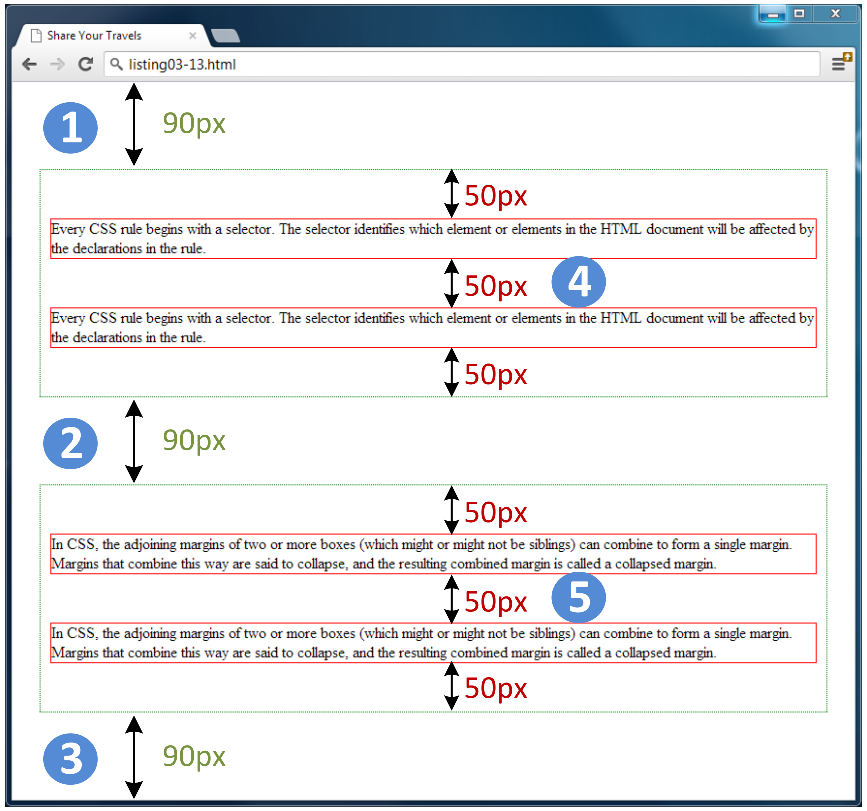
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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;
}
```

Collapsing Margins



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;
}
```

If overlapping margins did not collapse, then margin space for would be 180px 2 (90pixels for the bottom margin of the first <div> + 90 pixels for the top margin of the second <div>), while the margins 4 and 5 for would be 100px.

However, as you can see this is not the case.

Collapsing Margins

How it works

When the **vertical** margins of two elements touch,

- the largest margin value of the elements will be displayed
- the smaller margin value will be collapsed to zero.

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

To complicate matters even further, there are a large number of special cases in which adjoining vertical margins do **not** collapse.

Width and Height

Box Model Properties #5 and #6

The width and height properties specify the size of the element’s content area.

Perhaps the only rival for collapsing margins in troubling, box dimensions have a number of potential issues.

Width and Height

Potential Problem #1

Only block-level elements and non-text inline elements such as images have a **width** and **height** that you can specify.

By default the width of and height of elements is the actual size of the content.

For text,

- this is determined by the font size and font face;

For images,

- the width and height of the actual image in pixels determines the element box’s dimensions.

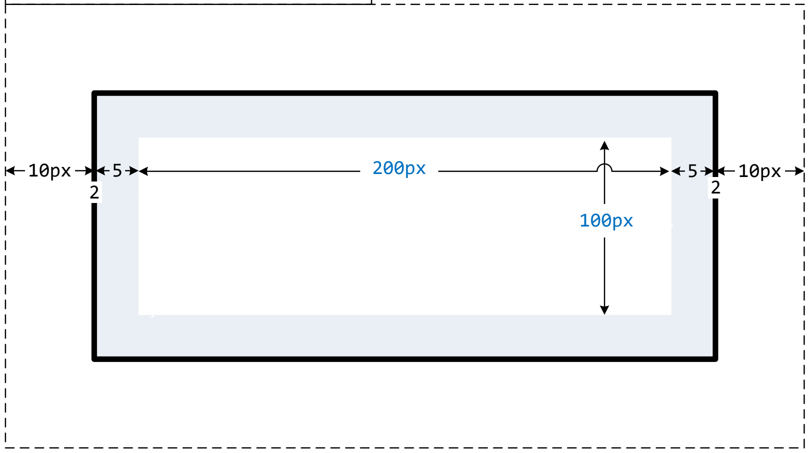
Width and Height

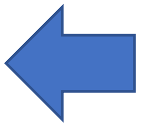
Potential Problem #2

Since the width and the height refer to the size of the content area, by default, the total size of an element is equal to not only its content area, but also to the sum of its padding, borders, and margins.

