# 3. Introduction to CSS

2023학년 2학기 웹응용프로그래밍

권 동 현

CSLAB

## Contents

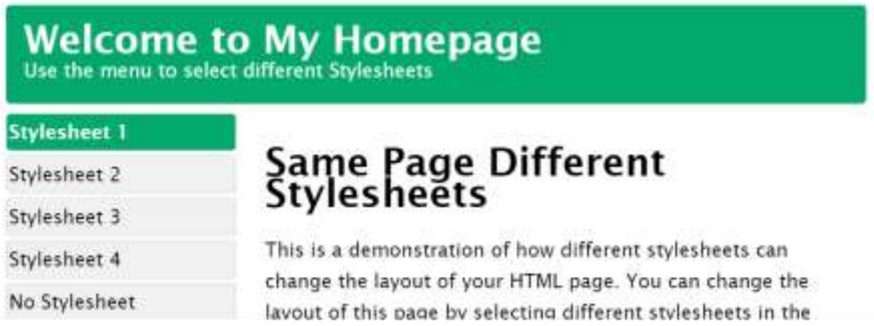
- What is CSS?
- CSS Syntax
- Selectors
- Location of Styles
- The Cascade: How Styles Interact
- The Box Model
- CSS Text Styling
- Layout: Normal Flow
- Layout: Positioning Elements
- Layout: Floating Elements
- Constructing Multicolumn Layouts

## 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.
- 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.

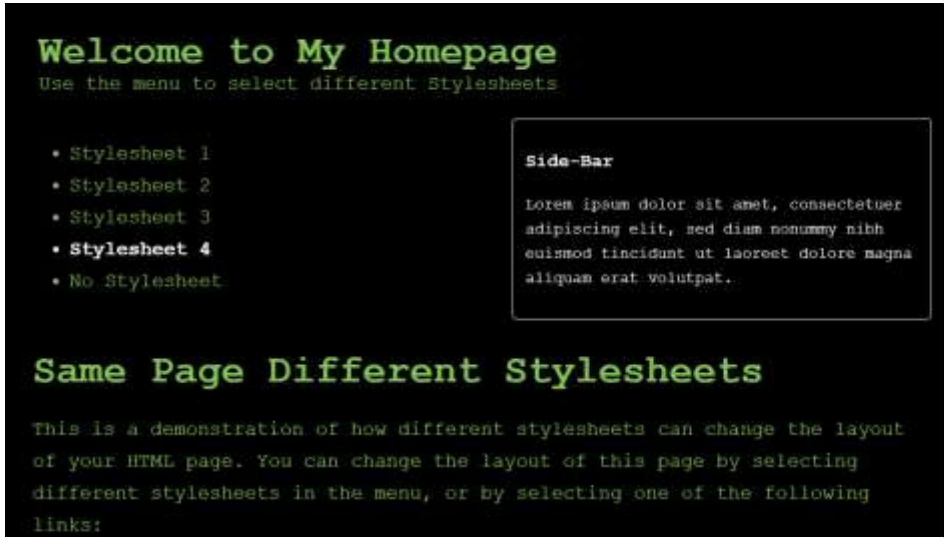
## Benefits of CSS

- 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.
  - https://www.w3schools.com/css/css_intro.asp



Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

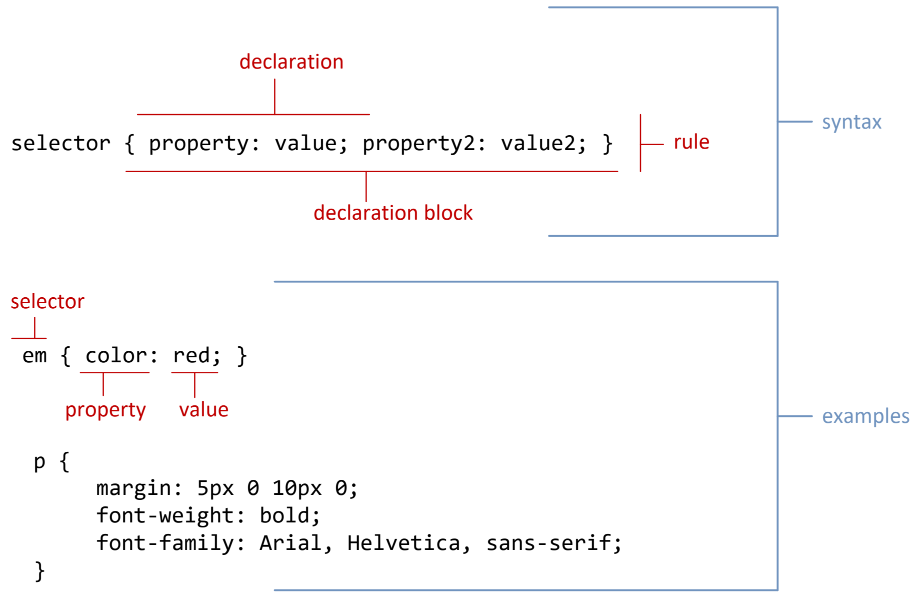
Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links:

# CSS Syntax

# CSS Syntax

- A CSS document consists of one or more **style rules**.
- A rule consists of a selector that identifies the HTML element or elements that will be affected, followed by a series of **property** and **value** pairs (each pair is also called a **declaration**).
- The series of declarations is also called the **declaration block**.
  - A declaration block can be together on a single line, or spread across multiple lines.
  - The browser ignores white space
  - Each declaration is terminated with a semicolon.



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

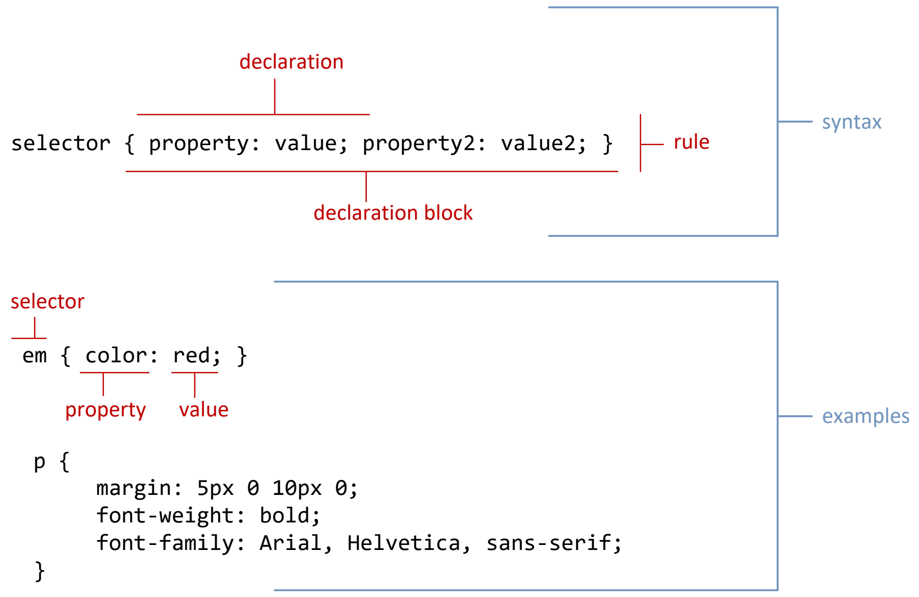
```
em { color: red; }
```

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

# CSS Syntax

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

- 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

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

# Properties

| Property Type | Property |
|---|---|
| **Spacing** | padding<br>padding-bottom, padding-left, padding-right, padding-top<br>margin<br>margin-bottom, margin-left, margin-right, margin-top |
| **Sizing** | height<br>max-height<br>max-width<br>min-height<br>min-width<br>width |
| **Layout** | bottom, left, right, top<br>clear<br>display<br>float<br>overflow<br>position<br>visibility<br>z-index |
| **Lists** | list-style<br>list-style-image<br>list-style-type |

## Values

- 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.

- 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.

# 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;<br>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);<br>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;<br>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%);<br>color: hsl(330,59%,100%); |

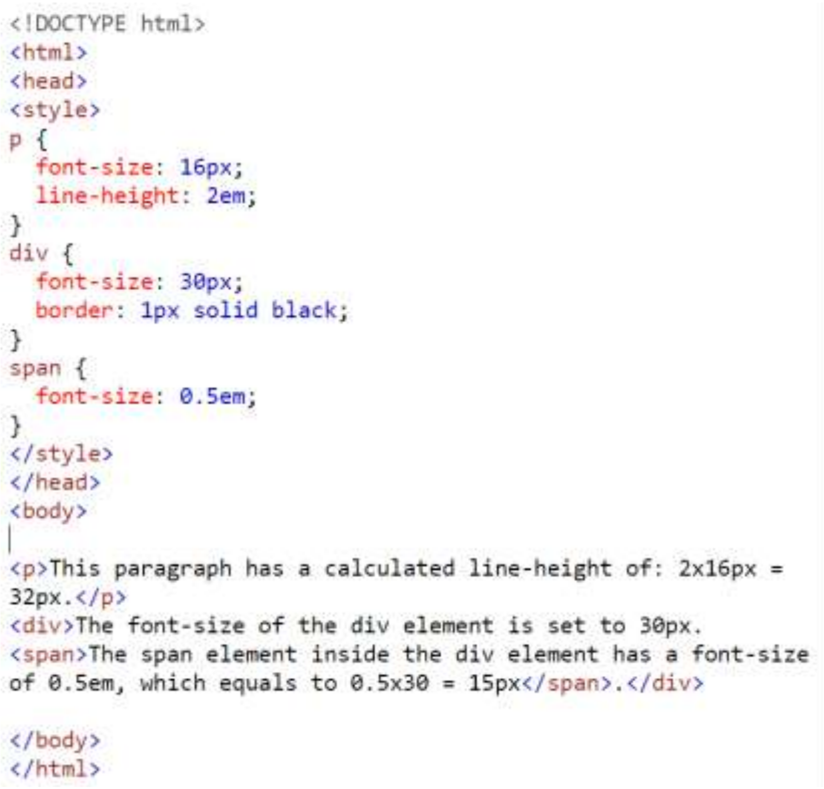
# Absolute Units

| Unit | Description | Type |
|---|---|---|
| **in** | Inches (1in = 96px = 2.54cm) | 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 |
| **px** | Pixesl (1px = 1/96 of an inch) | Absolute* |

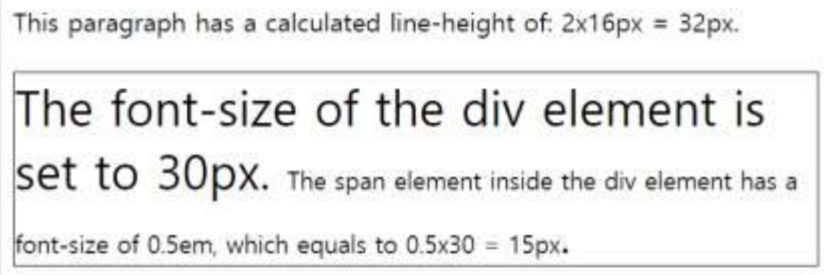
## Relative Units

| Unit | Description | Type |
|---|---|---|
| **em** | Relative to the font-size of the element (2em means 2 times the size of the current font) | Relative |
| **%** | Relative to the parent element | Relative |
| **ex** | Relative to the x-height of the current font (rarely used) | Relative |
| **ch** | Relative to the width of the "0" (zero) | Relative<br><br>(CSS3 only) |
| **rem** | Relative to font-size of the root element | Relative<br><br>(CSS3 only) |
| **vw, vh** | Relative to 1% of the width of the viewport*<br><br>Relative to 1% of the height of the viewport*<br><br>* Viewport = the browser window size. If the viewport is 50cm wide, 1vw = 0.5cm. | Relative<br><br>(CSS3 only) |

## 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>This paragraph has 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>
```



This paragraph has 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.

참조: https://www.w3schools.com/cssref/css_units.asp

## Comments in CSS

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

***/* comment goes here */***

## Selectors

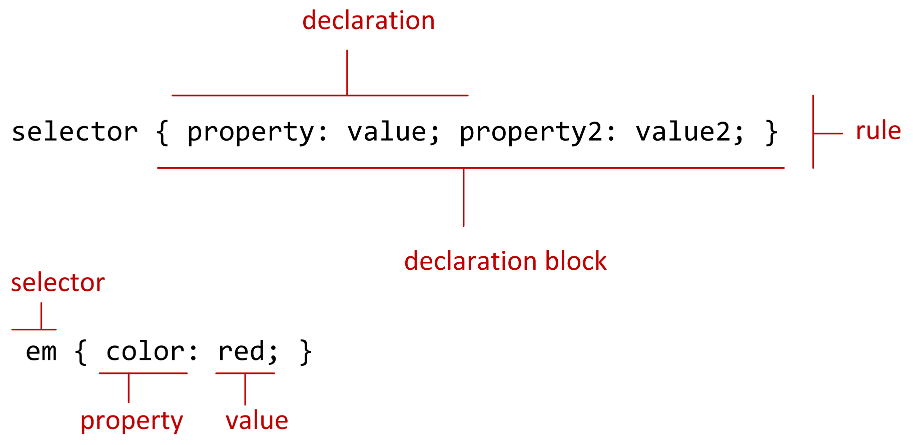
CSLAB

## Selectors

- 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**.



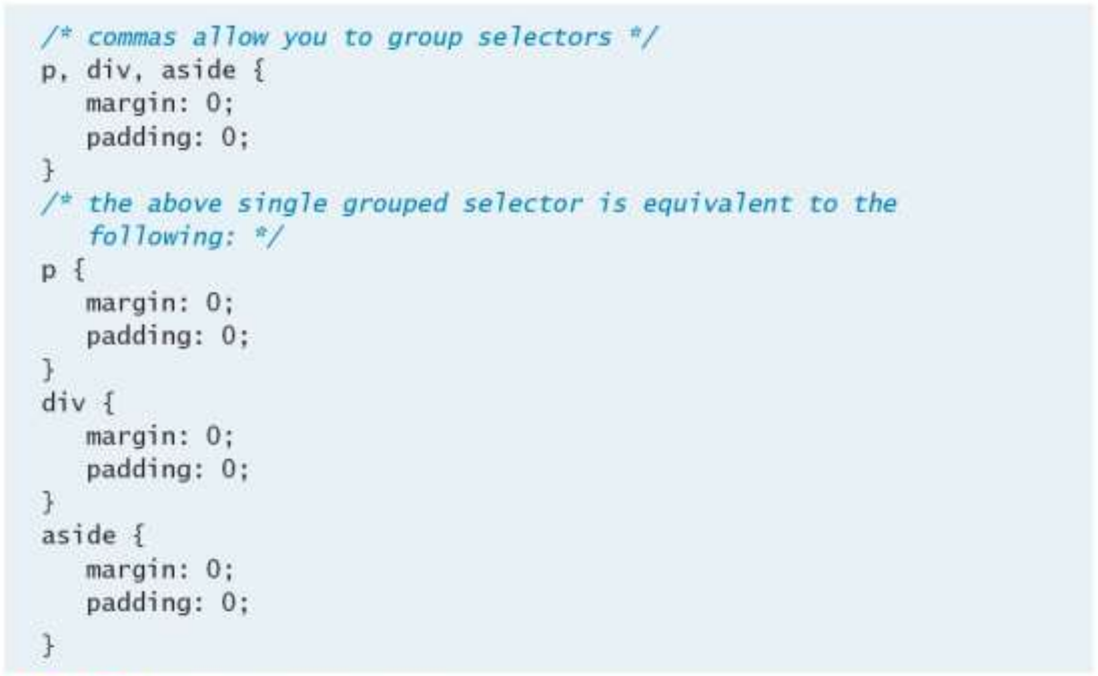
```
selector { property: value; property2: value2; }
```

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

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

## Grouped Selectors

- 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.

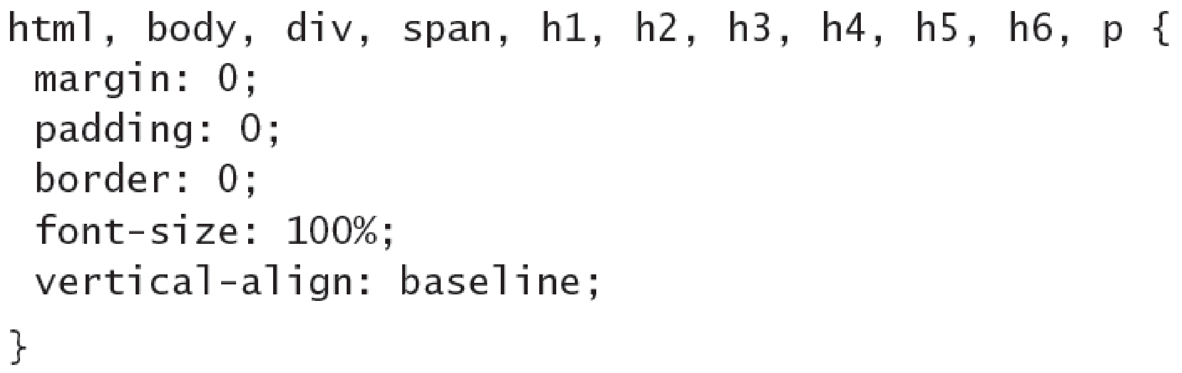


```
/* commas allow you to group selectors */
p, div, aside {
   margin: 0;
   padding: 0;
}
/* the above single grouped selector is equivalent to the
   following: */
p {
   margin: 0;
   padding: 0;
}
div {
   margin: 0;
   padding: 0;
}
aside {
   margin: 0;
   padding: 0;
}
```

LISTING 3.4 Sample grouped selector

## 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.
- 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.

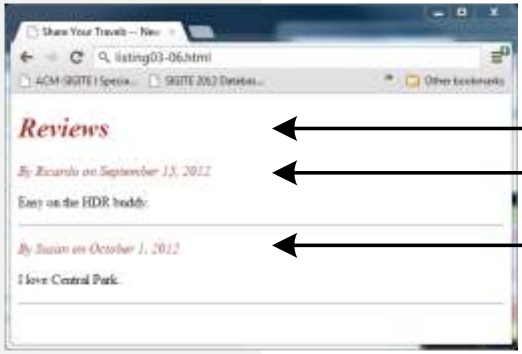


```
html, body, div, span, h1, h2, h3, h4, h5, h6, p {
 margin: 0;
 padding: 0;
 border: 0;
 font-size: 100%;
 vertical-align: baseline;
}
```

# Class Selectors

- 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.