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

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

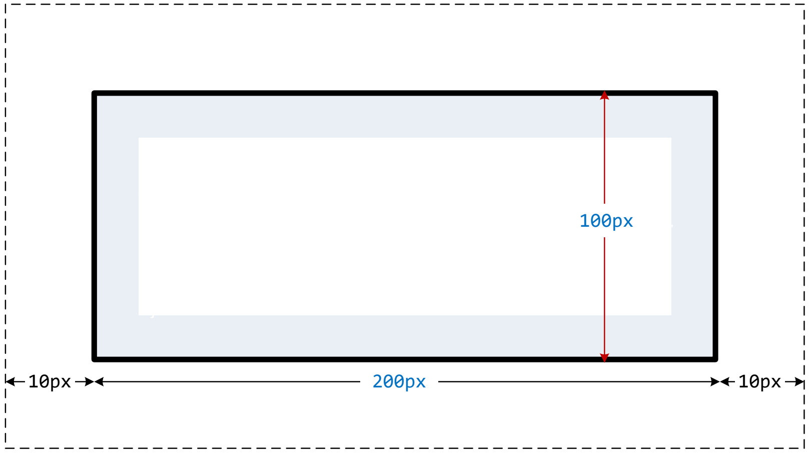




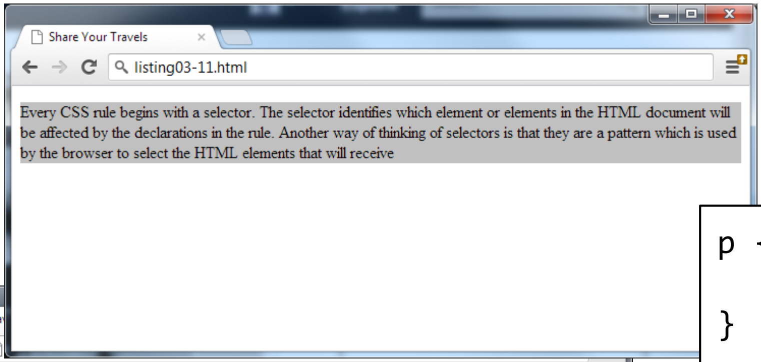
Default

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

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

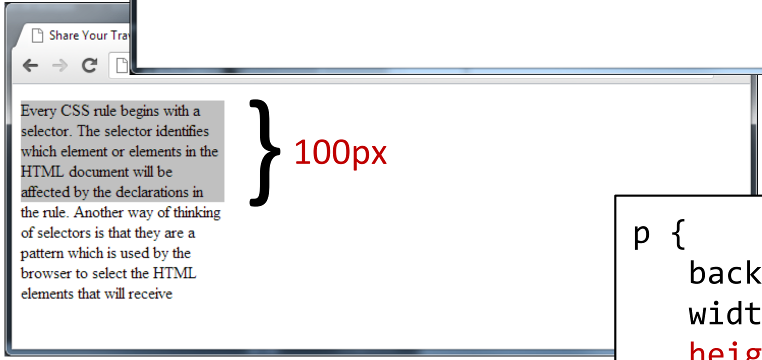


Width and Height



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 {
    background-color: silver;
}
```



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 {
    background-color: silver;
    width: 200px;
    height: 100px;
}
```