```
<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*

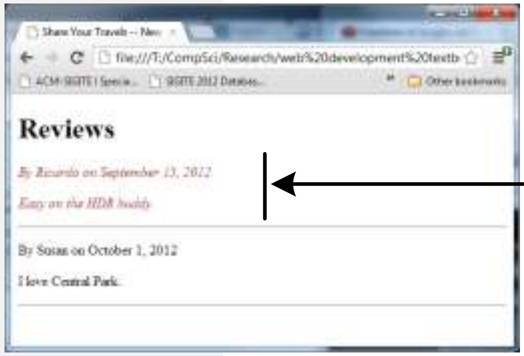
I love Central Park.

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

# Id Selectors

- 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

```
<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

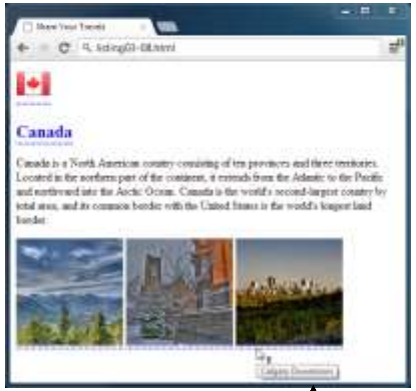
- Id selectors should only be used when **referencing a single HTML element** since an id attribute can only be assigned to a single HTML element.
- Class selectors should be used when (potentially) **referencing several related elements**.

## Attribute Selectors

- 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.

```
<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;
}
```



## Pseudo Selectors

- 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

| Selector | Type | Description |
|---|---|---|
| a:link | Pseudo-class | Selects links that have not been visited |
| a:visited | Pseudo-class | Selects links that have been visited |
| :focus | Pseudo-class | Selects elements that have the input focus |
| :hover | Pseudo-class | Selects elements that the mouse pointer is currently above |
| :active | Pseudo-class | Selects an element that is being activated by the user (e.g., a link that is being clicked) |
| :checked | Pseudo-class | Selects a form element that is currently checked. (e.g., radio button or check box) |
| :first-child | Pseudo-class | Selects an element that is the first child of its parent |
| :first-letter | Pseudo-element | Selects the first letter of an element |
| :first-line | Pseudo-elemetn | Selects the first line of an element |

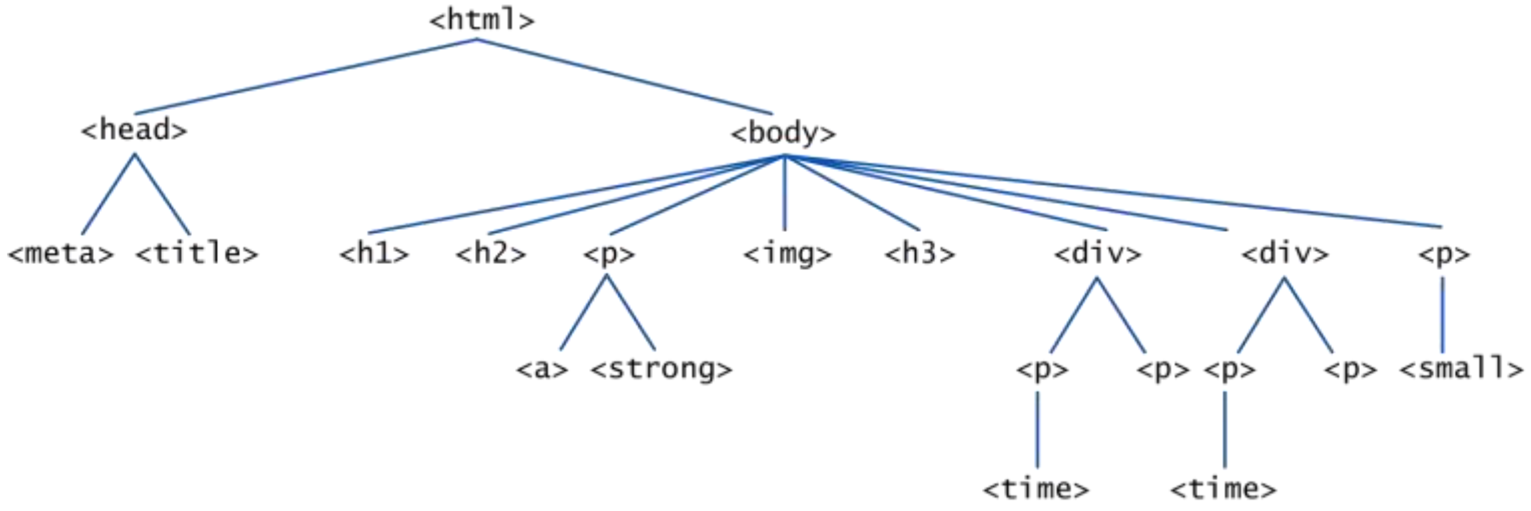
# Pseudo Selectors

```
<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

- 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.

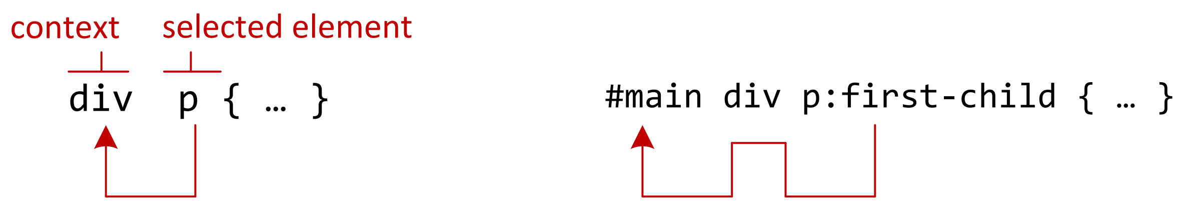


## Contextual Selectors

| Selector | Matches | Example |
|---|---|---|
| **Descendant** | A specified element that is contained somewhere within another specified element | div p<br><br>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<br><br>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<br><br>Selects the first <p> after any <h3>. |
| **General Sibling** | A specified element that shares the same parent as the specified element. | h3~p<br><br>Selects all the <p> elements that share the same parent as the <h3>. |

## Descendant Selector

- 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.



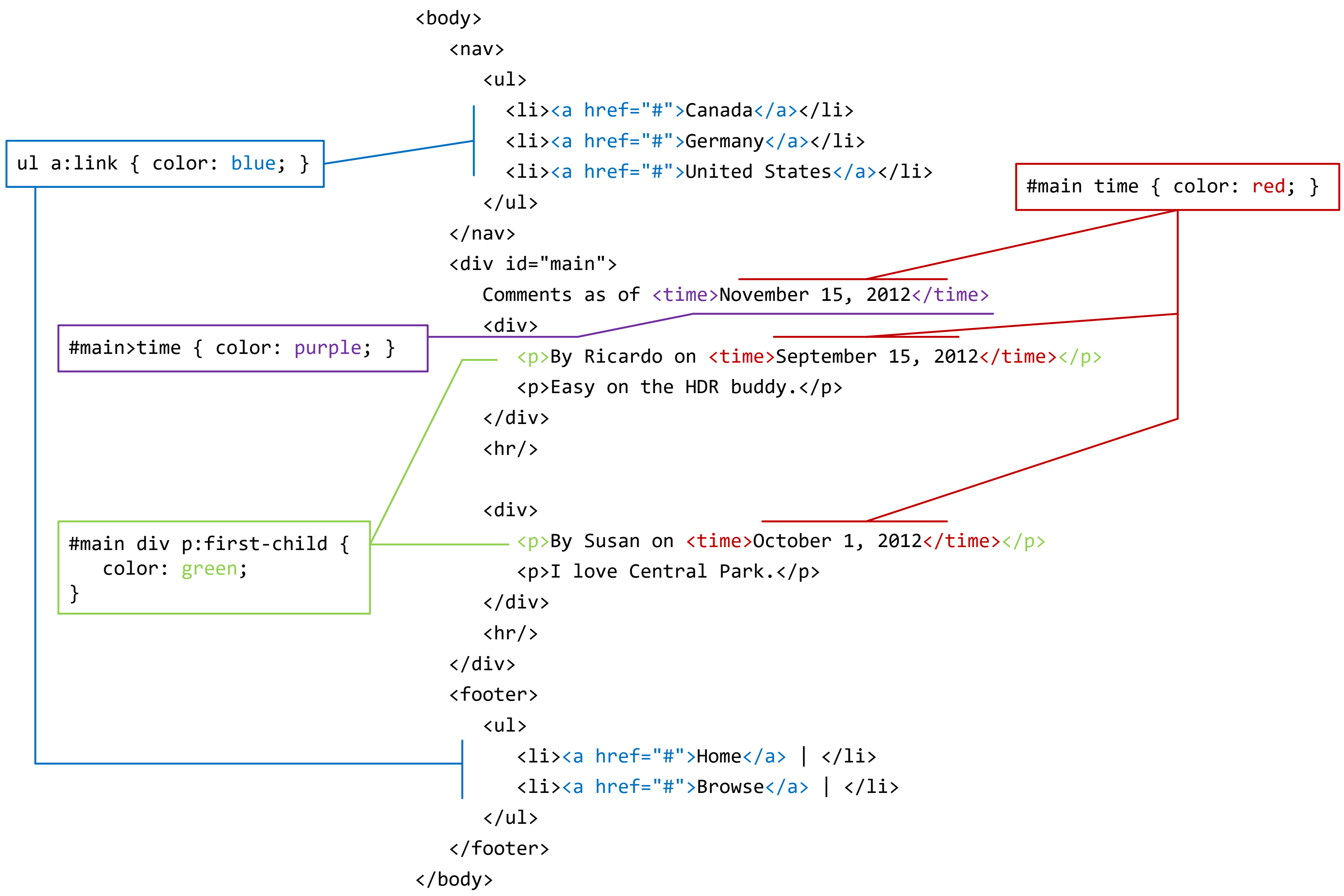
```
div  p { … }
```

```
#main div p:first-child { … }
```

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

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

# Contextual Selectors in Action



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

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

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

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

```
<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>
```

## Location of Styles

## Actually there are three …

- **Author-created style sheets** (what we are learning in this presentation).
  - three different locations.
    - **Inline**
    - **Embedded**
    - **External**
  - You can combine all 3!

- **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.

- (NOTE) Order: Browser < User < Author

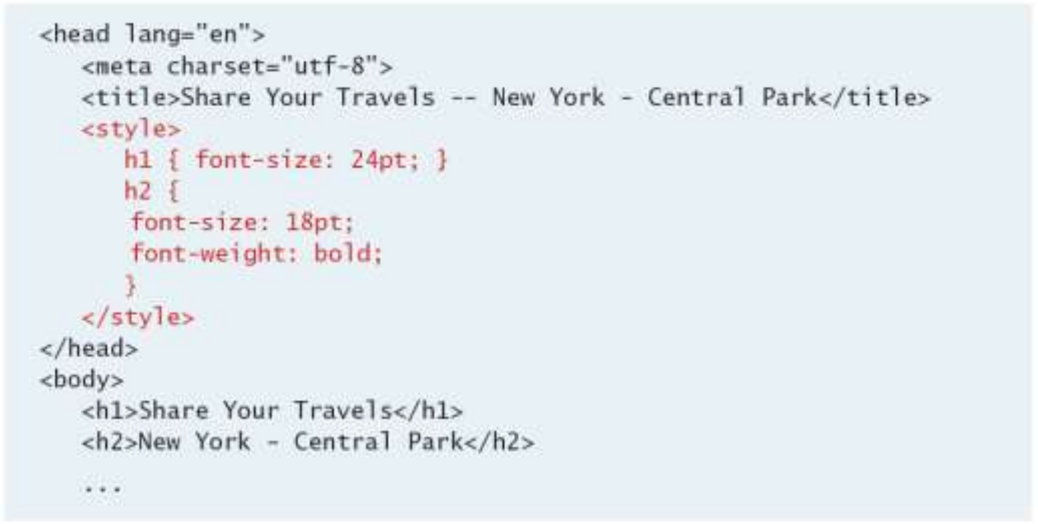
## Inline Styles

- 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.

```html
<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>
```

## Embedded Style Sheet

- 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.



```
<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

## 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

```
<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

## The Cascade: How Styles Interact

## Why Conflict Happens

- 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.

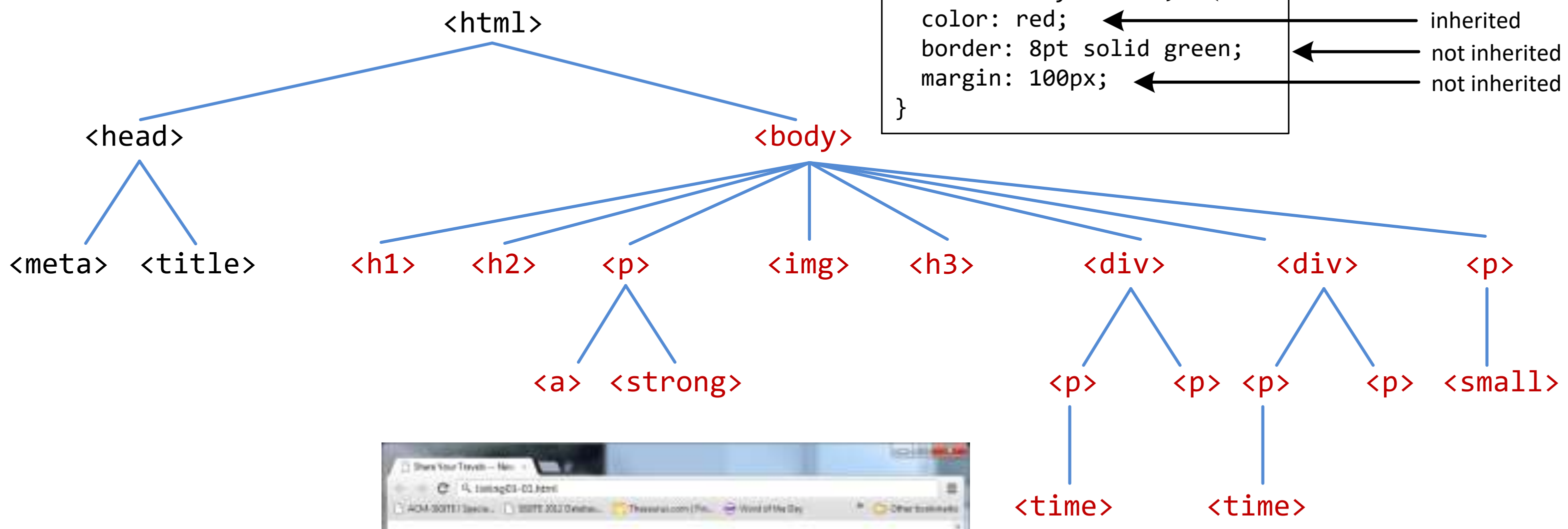
## Cascade

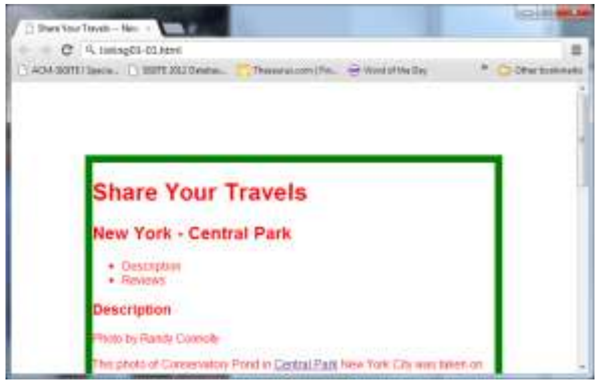
- The “Cascade” in CSS refers to how conflicting rules are handled.
- The visual metaphor behind the term **cascade** is that of a mountain stream progressing downstream over rocks.
- The downward movement of water down a cascade is meant to be analogous to how a given style rule will continue to take precedence with child elements.
- CSS uses the following **cascade principles** to help it deal with conflicts:
  - **inheritance, specificity, location**