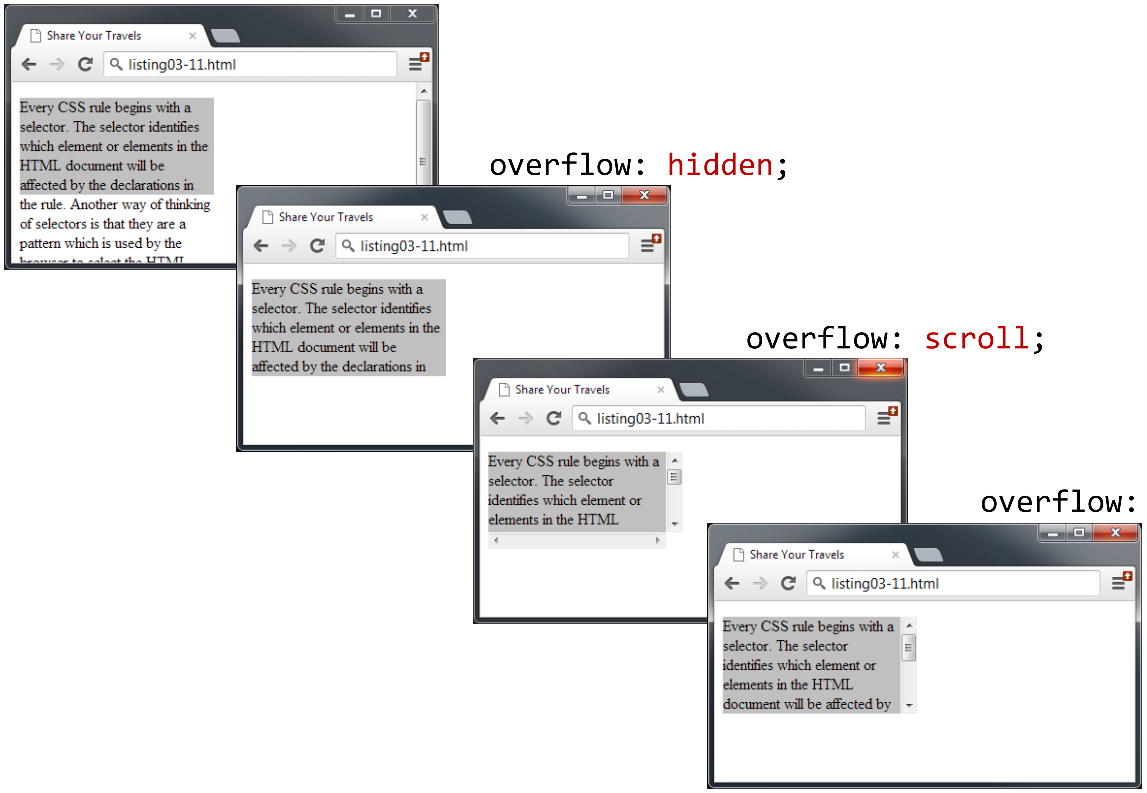
Overflow Property

`overflow: visible;`

`overflow: hidden;`

`overflow: scroll;`

`overflow: auto;`



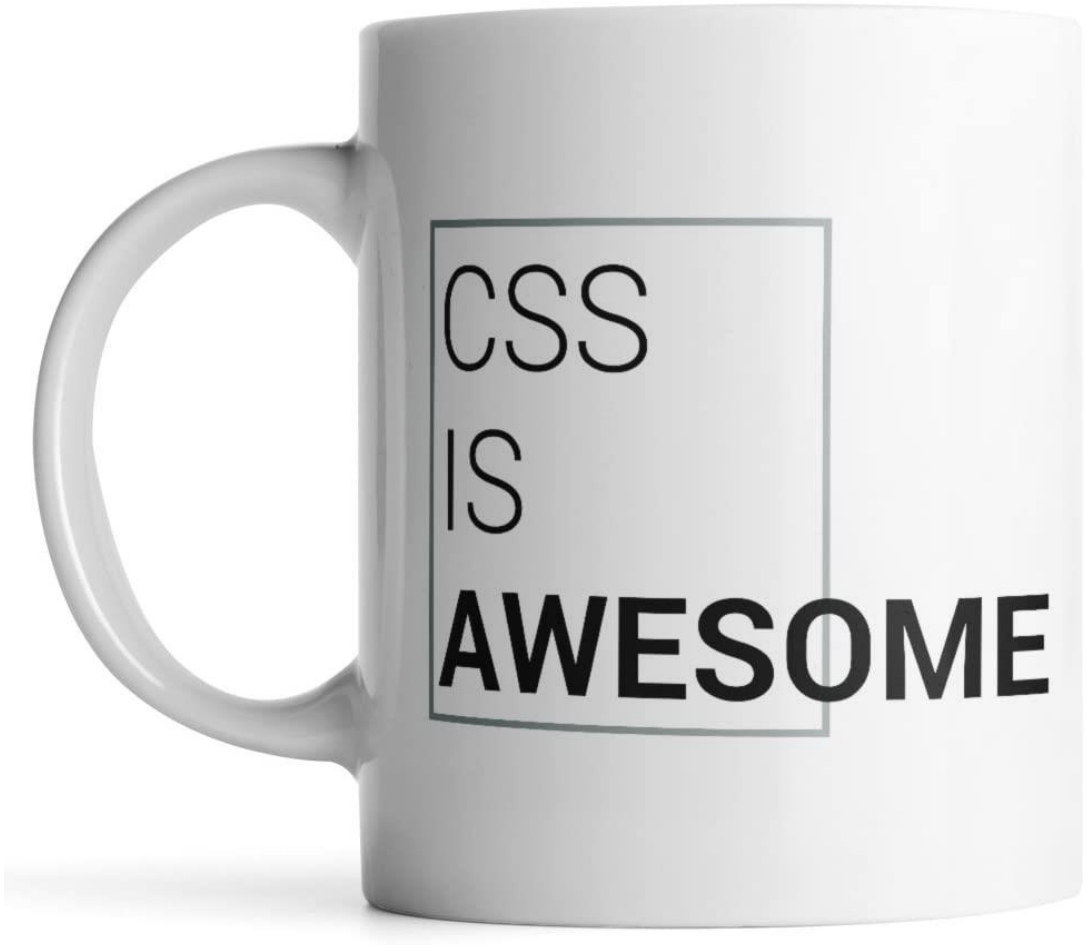
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML

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

Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML

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

Overflow Property



Sizing Elements

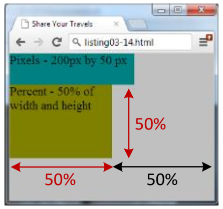
Time to embrace ems and percentages

While the previous examples used pixels for its measurement, many contemporary designers prefer to use **percentages** or **em** units for widths and heights.

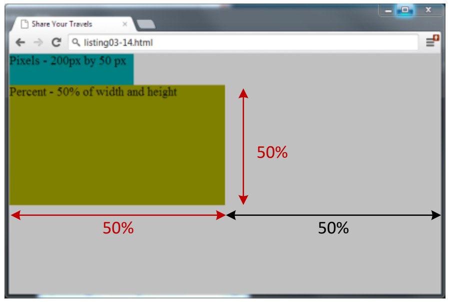
- When you use **percentages**, the size is **relative to** the size of the **parent** element.
- When you use **ems**, the size of the box is **relative to** the size of the **text** within it.

The rationale behind using these relative measures is to make one's design scalable to the size of the browser or device that is viewing it.

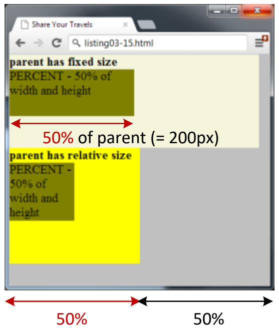
```
<style>
  html,body {
     margin:0;
     width:100%;
     height:100%;
     background: silver;
  }
  .pixels {
     width:200px;
     height:50px;
     background: teal;
  }
  .percent {
     width:50%;
     height:50%;
     background: olive;
  }
```



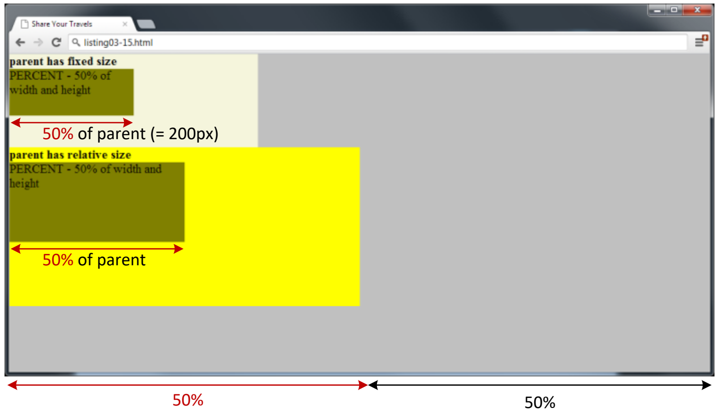
```
<body>
   <div class="pixels">
     Pixels - 200px by 50 px
   </div>
   <div class="percent">
     Percent - 50% of width and height
   </div>
</body>
```



```
  .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>
```



Developer Tools

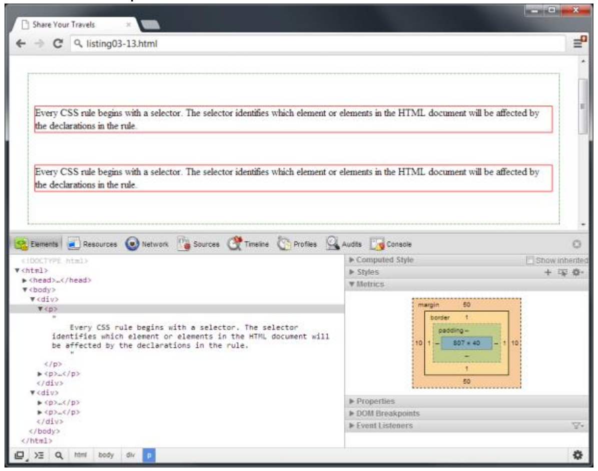
Help is on the way

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

You can use the various browsers’ CSS inspection tools to examine, for instance, the box values for a selected element.

Developer Tools

Chrome – Inspect Element

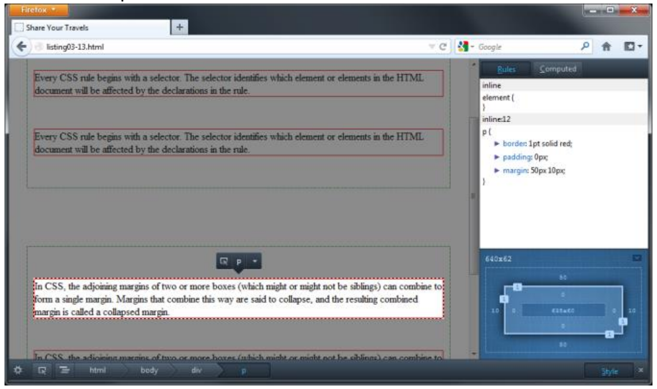


Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

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.

```
<!DOCTYPE html>
▼<html>
 ▶<head>…</head>
 ▼<body>
  ▼<div>
   ▼<p>
     "
       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.
     "
    </p>
   ▶<p>…</p>
   </div>
  ▼<div>
   ▶<p>…</p>
   ▶<p>…</p>
   </div>
  </body>
</html>
```

Firefox – Inspect



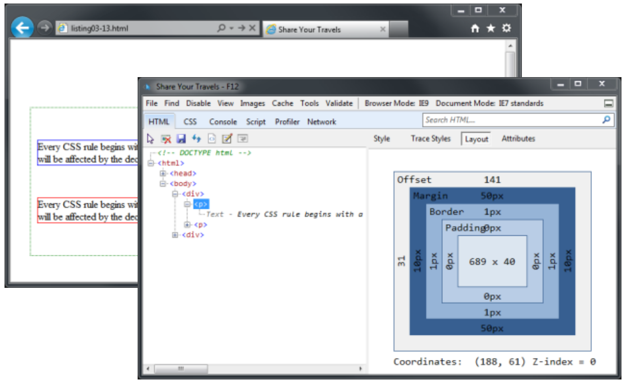
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