# Inheritance

- 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.

```
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

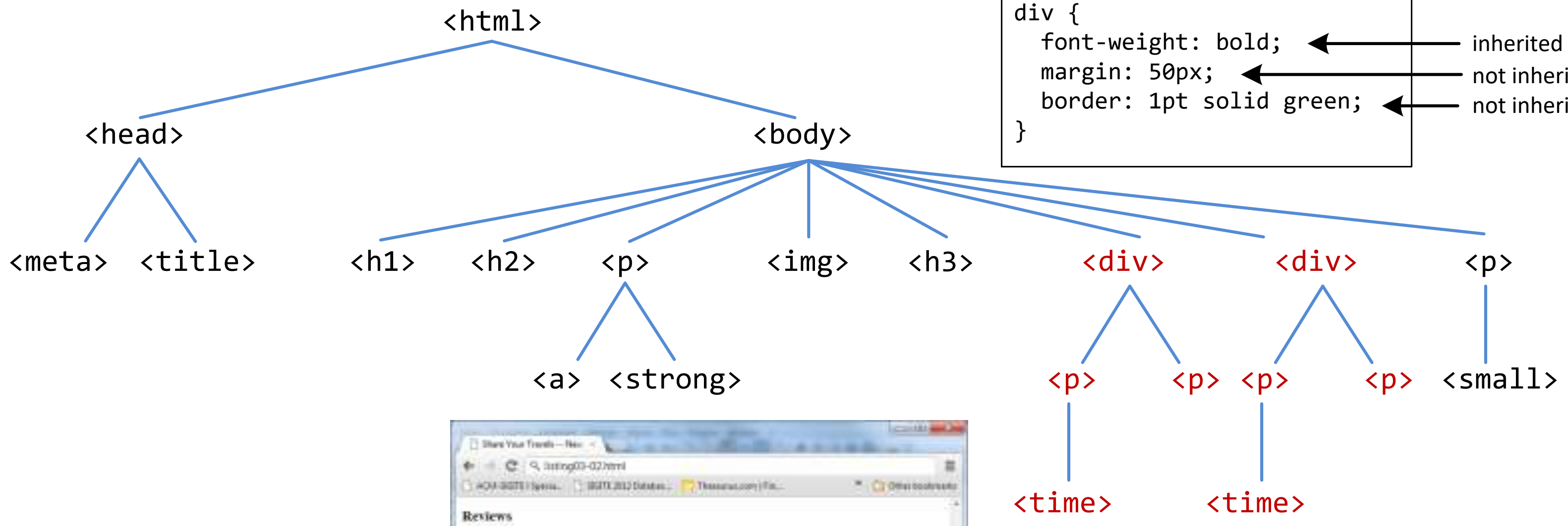
- Description
- Reviews

Description

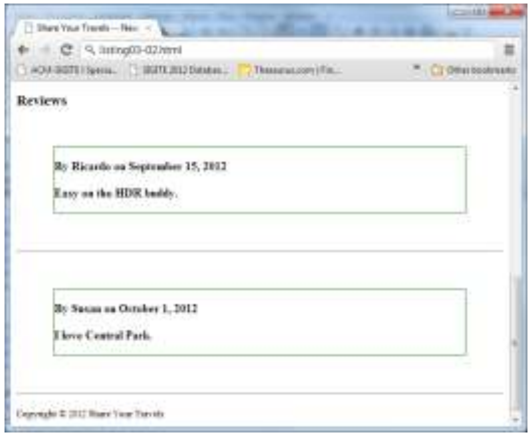
Photo by Randy Connolly

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

# Inheritance



```
div {
  font-weight: bold;        ← inherited
  margin: 50px;             ← not inherited
  border: 1pt solid green;  ← not inherited
}
```



Reviews

By Ricardo on September 15, 2012

Easy on the HDR buddy.

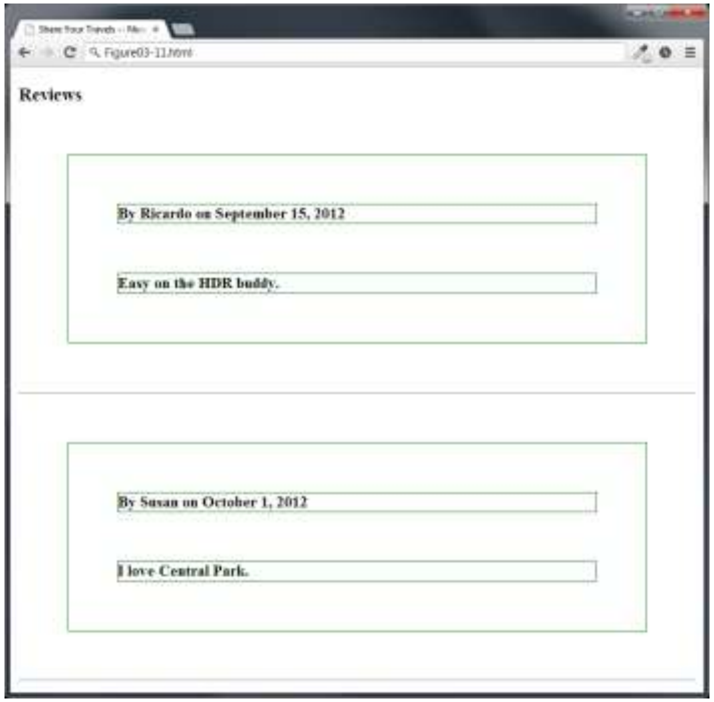
By Susan on October 1, 2012

I love Central Park.

Copyright © 2012 Share Your Travels

## 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/>
```

## Specificity

- **Specificity** is how the browser determines which style rule takes precedence when more than one style rule could be applied to the same element.
- The more **specific** the selector, the more it takes precedence (i.e., overrides the previous definition).
- The way that specificity works in the browser is that the browser assigns a weight to each style rule.
- When several rules apply, the one with **the greatest weight takes precedence**.

# Specificity

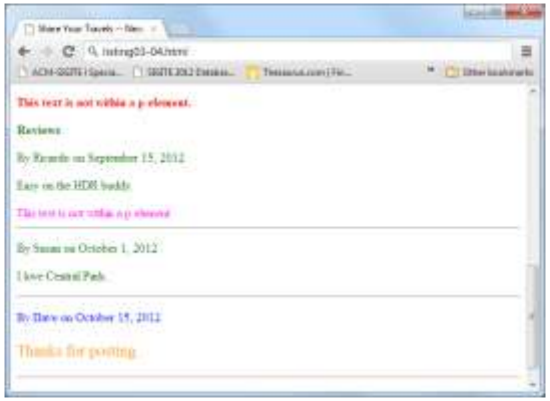
These color and font-weight properties are inheritable and thus potentially applicable to all the child elements contained within the body.
However, because the <div> and <p> elements also have the same properties set, they *override* the value defined for the <body> element because their selectors (div 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.

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

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

p {
  color: green;
}

.last {
  color: blue;
}

#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

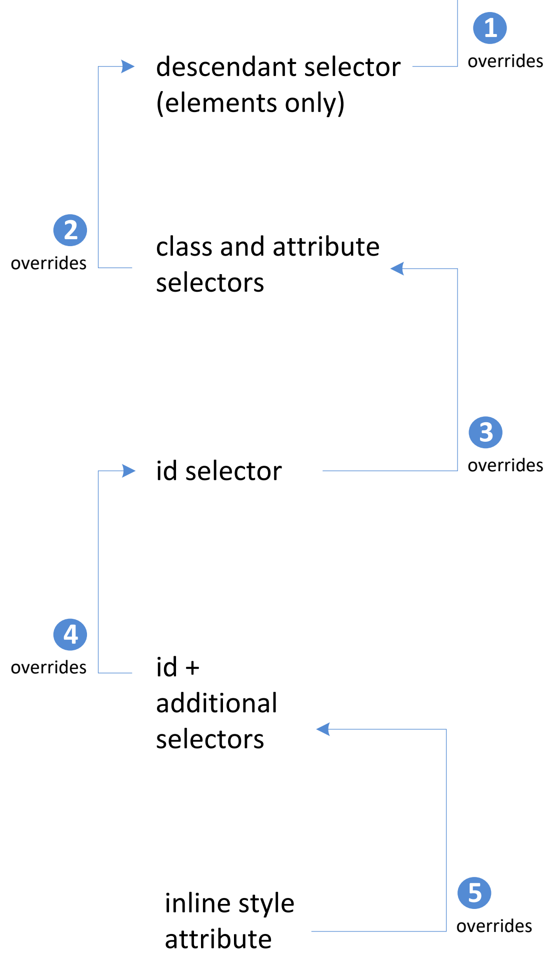
By Dave on October 15, 2012

Thanks for posting

## Specificity Algorithm

- 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;">
```

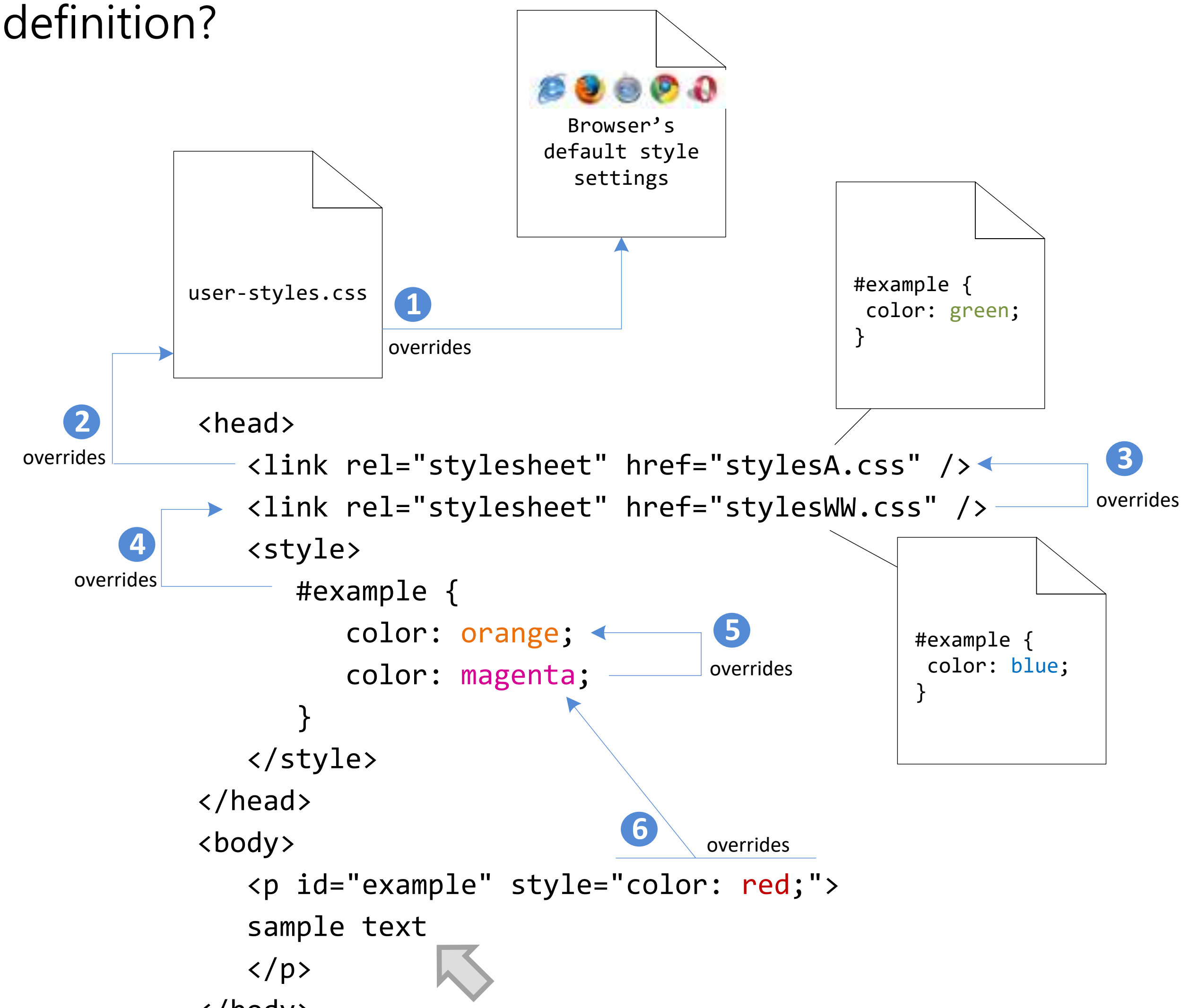
1000

## Location

- When inheritance and specificity cannot determine style precedence, the principle of **location** will be used.
- The principle of location is that when rules have the same specificity, then the latest are given more weight.

# Location

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



```
<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>
```

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

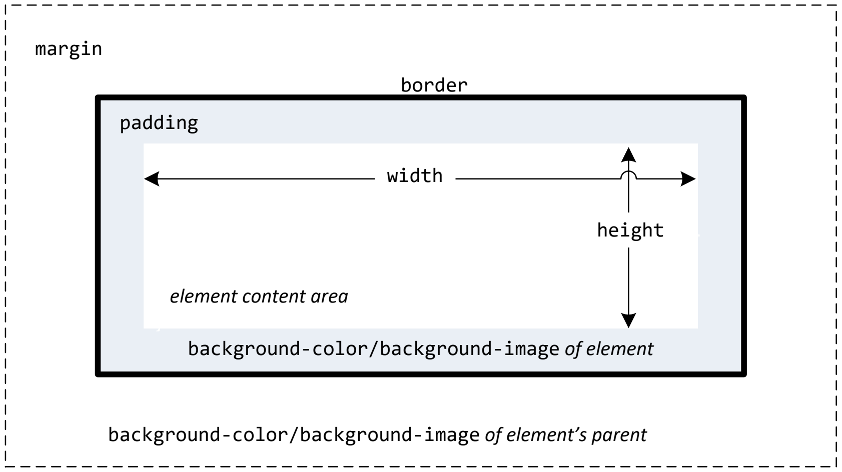
```
#example {
 color: blue;
}
```

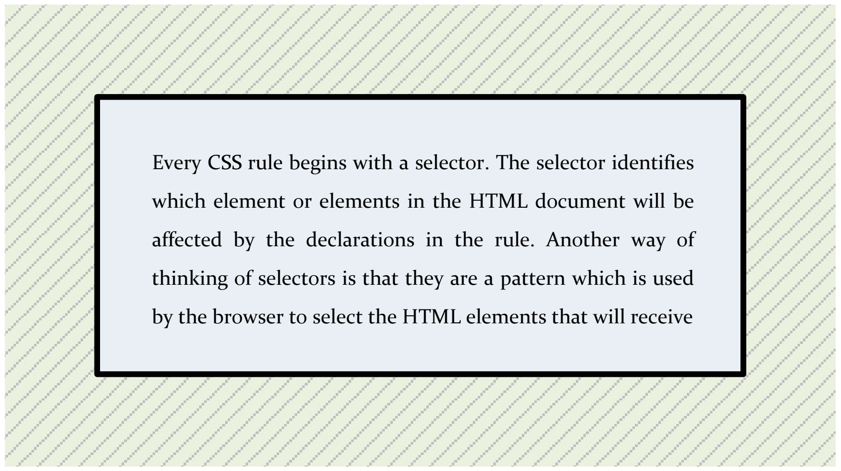
Can you guess what will be the color of the sample text ?

## The Box Model

## The Box Model

- 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 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 to use CSS to display purely presentational images (such as background gradients and patterns, decorative images, etc) rather than using the <img> element.


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

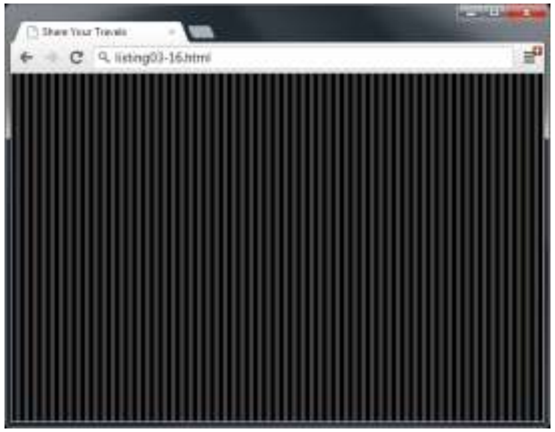


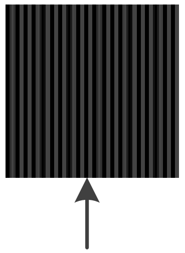
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

## Background Properties

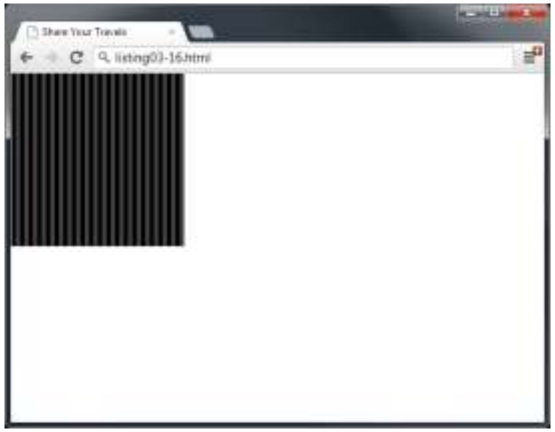
| Property | Description |
|---|---|
| **background** | A combined short-hand property that allows you to set the background values in one property. While you can omit properties with the short-hand, do remember that any omitted properties will be set to their default value.<br><br>Order: color image repeat attachment position |
| **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

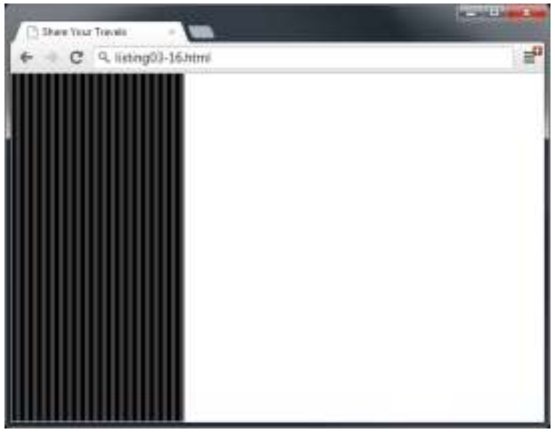




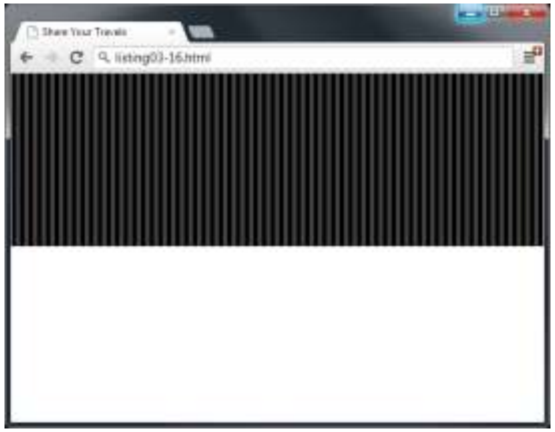
```
background-image: url(../images/backgrounds/body-background-tile.gif);
background-repeat: repeat;
```



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

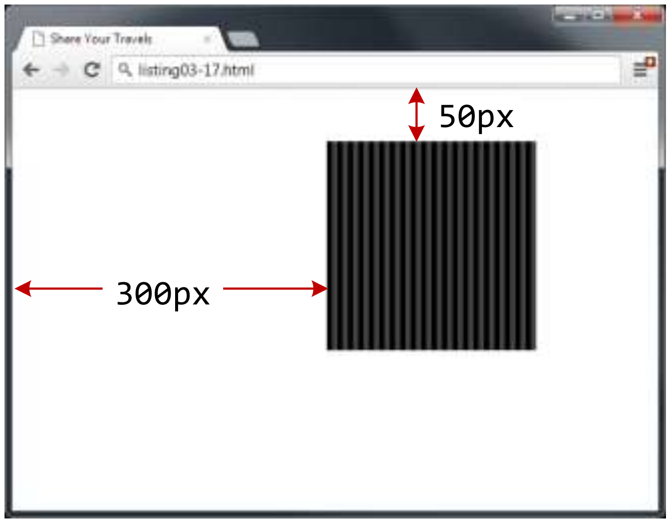


```
background-repeat: repeat-y;
```



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

# Background Position



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

## Borders

- 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.

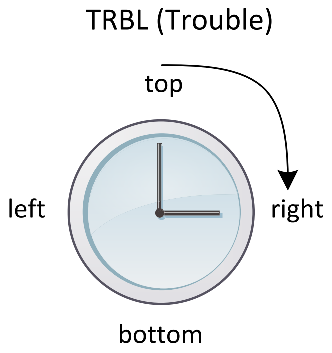
| Property | Description |
|---|---|
| **border** | A combined short-hand property that allows you to set the style, width, and color of a border in one property. The order is important and must be:<br><br>border-style border-width border-color |
| **border-style** | Specifies the line type of the border. 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

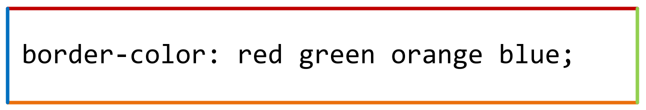
- With border, margin, and padding properties, there are long-form and shortcut methods to set the 4 sides

```
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.**



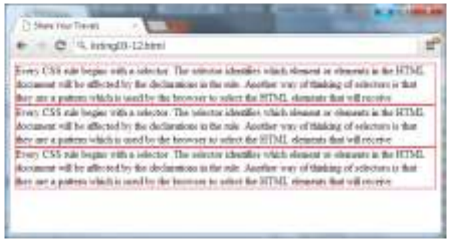
```
border-color: top right bottom left;
```



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

# Margins and Padding

- Box Model Properties #3 and #4



Share Your Travels

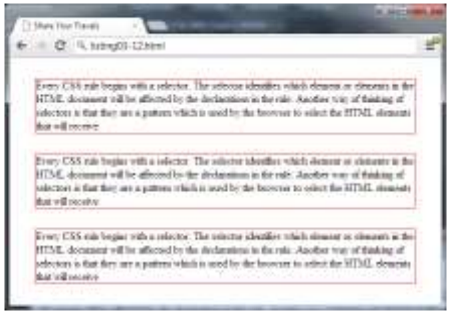
listing03-12.html

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

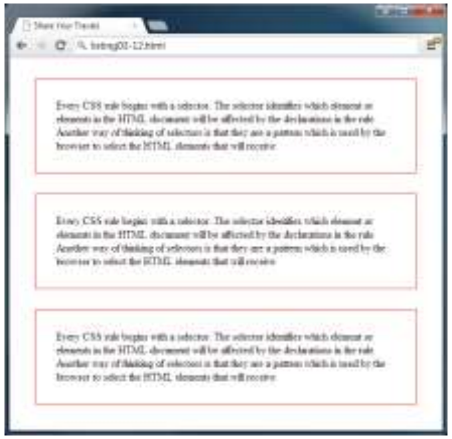
listing03-12.html

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

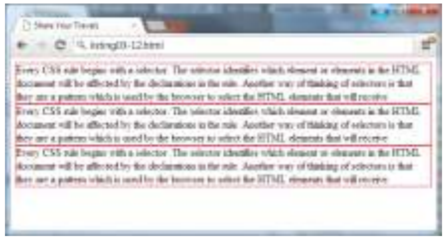
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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;
}
```

# Margins

- 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

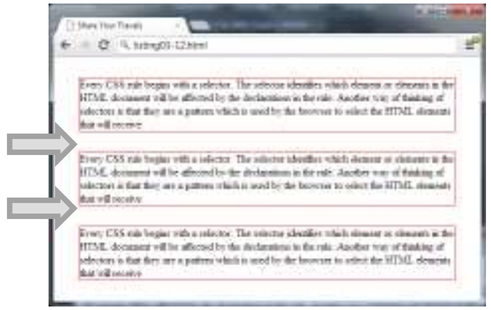
listing03-12.html

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

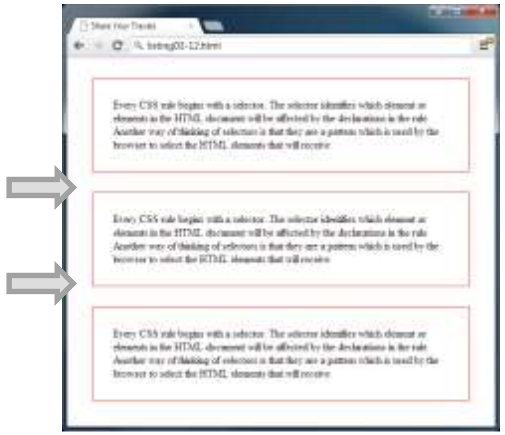
listing03-12.html

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

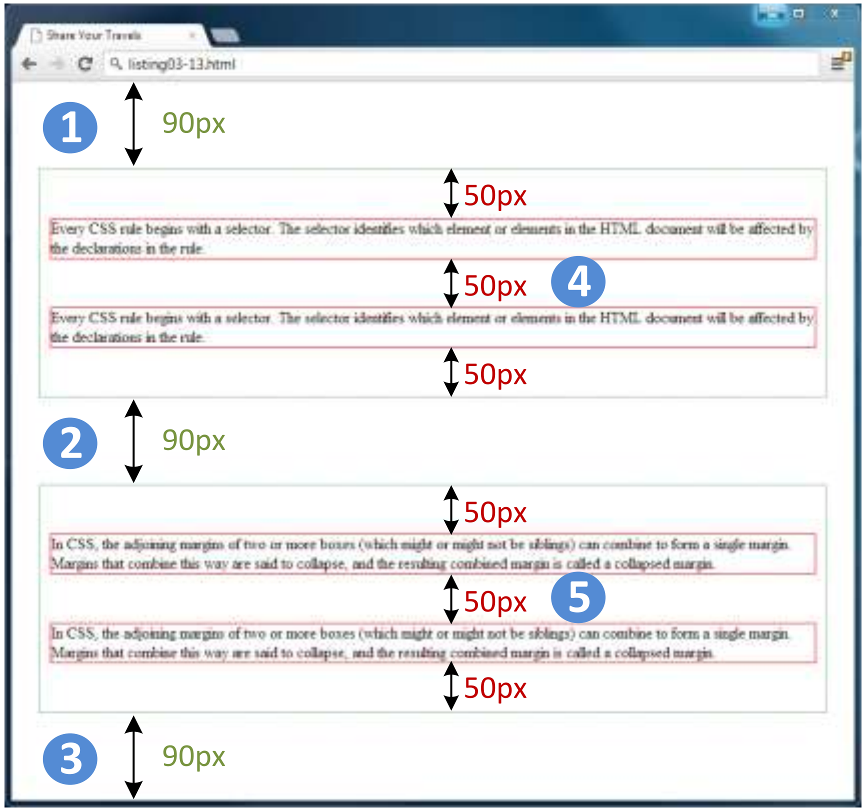
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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 2 would be 180p (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

- 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.

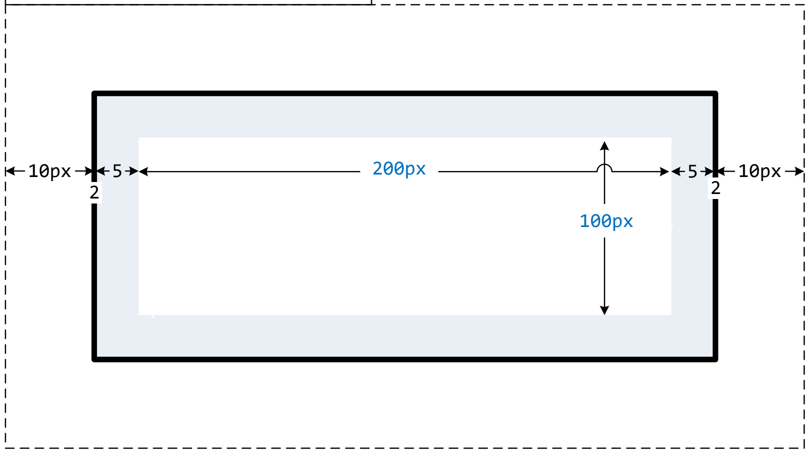
## Width and Height

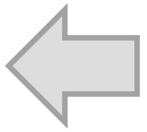
- The width and height properties specify the size of the element’s content area.
- Perhaps the only rival for collapsing margins in troubling our students, box dimensions have a number of potential issues.

- 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;
}
```

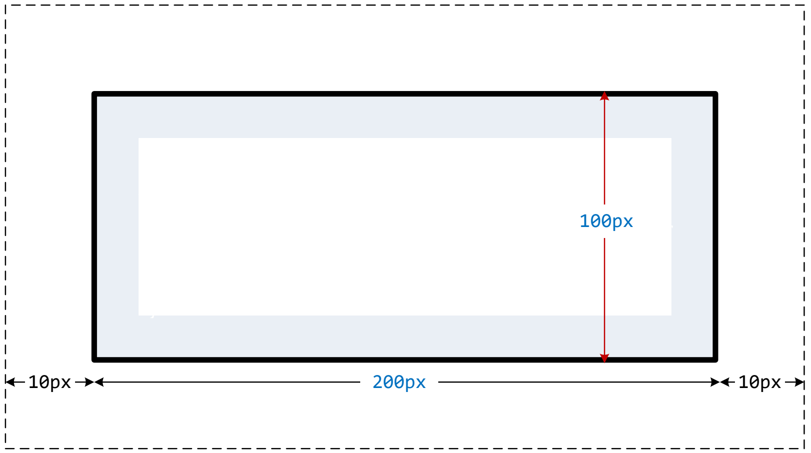
True element width = 10 + 2 + 5 + 200 + 5 + 2 + 10 = 234 px
True element height = 10 + 2 + 5 + 100 + 5 + 2 + 10 = 134 px



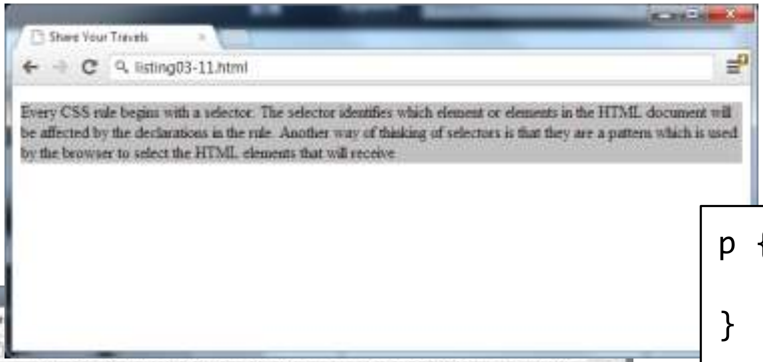


```
div {
  ...
  box-sizing: border-box;
}
```