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.

```
inline
element {
}
inline:12
p {
  ▶ border: 1pt solid red;
  ▶ padding: 0px;
  ▶ margin: 50px 10px;
}
```

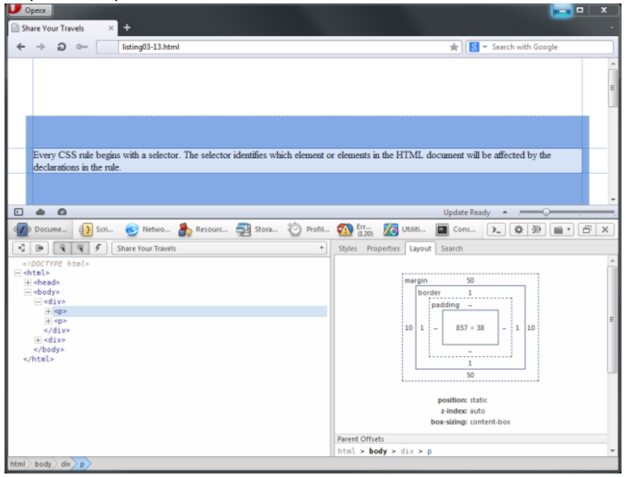
Internet Explorer – Developer Tools



```
<!-- DOCTYPE html -->
<html>
  <head>
  <body>
    <div>
      <p>
        Text - Every CSS rule begins with a
      <p>
    <div>
```

listing03-13.html
Share Your Travels
Share Your Travels - F12
File Find Disable View Images Cache Tools Validate Browser Mode: IE9 Document Mode: IE7 standards
HTML CSS Console Script Profiler Network
Search HTML...
Style Trace Styles Layout Attributes
Offset 141
Margin 50px
Border 1px
Padding 0px
689 x 40
31
Coordinates: (188, 61) Z-index = 0

Opera – Inspect Element



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.

```
<!DOCTYPE html>
<html>
  <head>
  <body>
    <div>
      <p>
      <p>
    </div>
    <div>
  </body>
</html>
```

CSS Selectors Level 3/4 – teaser

:has

A parent selector pseudo-class

It lets you change the parent element if it has a child or another element that follows it.

You can only style down, from parent to child, but never back up the tree. `:has` completely changes this because up until now there have been no parent selectors in CSS and there are some good reasons why.

Because of the way in which browsers parse HTML and CSS, selecting the parent if certain conditions are met could lead to all sorts of performance concerns.

It’s supported in some browser today.

:has

A parent selector pseudo-class

```
div:has(p) {

  background: red;

}

div:has(+ div) {

  color: blue;

}

a:has(> img) {

  border: 20px solid white;

}
```

:not

A negation selector pseudo-class

Selectors Level 3 only allowed `:not()` pseudo-class to accept a single simple selector, which the element must not match any of. Thus, `:not(a, .b, [c])` or `:not(a.b[c])` did not work.

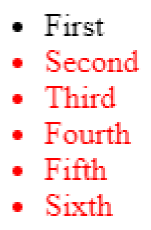
Selectors Level 4 allows `:not()` to accept a list of selectors. Thus, `:not(a):not(.b):not([c])` can instead be written as `:not(a, .b, [c])` and `:not(a.b[c])` works as intended.

It’s supported in major browser today.

:not

A negation selector pseudo-class

```
ul li:not(:first-of-type) {

  color: red;

}
```



- First
- Second
- Third
- Fourth
- Fifth
- Sixth

:is

A check selector pseudo-class

The `:is()` (formerly `:matches()`, formerly `:any()`) pseudo-class checks whether the element at its position in the outer selector matches any of the selectors in its selector list. It's useful syntactic sugar that allows you to avoid writing out all the combinations manually as separate selectors.

The effect is similar to nesting in Sass and most other CSS preprocessors.

It’s supported in major browser today.

:is

A check selector pseudo-class

```
:is(section, article, aside, nav) :is(h1, h2,
h3, h4, h5, h6) {

  color: #BADA55;

}




/* ... which would be the equivalent of: */

section h1, section h2, section h3, section h4,
section h5, section h6, article h1, article h2,
article h3, article h4, article h5, article h6,
aside h1, aside h2, aside h3, aside h4, aside
h5, aside h6, nav h1, nav h2, nav h3, nav h4,
nav h5, nav h6 {

  color: #BADA55;

}
```

CSS Variables

A strong point of CSS preprocessors is the possibility of using variables to create re-usable values and avoid code redundancy.

While tools like SASS are very useful for front-end web development, they aren’t required for using variables, as this can be done in native CSS.

Variables are declared by giving them a name preceded by two dashes. When wanting to use a previously created variable, use the `var()` function.