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

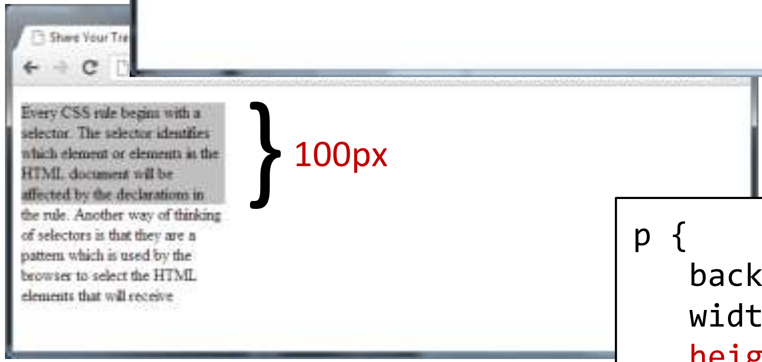


# Width and Height



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;
}
```

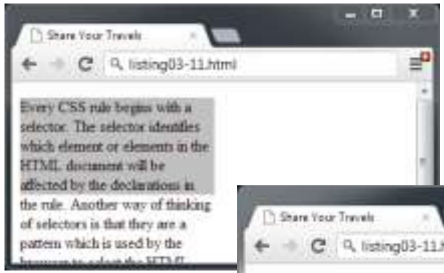


Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML elements that will receive

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

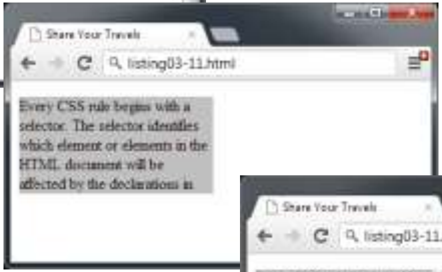
# Overflow Property

`overflow: visible;`



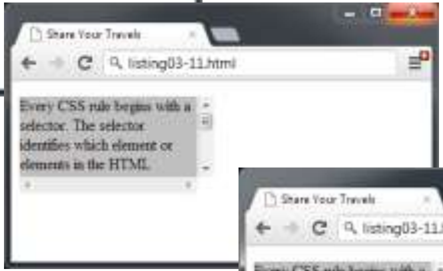
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. Another way of thinking of selectors is that they are a pattern which is used by the browser to select the HTML

`overflow: hidden;`



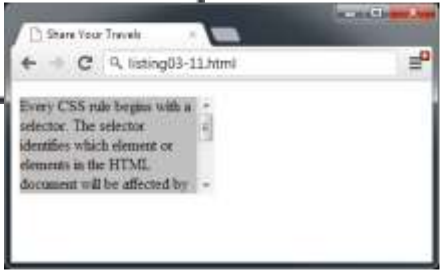
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in

`overflow: scroll;`



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

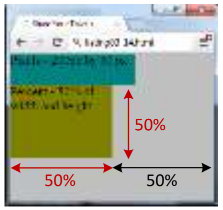
`overflow: auto;`



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

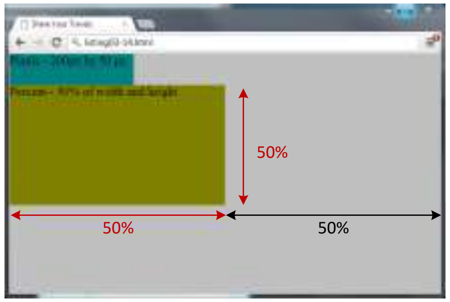
## Sizing Elements

- 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 **percentage**s, the size is relative to the size of the parent element.
  - When you use **em**s, 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.

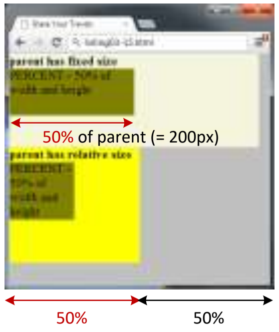


```
<body>
   <div class="pixels">
     Pixels - 200px by 50 px
   </div>
   <div class="percent">
     Percent - 50% of width and height
   </div>
</body>
```

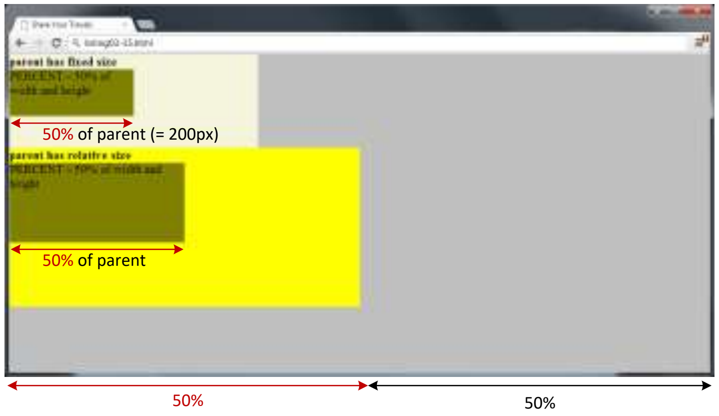
```
<style>
  html,body {
     margin:0;
     width:100%;
     height:100%;
     background: silver;
  }
  .pixels {
     width:200px;
     height:50px;
     background: teal;
  }
  .percent {
     width:50%;
     height:50%;
     background: olive;
  }
```



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

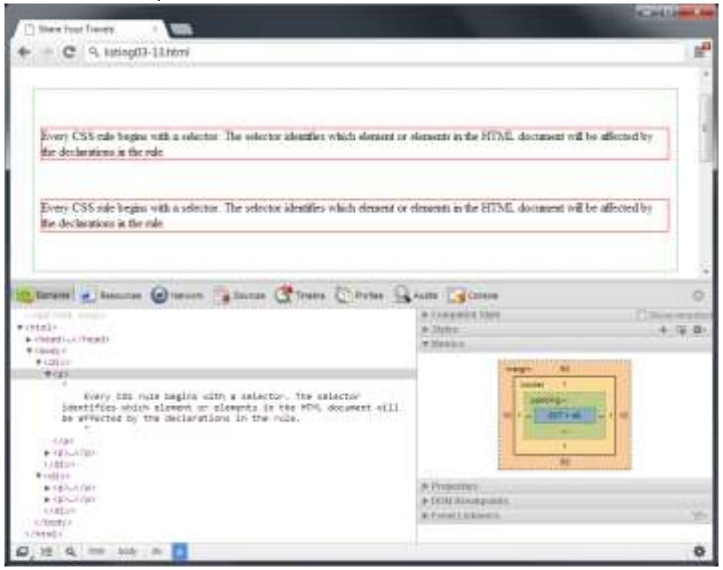


```
<body>
<div class="parentFixed">
   <strong>parent has fixed size</strong>
   <div class="percent">
      PERCENT - 50% of width and height
   </div>
</div>
<div class="parentRelative">
   <strong>parent has relative size</strong>
   <div class="percent">
      PERCENT - 50% of width and height
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



# Developer Tools

Chrome – Inspect Element



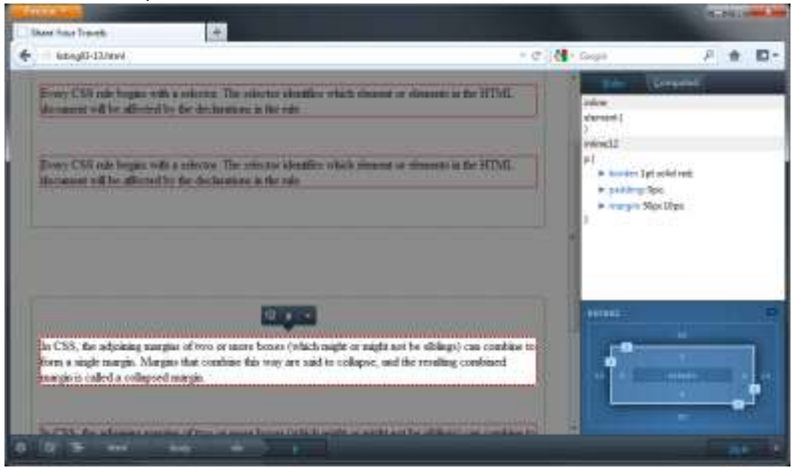
Share Your Travels

listing03-13.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.

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.

Firefox – Inspect



Share Your Travels

listing03-13.html

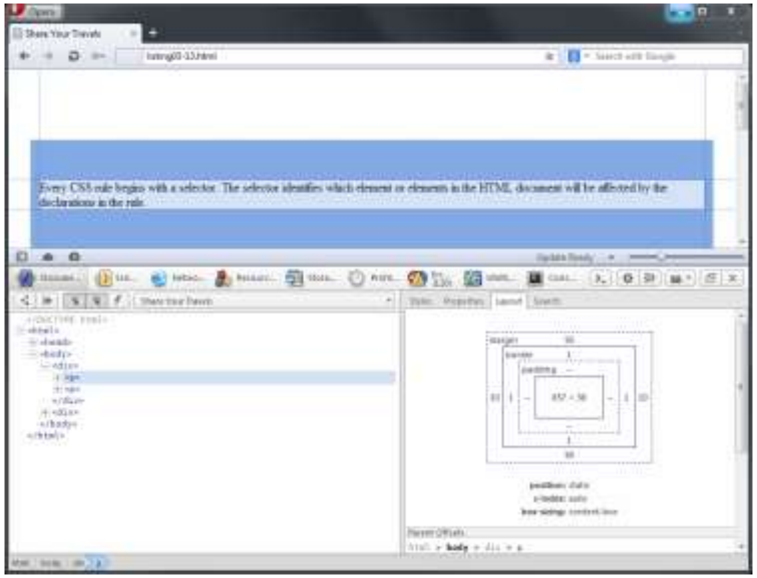
Every CSS rule begins with a selector. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

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.

```
p {
  border: 1px solid red;
  padding: 5px;
  margin: 50px 10px;
}
```

Opera – Inspect Element

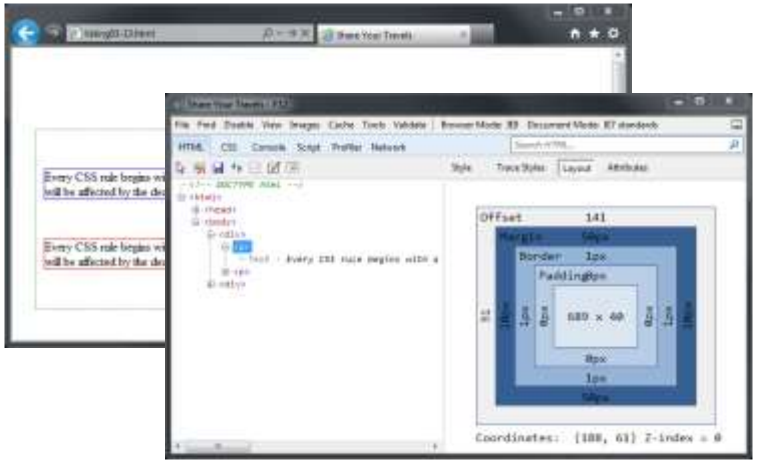


Share Your Travels

listing03-13.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.

Internet Explorer – Developer Tools



listing03-13.html

Share Your Travels

Every CSS rule begins wi
will be affected by the de

Every CSS rule begins wi
will be affected by the de

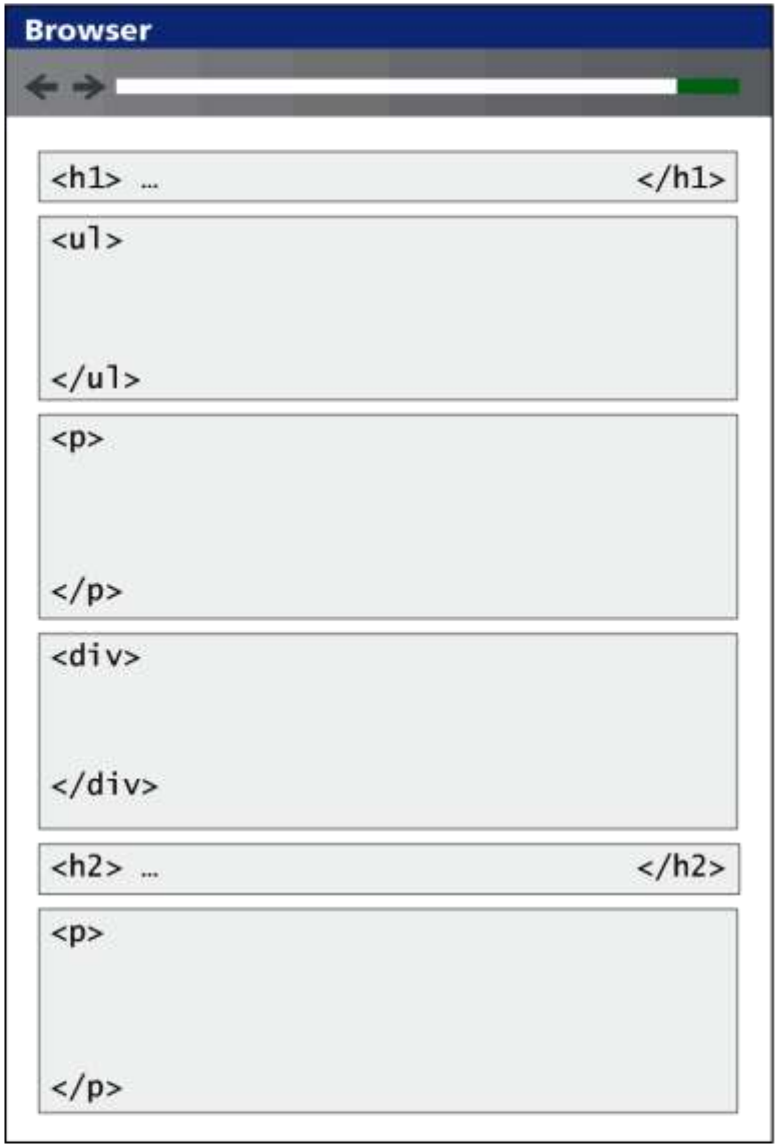
File Find Disable View Images Cache Tools Validate Browser Mode: IE9 Document Mode: IE7 standards
HTML CSS Console Script Profiler Network
Style Trace Styles Layout Attributes
Offset 141
Margin 50px
Border 1px
Padding 8px
680 x 40
Coordinates: (188, 61) Z-index = 0

## Layout: Normal Flow

## Normal Flow

- **Normal flow** refers here to how the browser will normally display block-level elements and inline elements from left to right and from top to bottom
  - **Block-level elements** are each contained on their own line
    - <p>, <div>, <h2>, <ul>, and <table>
  - **Inline elements** do not form their own blocks but instead are displayed within lines
    - as <em>, <a>, <img>, and <span>

## Normal Flow



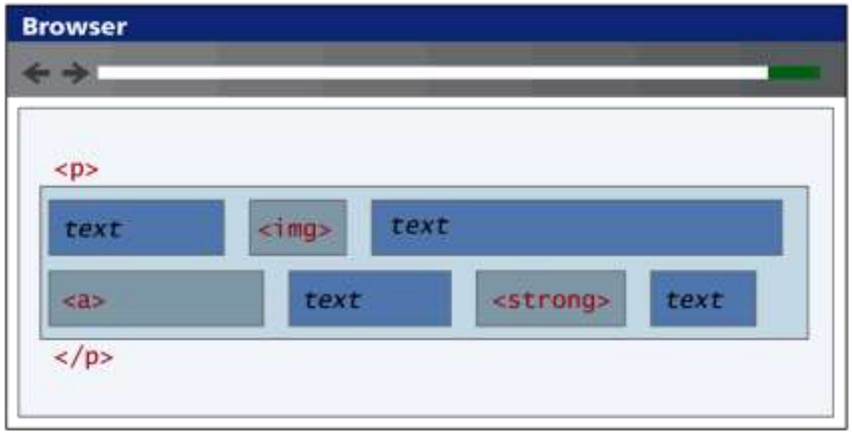
Each block exists on its own line and is displayed in normal flow from the browser window's top to its bottom.

By default each block-level element fills up the entire width of its parent (in this case, it is the <body>, which is equivalent to the width of the browser window).

You can use CSS box model properties to customize, for instance, the width of the box and the margin space between other block-level elements.

## Normal Flow

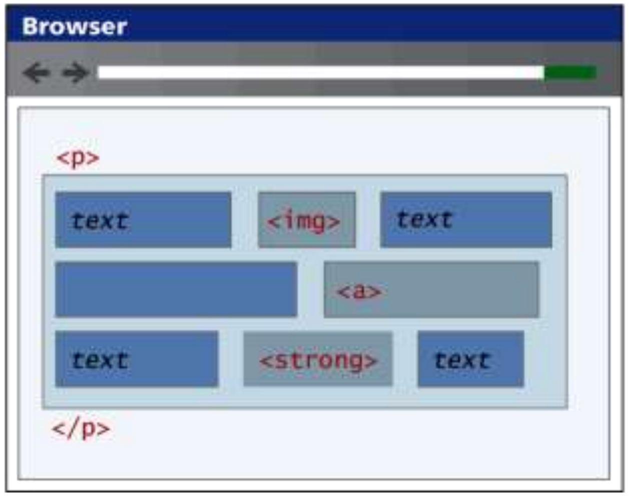
```
<p>
This photo <img src="photo-con.png" alt="" /> of Conservatory Pond in
<a href="http://www.centralpark.com/">Central Park</a> New York City
was taken on October 22, 2015 with a <strong>Canon EOS 30D</strong>
camera.
</p>
```



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

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

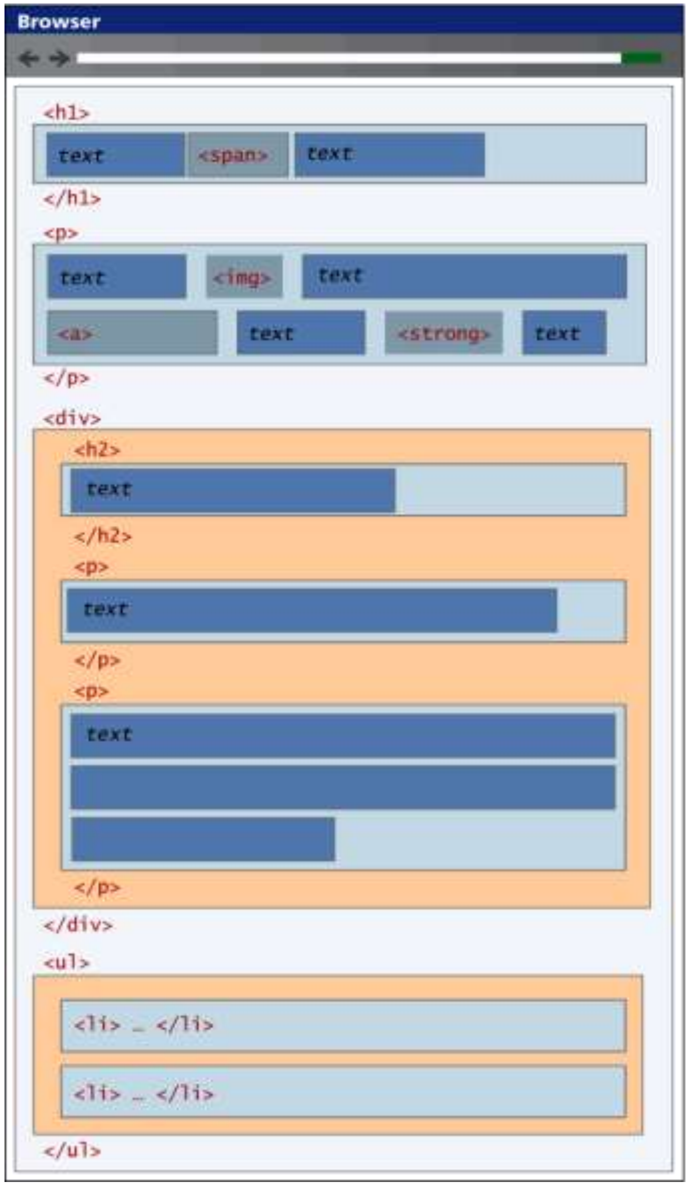
Here the content of this <p> element is displayed on two lines.



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

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

# Working Together



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

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

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

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

## Inline Elements

- There are actually two types of inline elements: replaced and nonreplaced.
  - **Replaced inline elements** are elements whose content and thus appearance is defined by some external resource, such as <img> and the various form elements.
  - **Nonreplaced inline elements** are those elements whose content is defined within the document, which includes all the other inline elements.

## Take control

- It is possible to change whether an element is block-level or inline via the CSS **display** property. Consider the following two CSS rules:
  - **span { display: block; }**
  - **li { display: inline; }**
- These two rules will make all <span> elements behave like block-level elements and all <li> elements like inline (that is, each list item will be displayed on the same line).

## Layout: Positioning Elements

CSLAB

## Positioning Elements

- It is possible to move an item from its regular position in the normal flow, and
  - move an item outside of the browser viewport so it is not visible
  - position it so it is always visible in a fixed position while the rest of the content scrolls.

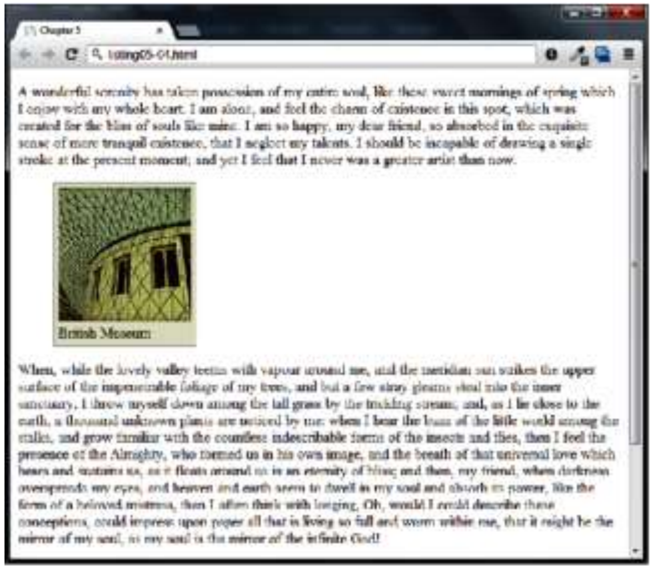
- The **position** property is used to specify the type of positioning

| Value | Description |
|---|---|
| **absolute** | The element is removed from normal flow and positioned in relation to its nearest positioned ancestor. |
| **fixed** | The element is fixed in a specific position in the window even when the document is scrolled. |
| **relative** | The element is moved relative to where it would be in the normal flow. |
| **static** | The element is positioned according to the normal flow. This is the default. |

## Relative Positioning

- In **relative positioning** an element is displaced out of its normal flow position and moved relative to where it would have been placed
- The other content around the relatively positioned element “remembers” the element’s old position in the flow; thus the space the element would have occupied is preserved

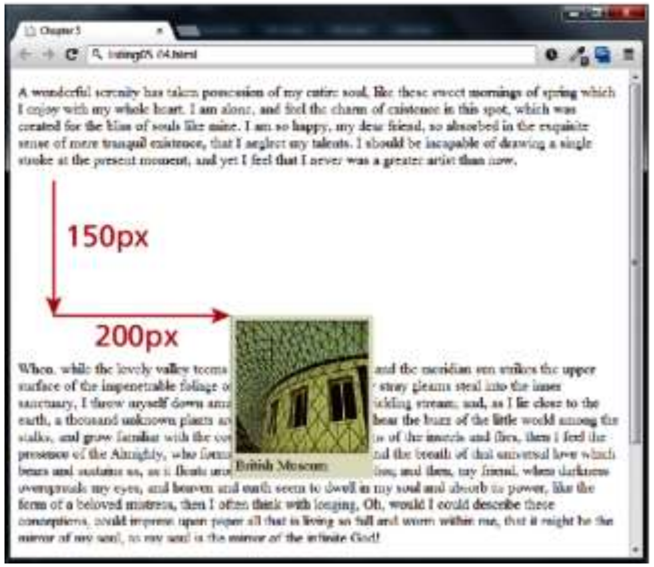
## Relative Positioning



A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart. I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single stroke at the present moment; and yet I feel that I never was a greater artist than now.

When, while the lovely valley teems with vapour around me, and the meridian sun strikes the upper surface of the impenetrable foliage of my trees, and but a few stray gleams steal into the inner sanctuary, I throw myself down among the tall grass by the trickling stream; and, as I lie close to the earth, a thousand unknown plants are noticed by me: when I hear the buzz of the little world among the stalks, and grow familiar with the countless indescribable forms of the insects and flies, then I feel the presence of the Almighty, who formed us in his own image, and the breath of that universal love which bears and sustains us, as it floats around us in an eternity of bliss; and then, my friend, when darkness overspreads my eyes, and heaven and earth seem to dwell in my soul and absorb its power, like the form of a beloved mistress, then I often think with longing, Oh, would I could describe these conceptions, could impress upon paper all that is living so full and warm within me, that it might be the mirror of my soul, as my soul is the mirror of the infinite God!

```
<p>A wonderful serenity has taken possession of my …

<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>

<p>When, while the lovely valley …
```



A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart. I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single stroke at the present moment; and yet I feel that I never was a greater artist than now.

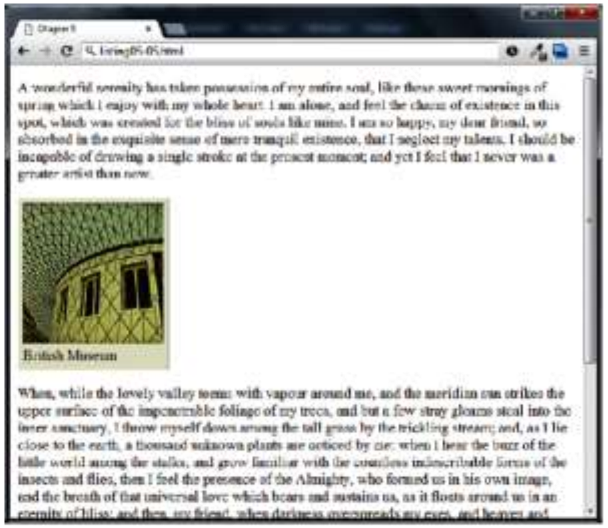
When, while the lovely valley teems and the meridian sun strikes the upper
surface of the impenetrable foliage o stray gleams steal into the inner
sanctuary, I throw myself down amo ickling stream; and, as I lie close to the
earth, a thousand unknown plants ar hear the buzz of the little world among the
stalks, and grow familiar with the cou s of the insects and flies, then I feel the
presence of the Almighty, who form nd the breath of that universal love which
bears and sustains us, as it floats arou liss; and then, my friend, when darkness
overspreads my eyes, and heaven and earth seem to dwell in my soul and absorb its power, like the form of a beloved mistress, then I often think with longing, Oh, would I could describe these conceptions, could impress upon paper all that is living so full and warm within me, that it might be the mirror of my soul, as my soul is the mirror of the infinite God!

```
figure {
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   padding: 5px;
   width: 150px;
   position: relative;
   top: 150px;
   left: 200px;
}
```

## Absolute Positioning

- When an element is positioned absolutely, it is removed completely from normal flow.
- Unlike with relative positioning, space is not left for the moved element, as it is no longer in the normal flow.
- **absolute positioning** can get confusing

# Absolute Positioning

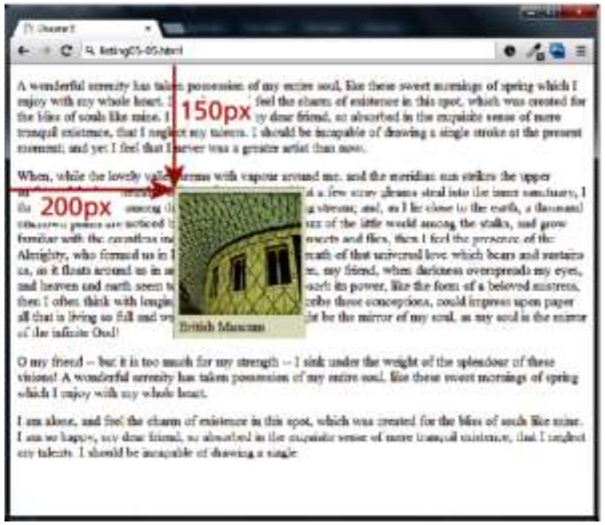


A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart. I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single stroke at the present moment; and yet I feel that I never was a greater artist than now.

British Museum

When, while the lovely valley teems with vapour around me, and the meridian sun strikes the upper surface of the impenetrable foliage of my trees, and but a few stray gleams steal into the inner sanctuary, I throw myself down among the tall grass by the trickling stream; and, as I lie close to the earth, a thousand unknown plants are noticed by me: when I hear the buzz of the little world among the stalks, and grow familiar with the countless indescribable forms of the insects and flies, then I feel the presence of the Almighty, who formed us in his own image, and the breath of that universal love which bears and sustains us, as it floats around us in an eternity of bliss; and then, my friend, when darkness overspreads my eyes, and heaven and

```
<p>A wonderful serenity has taken possession of my …

<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>

<p>When, while the lovely valley …
```



A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart. I feel the charm of existence in this spot, which was created for the bliss of souls like mine. I my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single stroke at the present moment; and yet I feel that I never was a greater artist than now.

When, while the lovely valley teems with vapour around me, and the meridian sun strikes the upper a few stray gleams steal into the inner sanctuary, I the among the g stream; and, as I lie close to the earth, a thousand unknown plants are noticed buzz of the little world among the stalks, and grow familiar with the countless in insects and flies, then I feel the presence of the Almighty, who formed us in breath of that universal love which bears and sustains us, as it floats around us in e, my friend, when darkness overspreads my eyes, and heaven and earth seem to dwell in my soul and absorb its power, like the form of a beloved mistress, then I often think with longing describe these conceptions, could impress upon paper all that is living so full and w it be the mirror of my soul, as my soul is the mirror of the infinite God!

British Museum

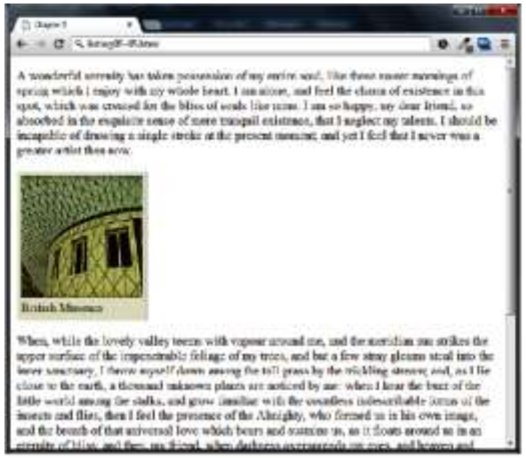
O my friend – but it is too much for my strength – I sink under the weight of the splendour of these visions! A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart.

I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single

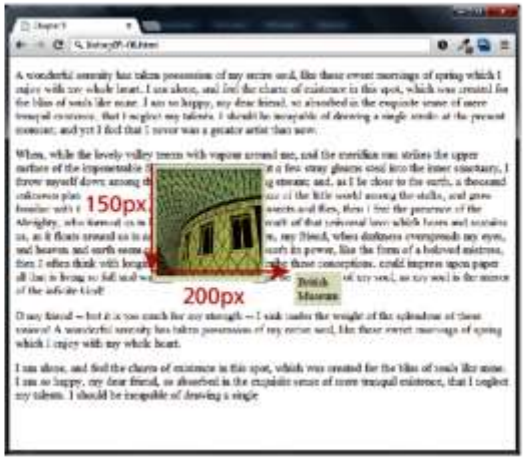
```
figure {
   margin: 0;
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   padding: 5px;
   width: 150px;
   position: absolute;
   top: 150px;
   left: 200px;
}
```

## Absolute Positioning

- A moved element via absolute position is actually positioned relative to its nearest **positioned** ancestor container



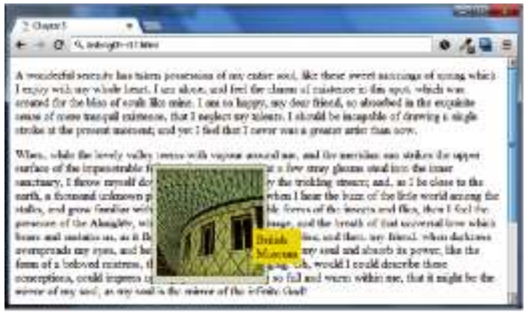
```
<p>A wonderful serenity has taken possession of my …

<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>

<p>When, while the lovely valley …
```



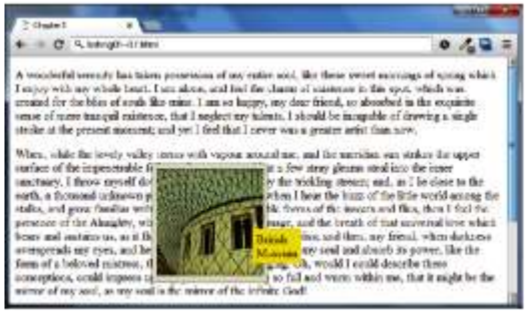
```
figure {
   margin: 0;
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   padding: 5px;
   width: 150px;
   position: absolute;
   top: 150px;
   left: 200px;
}
figcaption {
   background-color: #EDEDDD;
   padding: 5px;
   position: absolute;
   top: 150px;
   left: 200px;
}
```

# Z-Index

- When overlapping elements items closest to the viewer (and thus on the top) have a larger z-index
  - only positioned elements will make use of their z-index
  - simply setting the z-index value of elements will not necessarily move them on top or behind other items.