CSS Variables

```
:root {
  --main-bg-color: coral;
  --main-txt-color: #fff;
  --main-padding: 15px;
}

#div1 {
  background-color: var(--main-bg-color);
  color: var(--main-txt-color);
  padding: var(--main-padding);
}
```

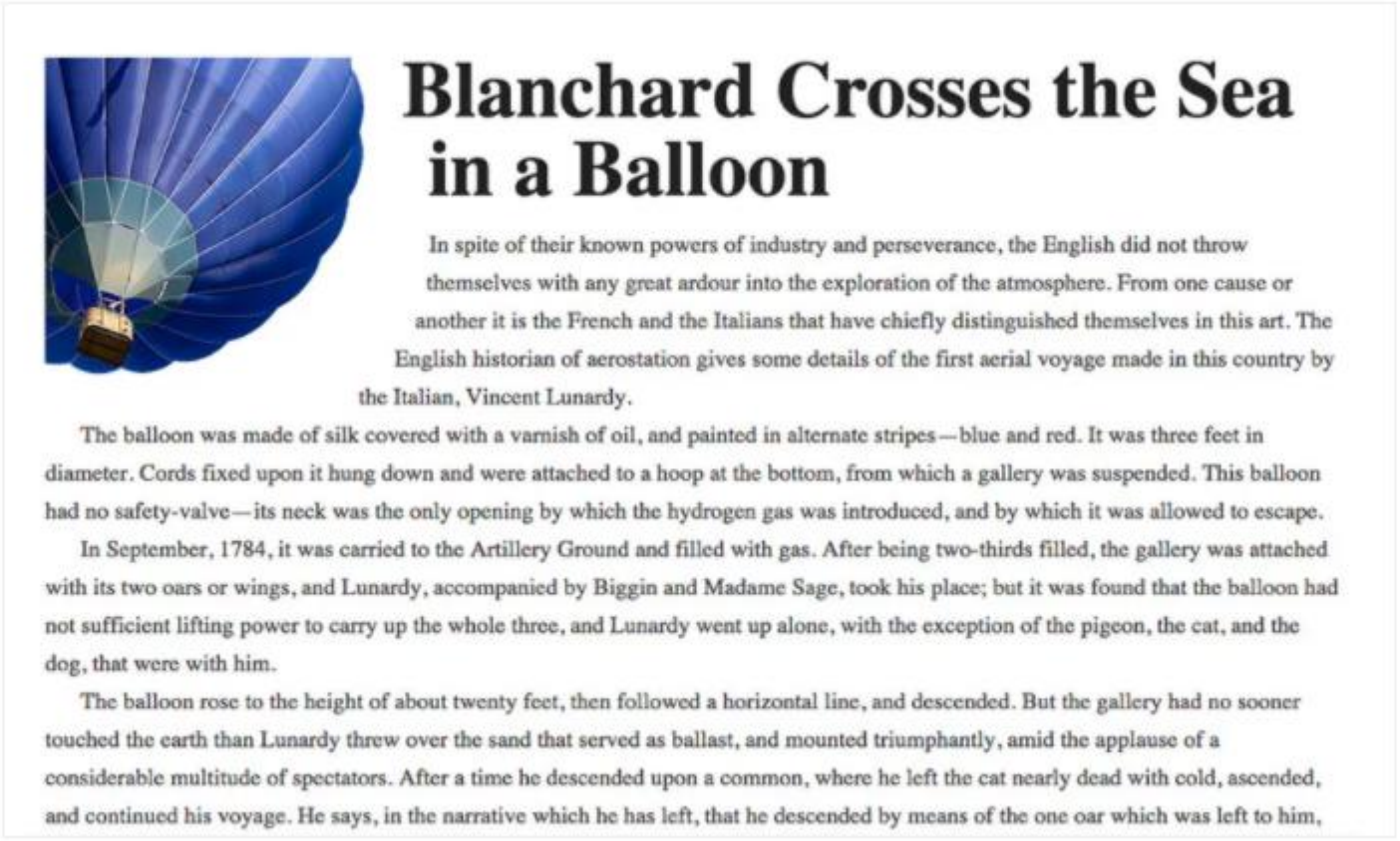
Curve Text Around a Floating Image

`shape-outside` is a CSS property that allows geometric shapes to be set, in order to define an area for text to flow around.

It’s supported in major browser today.

```
.shape {

  width: 300px;

  float: left;

  shape-outside: circle(50%);

}
```

Curve Text Around a Floating Image



Blanchard Crosses the Sea in a Balloon

In spite of their known powers of industry and perseverance, the English did not throw themselves with any great ardour into the exploration of the atmosphere. From one cause or another it is the French and the Italians that have chiefly distinguished themselves in this art. The English historian of aerostation gives some details of the first aerial voyage made in this country by the Italian, Vincent Lunardy.

The balloon was made of silk covered with a varnish of oil, and painted in alternate stripes—blue and red. It was three feet in diameter. Cords fixed upon it hung down and were attached to a hoop at the bottom, from which a gallery was suspended. This balloon had no safety-valve—its neck was the only opening by which the hydrogen gas was introduced, and by which it was allowed to escape.

In September, 1784, it was carried to the Artillery Ground and filled with gas. After being two-thirds filled, the gallery was attached with its two oars or wings, and Lunardy, accompanied by Biggin and Madame Sage, took his place; but it was found that the balloon had not sufficient lifting power to carry up the whole three, and Lunardy went up alone, with the exception of the pigeon, the cat, and the dog, that were with him.

The balloon rose to the height of about twenty feet, then followed a horizontal line, and descended. But the gallery had no sooner touched the earth than Lunardy threw over the sand that served as ballast, and mounted triumphantly, amid the applause of a considerable multitude of spectators. After a time he descended upon a common, where he left the cat nearly dead with cold, ascended, and continued his voyage. He says, in the narrative which he has left, that he descended by means of the one oar which was left to him,

Text Styling

Text Properties

Two basic types

CSS provides two types of properties that affect text.

- **font properties** that affect the font and its appearance.
- **paragraph properties** that affect the text in a similar way no matter which font is being used.

Font-Family

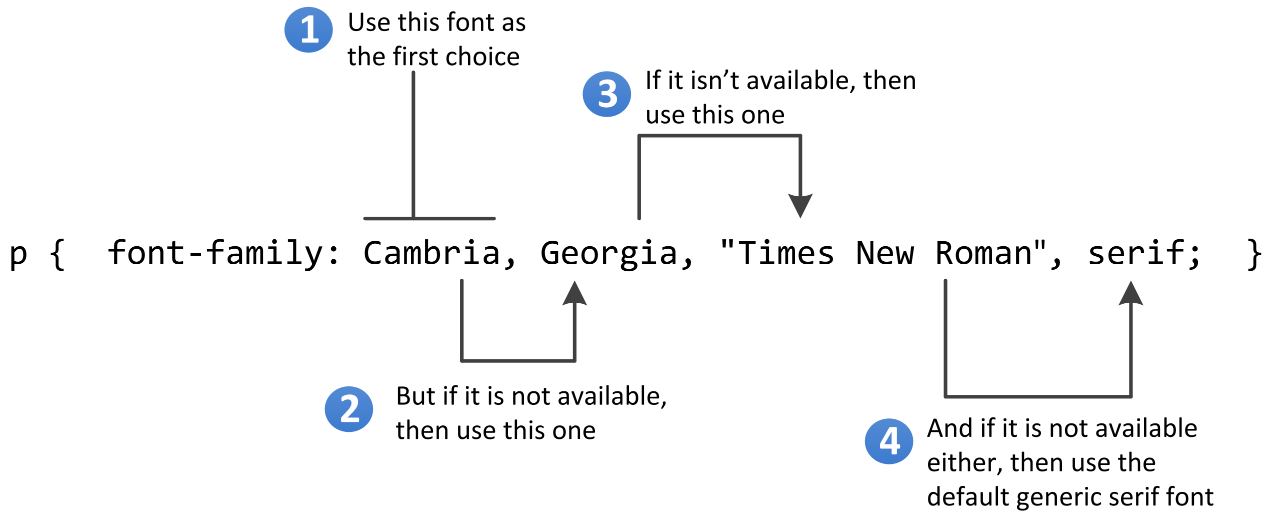
A few issues here

A word processor on a desktop machine can make use of any font that is installed on the computer; browsers are no different.

However, just because a given font is available on the web developer's computer, it does not mean that that same font will be available for all users who view the site.

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

Specifying the Font-Family



```
p {  font-family: Cambria, Georgia, "Times New Roman", serif;  }
```

1 Use this font as the first choice

2 But if it is not available, then use this one

3 If it isn’t available, then use this one

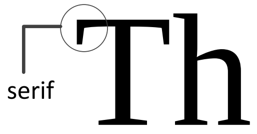
4 And if it is not available either, then use the default generic serif font

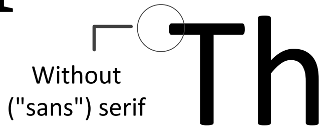
Generic Font-Family

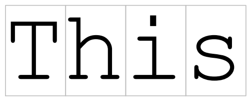
The font-family property supports five different generic families.

The browser supports a typeface from each family.

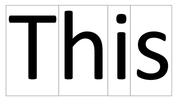
	Generic Font-Family Name
This	`serif`
This	`sans-serif`
This	`monospace`
This	`cursive`
This	`fantasy`







In a monospace font, each letter has the same width



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

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

@font-face

The future is now

Over the past few years, the most recent browser versions have begun to support the **@font-face** selector in CSS.

This selector allows you to use a font on your site even if it is not installed on the end user's computer.

Due to the on-going popularity of open source font sites such as Google Web Fonts (http://www.google.com/webfonts) and Font Squirrel (http://www.fontsquirrel.com/), @font-face seems to have gained a critical mass of widespread usage.

Font Sizes

Mo control, mo problems

The issue of font sizes is unfortunately somewhat tricky.

In a print-based program such as a word processor, specifying a font size in points is unproblematic.

However, absolute units such as points and inches do not translate very well to pixel-based devices.

Somewhat surprisingly, pixels are also a problematic unit.