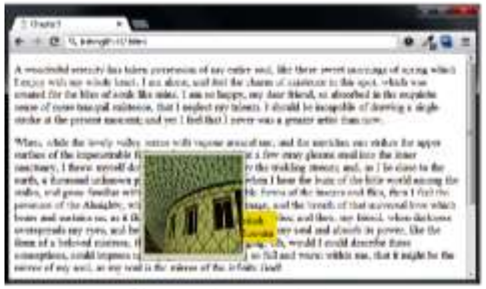


```
figure {
   position: absolute;
   top: 150px;
   left: 200px;
}
figcaption {
   position: absolute;
   top: 90px;
   left: 140px;
}
```



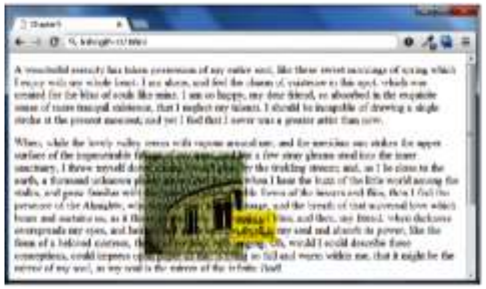
```
figure {
   ...
   z-index: 5;
}
figcaption {
   ...
   z-index: 1;
}
```

Note that this did **not** move the `<figure>` on top of the `<figcaption>` as one might expect. This is due to the nesting of the caption within the figure.



```
figure {
   ...
   z-index: 1;
}
figcaption {
   ...
   z-index: -1;
}
```

Instead the `<figcaption>` `z-index` must be set below 0. The `<figure>` `z-index` could be any value equal to or above 0.



```
figure {
   ...
   z-index: -1;
}
figcaption {
   ...
   z-index: 1;
}
```

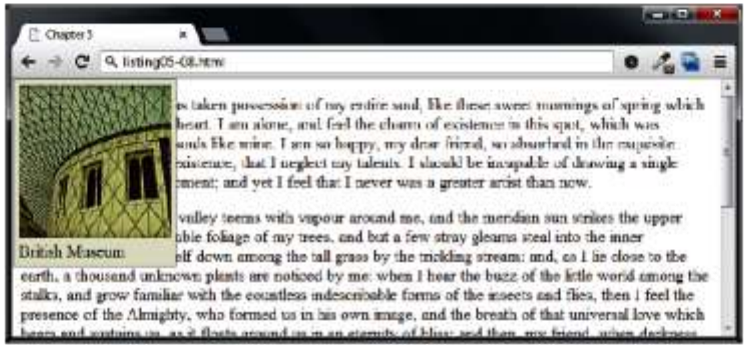
If the `<figure>` `z-index` is given a value less than 0, then any of its positioned descendants change as well. Thus both the `<figure>` and `<figcaption>` move underneath the body text.

## Fixed Position

- Elements with **fixed positioning** do not move when the user scrolls up or down the page

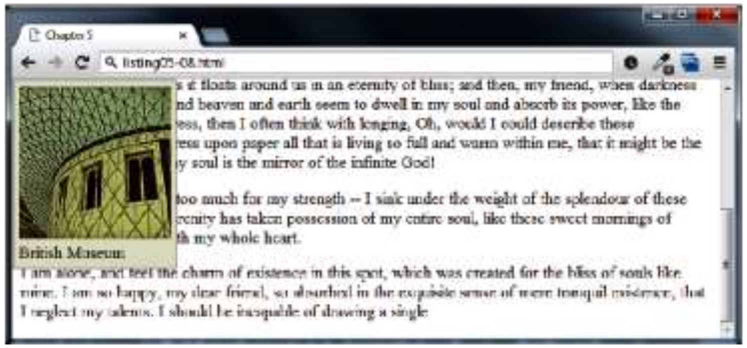
```
figure {
    ...
    position: fixed;
    top: 0;
    left: 0;
}
```

Notice that figure is fixed in its position regardless of what part of the page is being viewed.



s taken possession of my entire soul, like these sweet mornings of spring which
heart. I am alone, and feel the charm of existence in this spot, which was
souls like mine. I am so happy, my dear friend, so absorbed in the exquisite
existence, that I neglect my talents. I should be incapable of drawing a single
oment; and yet I feel that I never was a greater artist than now.

valley teems with vapour around me, and the meridian sun strikes the upper
ible foliage of my trees, and but a few stray gleams steal into the inner
elf down among the tall grass by the trickling stream; and, as I lie close to the
earth, a thousand unknown plants are noticed by me: when I hear the buzz of the little world among the
stalks, and grow familiar with the countless indescribable forms of the insects and flies, then I feel the
presence of the Almighty, who formed us in his own image, and the breath of that universal love which



s it floats around us in an eternity of bliss; and then, my friend, when darkness
nd heaven and earth seem to dwell in my soul and absorb its power, like the
ess, then I often think with longing, Oh, would I could describe these
ress upon paper all that is living so full and warm within me, that it might be the
y soul is the mirror of the infinite God!

too much for my strength -- I sink under the weight of the splendour of these
ernity has taken possession of my entire soul, like these sweet mornings of
th my whole heart.

I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like
mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that
I neglect my talents. I should be incapable of drawing a single

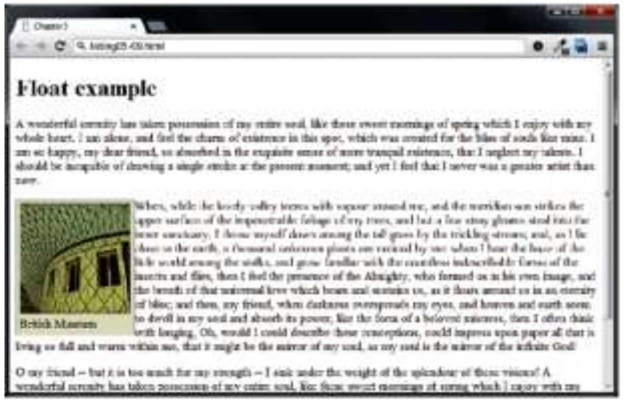
## Floating Elements

## Floating Elements

- It is possible to displace an element out of its position in the normal flow via the CSS float **property**
- An element can be floated to the **left** or floated to the **right**
- When an item is floated, it is moved all the way to the far left or far right of its containing block and the rest of the content is “re-flowed” around the floated element

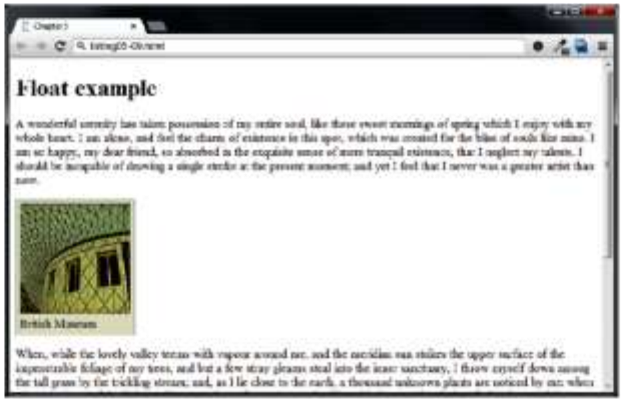
| Value | Description |
|---|---|
| **left** | The left-hand edge of the element cannot be adjacent to another element. |
| **right** | The right-hand edge of the element cannot be adjacent to another element. |
| **both** | Both the left-hand and right-hand edges of the element cannot be adjacent to another element. |
| **none** | The element can be adjacent to other elements. |

# Floating Elements

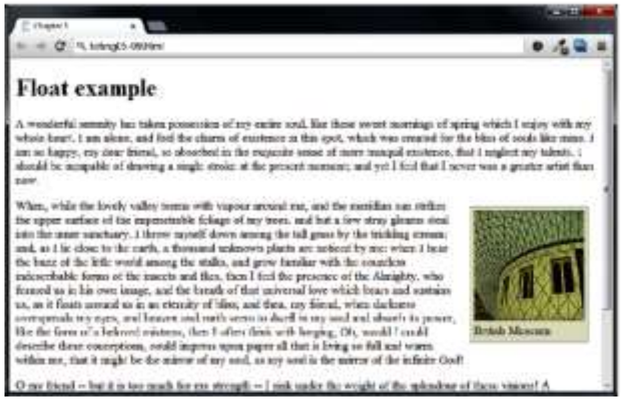


```
<h1>Float example</h1>
<p>A wonderful serenity has taken ...</p>
<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>
<p>When, while the lovely valley …</p>

figure {
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   margin: 0;
   padding: 5px;
   width: 150px;
}
```

Notice that a floated block-level element **must** have a width specified.



```
figure {
   …
   width: 150px;
   float: left;
}
```



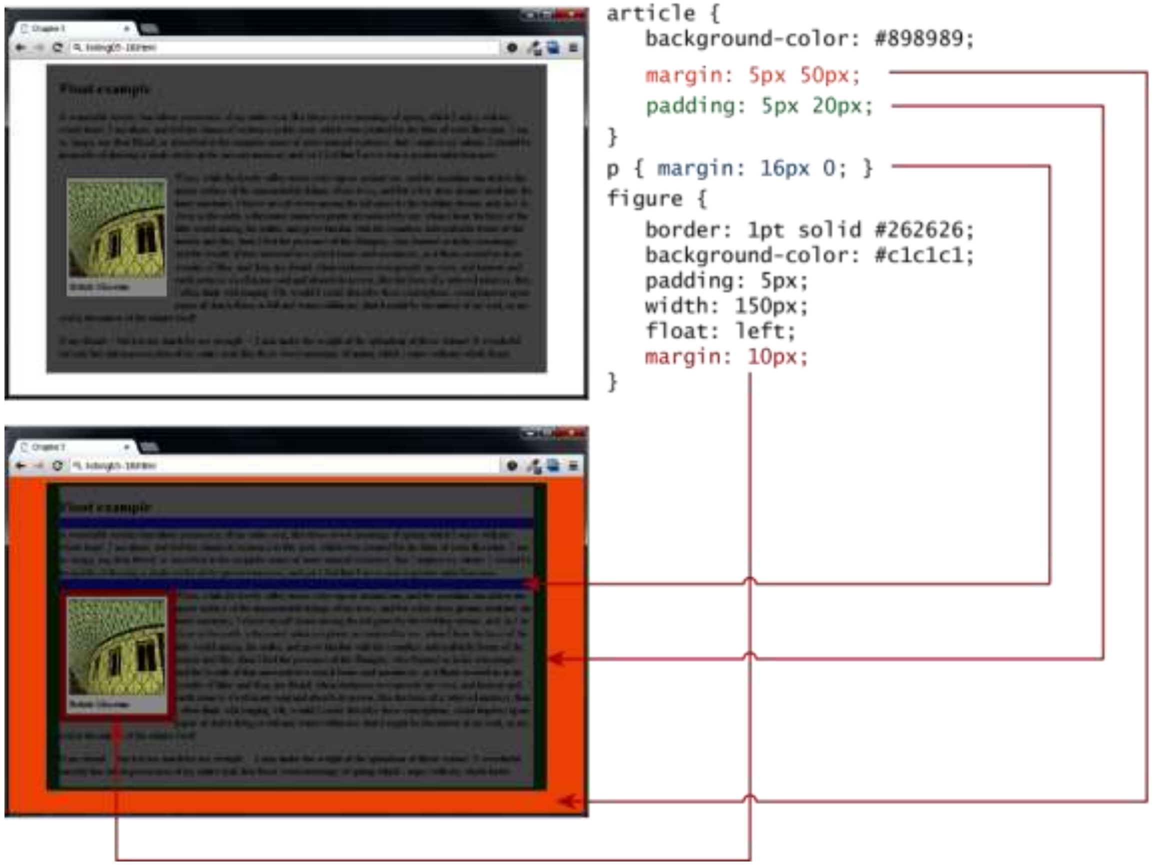
```
figure {
   …
   width: 150px;
   float: right;
   margin: 10px;
}
```

## Floating Elements

- Notice that a **floated block-level element must have a width specified**, if you do not, then the width will be set to auto, which will mean it implicitly fills the entire width of the containing block, and there thus will be no room available to flow content around the floated item.
- It should be reiterated that a floated item moves to the left or right of its container (also called its **containing block**).

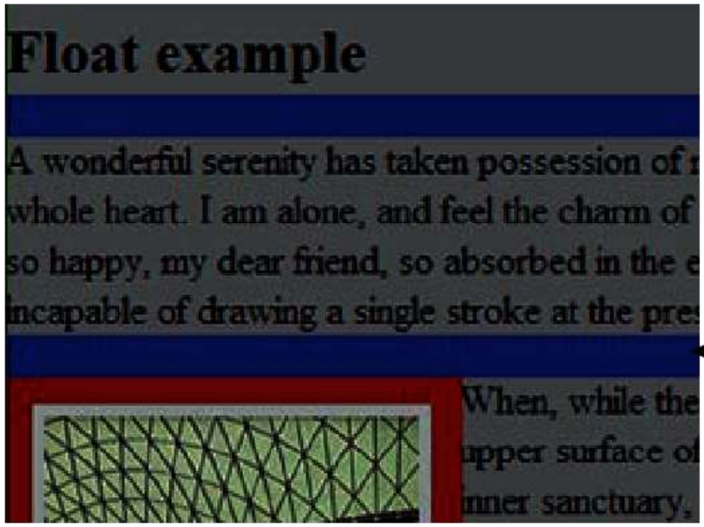
# Floating within a container

```
<article>
  <h1>Float example</h1>
  <p>A wonderful serenity has taken possession of … </p>

  <figure>
    <img src="images/828.jpg" alt="" />
    <figcaption>British Museum</figcaption>
  </figure>

  <p>When, while the lovely valley teems with ...</p>

  <p>O my friend -- but it is too much for my ...</p>
</article>
```



```
article {
   background-color: #898989;
   margin: 5px 50px;
   padding: 5px 20px;
}
p { margin: 16px 0; }
figure {
   border: 1pt solid #262626;
   background-color: #c1c1c1;
   padding: 5px;
   width: 150px;
   float: left;
   margin: 10px;
}
```

# Floating within a container



Float example

A wonderful serenity has taken possession of
whole heart. I am alone, and feel the charm of
so happy, my dear friend, so absorbed in the e
incapable of drawing a single stroke at the pres

When, while the
upper surface of
inner sanctuary,

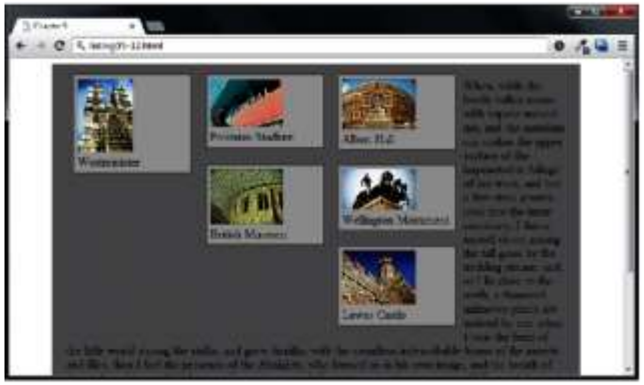
<p> margin-bottom: 16px;
Notice these margins collapse (normal behavior).
<p> margin-top: 16px;
But these margins did **not** collapse.
<figure> margin-top: 10px;

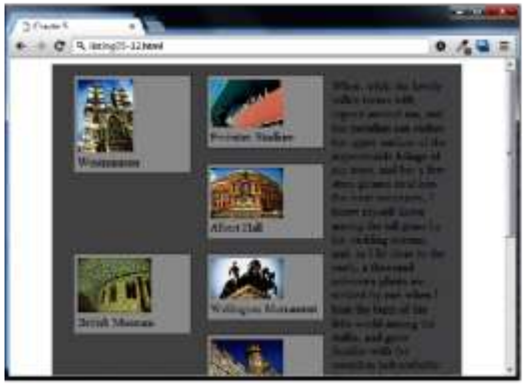
## Floating Multiple Items side by side

- When you float multiple items that are in proximity, each floated item in the container will be **nestled up beside the previously floated item**
- All other content in the containing block (including other floated elements) will flow around all the floated elements

# Floating side by side

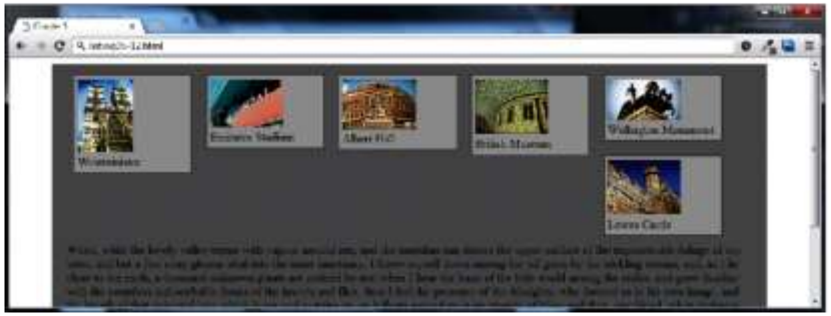
- Example





```
<article>
  <figure>
     <img src="images/tiny/275.jpg" alt="" />
     <figcaption>Westminister</figcaption>
  </figure>
  <figure>
     <img src="images/tiny/700.jpg" alt="" />
     <figcaption>Emirates Stadium</figcaption>
  </figure>
  <figure>
     <img src="images/tiny/537.jpg" alt="" />
     <figcaption>Albert Hall</figcaption>
  </figure>
  <figure>
     <img src="images/tiny/828.jpg" alt="" />
     <figcaption>British Museum</figcaption>
  </figure>
  <figure>
     <img src="images/tiny/464.jpg" alt="" />
     <figcaption>Wellington Monument</figcaption>
  </figure>
  <figure>
     <img src="images/tiny/224.jpg" alt="" />
     <figcaption>Lewes Castle</figcaption>
  </figure>
  <p>When, while the lovely valley teems ..
</article>
```

```
figure {
   ...
   width: 150px;
   float: left;
}
```

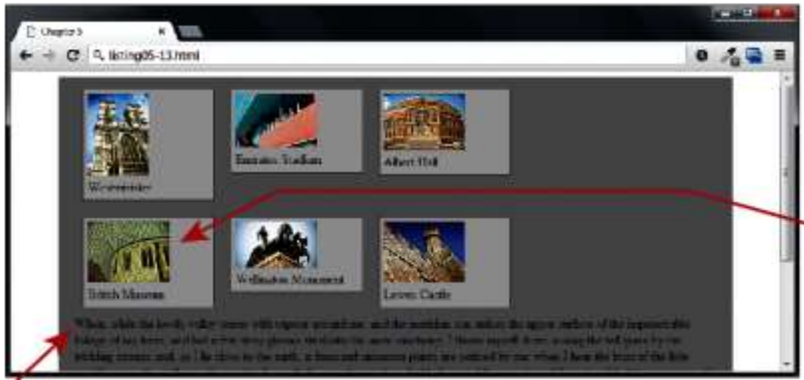


As the window resizes, the content in the containing block (the `<article>` element), will try to fill the space that is available to the right of the floated elements.