> Newer mobile devices in recent years have been increasing pixel densities so that a given CSS pixel does not correlate to a single device pixel.

Font Sizes

Welcome ems and percents again

If we wish to create web layouts that work well on different devices, we should learn to use relative units such as **em** units or **percentages** for our font sizes (and indeed for other sizes in CSS as well).

One of the principles of the web is that the user should be able to change the size of the text if he or she so wishes to do so.

Using percentages or em units ensures that this user action will work.

How to use ems and percents

When used to specify a font size, both em units and percentages are relative to the parent’s font size.

How to use ems and percents

`<body>`	Browser’s default text size is usually 16 pixels
`<p>`	100% or 1em is 16 pixels
`<h3>`	125% or 1.125em is 18 pixels
`<h2>`	150% or 1.5em is 24 pixels
`<h1>`	200% or 2em is 32 pixels

```
/* using 16px scale */

body { font-size: 100%; }
h3 { font-size: 1.125em; }  /* 1.25 x 16 = 18 */
h2 { font-size: 1.5em; }    /* 1.5 x 16  = 24 */
h1 { font-size: 2em; }      /* 2 x 16 = 32 */
```

```
<body>
  <p>this will be about 16 pixels</p>
  <h1>this will be about 32 pixels</h1>
  <h2>this will be about 24 pixels</h2>
  <h3>this will be about 18 pixels</h3>
  <p>this will be about 16 pixels</p>
</body>
```

How to use ems and percents

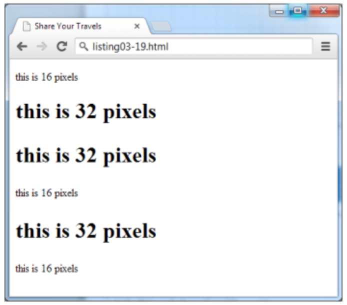
It might seem easy … but it’s not …

While this looks pretty easy to master, things unfortunately can quickly become quite complicated.

Remember that percents and em units are relative to their parents, so if the parent font size changes, this affects all of its contents.

ems and percents

```
<body>
  <p>this is 16 pixels</p>
  <h1>this is 32 pixels</h1>
  <article>
     <h1>this is 32 pixels</h1>
     <p>this is 16 pixels</p>
     <div>
        <h1>this is 32 pixels</h1>
        <p>this is 16 pixels</p>
     </div>
  </article>
</body>
```



this is 16 pixels

this is 32 pixels

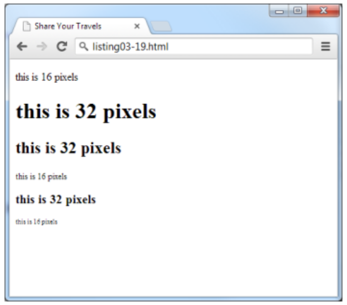
this is 32 pixels

this is 16 pixels

this is 32 pixels

this is 16 pixels

```
/* using 16px scale */

body { font-size: 100%; }
p    { font-size: 1em; }      /* 1 x 16 = 16px */
h1   { font-size: 2em; }      /* 2 x 16 = 32px */
```



this is 16 pixels

this is 32 pixels

this is 32 pixels

this is 16 pixels

this is 32 pixels

this is 16 pixels

```
/* using 16px scale */

body { font-size: 100%; }
p    { font-size: 1em; }
h1   { font-size: 2em; }

article { font-size: 75% }   /* h1 = 2 * 16 * 0.75 = 24px
                                 p = 1 * 16 * 0.75 = 12px  */

div  { font-size: 75% }      /* h1 = 2 * 16 * 0.75 * 0.75 = 18px
                                 p = 1 * 16 * 0.75 * 0.75 = 9px   */
```

The rem unit

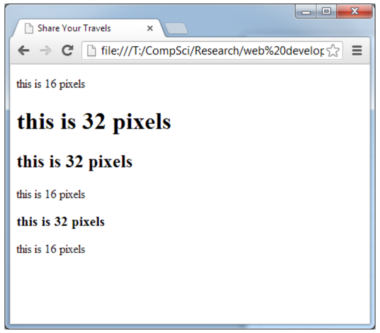
Solution to font sizing hassles?

CSS3 now supports a new relative measure, the **rem** (for root em unit).

This unit is always relative to the size of the root element (i.e., the <html> element).

However, since Internet Explorer prior to version 9 do not support the rem units, you need to provide some type of fallback for those browsers.

The rem unit



this is 16 pixels

this is 32 pixels

this is 32 pixels

this is 16 pixels

this is 32 pixels

this is 16 pixels

```
/* using 16px scale */

body { font-size: 100%; }
p {
       font-size: 16px;  /* for older browsers: won’t scale properly though */
       font-size: 1rem;  /* for new browsers: scales and simple too */
}
h1   { font-size: 2em; }

article { font-size: 75% }  /* h1 = 2 * 16 * 0.75 = 24px
                               p = 1 * 16 = 16px   */

div  { font-size: 75% }      /* h1 = 2 * 16 * 0.75 * 0.75 = 18px
                                p = 1 * 16 = 16px    */
```