# The Clear Property

- You can stop elements from flowing around a floated element by using the clear **property**

```
.first { clear: left; }
```

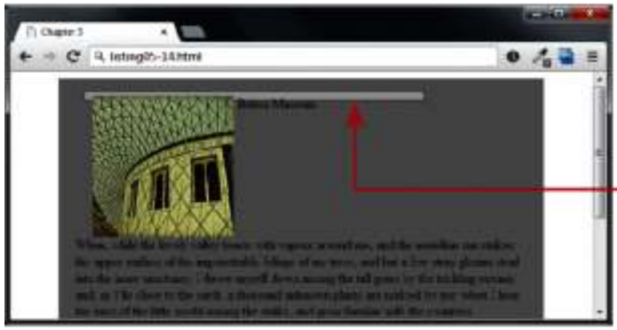


```
<article>
   <figure>
      <img src="images/tiny/275.jpg" alt="" />
      <figcaption>Westminister</figcaption>
   </figure>
   <figure>
      <img src="images/tiny/700.jpg" alt="" />
      <figcaption>Emirates Stadium</figcaption>
   </figure>
   <figure>
      <img src="images/tiny/537.jpg" alt="" />
      <figcaption>Albert Hall</figcaption>
   </figure>
   <figure class="first">
      <img src="images/tiny/828.jpg" alt="" />
      <figcaption>British Museum</figcaption>
   </figure>
   <figure>
      <img src="images/tiny/464.jpg" alt="" />
      <figcaption>Wellington Monument</figcaption>
   </figure>
   <figure>
      <img src="images/tiny/224.jpg" alt="" />
      <figcaption>Lewes Castle</figcaption>
   </figure>
   <p class="first">When, while the lovely valley ..
</article>
```

## Containing Floats

- Another problem that can occur with floats is when an element is floated within a containing block that contains *only* floated content. In such a case, the containing block essentially disappears

```
<article>
 <figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
 </figure>
 <p class="first">When, while the lovely valley …
</article>
```

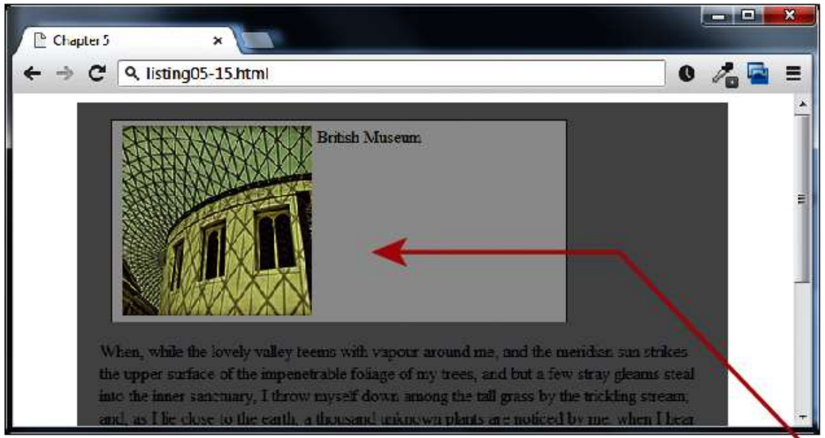


Notice that the <figure> element's content area has shrunk down to zero (it now just has padding space and borders).

```
figure img {
   width: 170px;
   float: left;
   margin: 0 5px;
}
figure figcaption {
   width: 100px;
   float: left;
}
figure {
   border: 1pt solid #262626;
   background-color: #c1c1c1;
   padding: 5px;
   width: 400px;
   margin: 10px;
}
.first { clear: left; }
```

## Containing Floats

- One solution would be to float the container as well, but depending on the layout this might not be possible. A better solution would be to use the **overflow** property



When, while the lovely valley teems with vapour around me, and the meridian sun strikes the upper surface of the impenetrable foliage of my trees, and but a few stray gleams steal into the inner sanctuary, I throw myself down among the tall grass by the trickling stream, and, as I lie close to the earth, a thousand unknown plants are noticed by me: when I hear

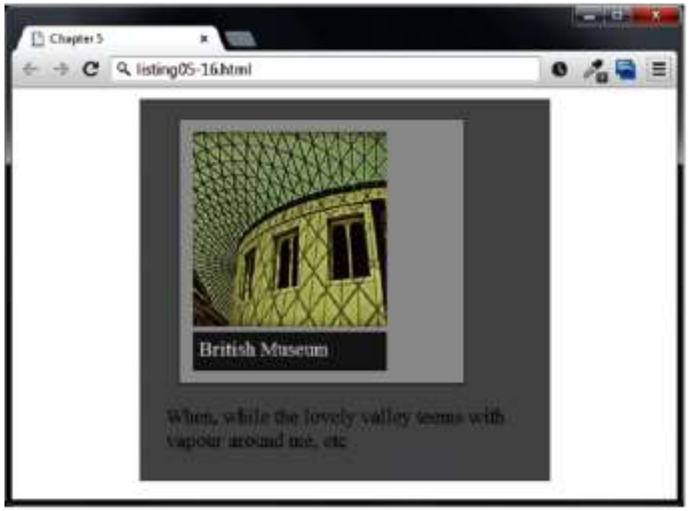
Setting the `overflow` property to auto solves the problem.

```
figure img {
   width: 170px;
   float: left;
   margin: 0 5px;
}
figure figcaption {
   width: 100px;
   float: left;
}
figure {
   border: 1pt solid #262626;
   background-color: #c1c1c1;
   padding: 5px;
   width: 400px;
   margin: 10px;
   overflow: auto;
}
```

## Overlaying and Hiding Elements

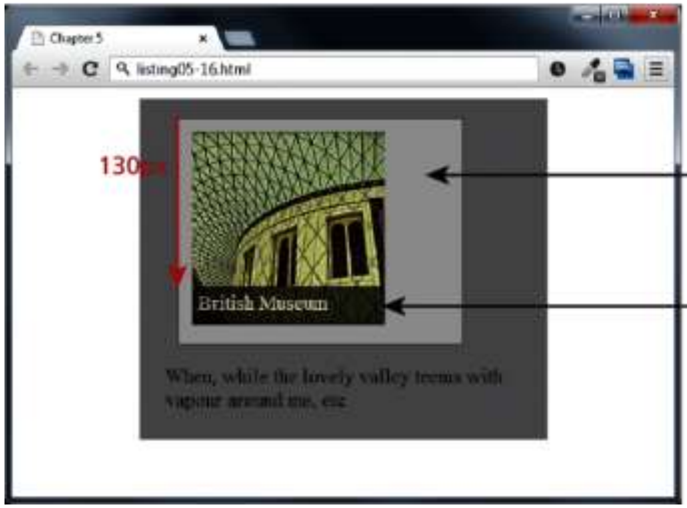
- One of the more common design tasks with CSS is to place two elements on top of each other, or to selectively hide and display elements. Positioning is important to both of these tasks.
- Positioning is often used for smaller design changes, such as moving items relative to other elements within a container
- In such a case, relative positioning is used to create the **positioning context** for a subsequent absolute positioning move

# Positioning Context



Chapter 5
listing05-16.html
British Museum
When, while the lovely valley teems with vapour around me, etc

```
figure {
   border: 1pt solid #262626;
   background-color: #c1c1c1;
   padding: 10px;
   width: 200px;
   margin: 10px;
}

figcaption {
   background-color: black;
   color: white;
   opacity: 0.6;
   width: 140px;
   height: 20px;
   padding: 5px;
}
```



Chapter 5
listing05-16.html
130px
British Museum
When, while the lovely valley teems with vapour around me, etc

```
figure {
   ...
   position: relative;
}
figcaption {
   ...
   position: absolute;
   top: 130px;
   left: 10px;
}
```

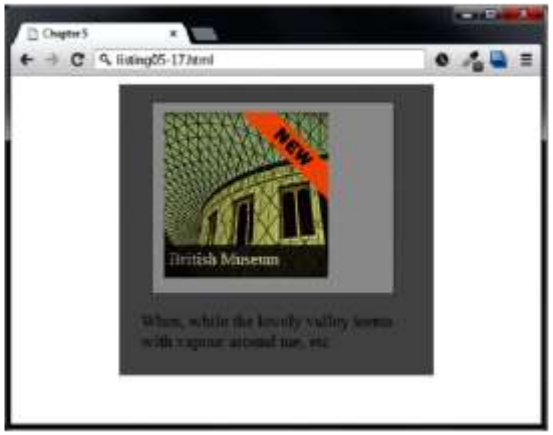
This creates the positioning context.

This does the actual move.

# Positioning Context

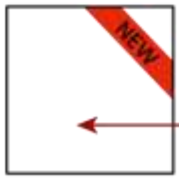
- A realistic example

```
<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
   <img src="images/new-banner.png" alt="" class="overlayed"/>
</figure>
```



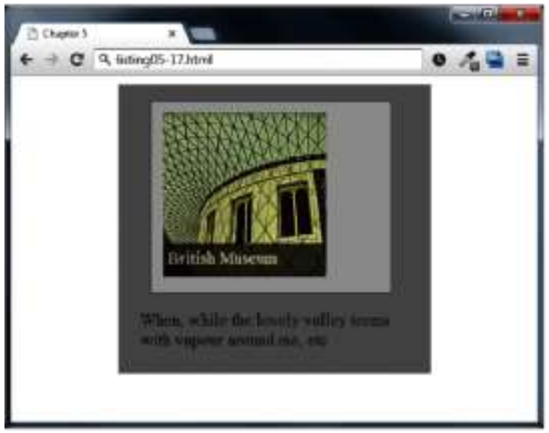
When, while the lovely valley teems with vapour around me, etc

```
.overlayed {
   position: absolute;
   top: 10px;
   left: 10px;
}
```



Transparent area

new-banner.png



When, while the lovely valley teems with vapour around me, etc

```
.overlayed {
   position: absolute;
   top: 10px;
   left: 10px;
   display: none;
}
```

This hides the overlayed image.

```
.hide {
   display: none;
}
```

This is the preferred way to hide: by adding this class to another element. This makes it clear in the markup that an element is not visible.

```
<img ... class="overlayed hide"/>
```

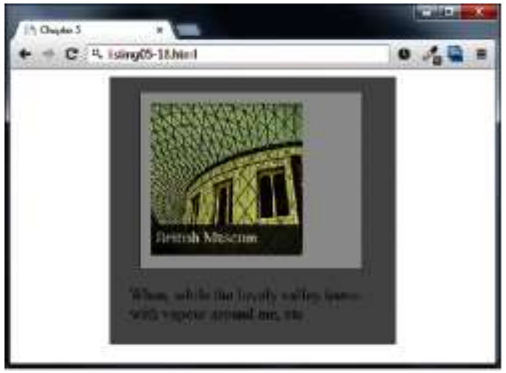
## Hiding elements

- Two different ways to hide elements in CSS:

1. using the display property
   - The display property takes an item out of the flow: it is as if the element no longer exists
2. using the visibility property
   - The visibility property just hides the element, but the space for that element remains.

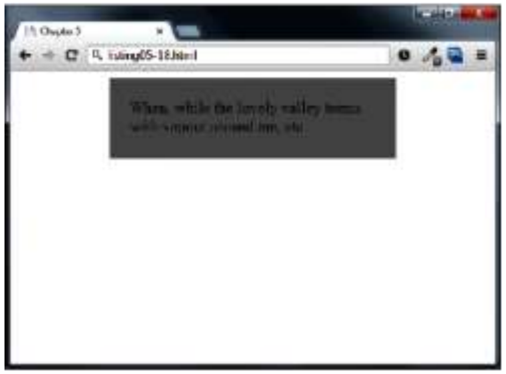
# Hiding elements

- Two ways to hide: display and visibility



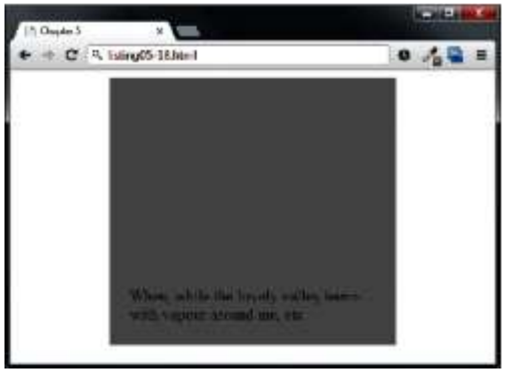
When, while the lovely valley teems with vapour around me, etc

```
figure {
    ...
    display: auto;
}
```



When, while the lovely valley teems with vapour around me, etc

```
figure {
    ...
    display: none;
}
```



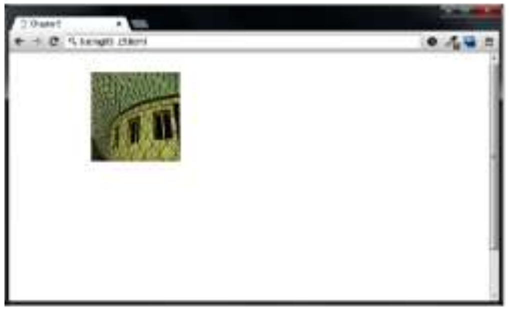
When, while the lovely valley teems with vapour around me, etc

```
figure {
    ...
    visibility: hidden;
}
```

## Using :hover to make thumbnails

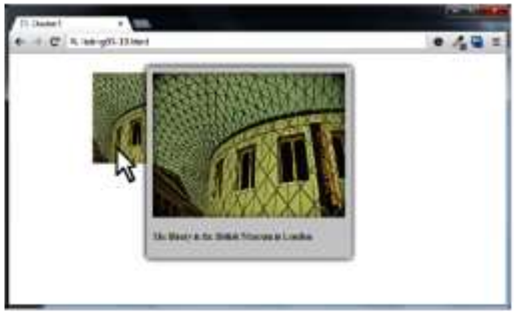
- Next slide shows how the combination of absolute positioning, the :hover pseudo-class, and the visibility property can be used to display a larger version of an image

```
<figure class="thumbnail">
  <img src="images/828.jpg" alt="" />
  <figcaption class="popup">
     <img src="images/828-bigger.jpg" alt="" />
     <p>The library in the British Museum in London</p>
  </figcaption>
</figure>
```



When the page is displayed, the larger version of the image, which is within the <figcaption> element, is hidden.

```
figcaption.popup {
   padding: 10px;
   background: #e1e1e1;
   position: absolute;

   /* add a drop shadow to the frame */
   -webkit-box-shadow: 0 0 15px #A9A9A9);
   -moz-box-shadow: 0 0 15px #A9A9A9;
   box-shadow: 0 0 15px #A9A9A9;

   /* hide it until there is a hover */
   visibility: hidden;
}
```



When the user moves/hovers the mouse over the thumbnail image, the visibility property of the <figcaption> element is set to visible.

```
figure.thumbnail:hover figcaption.popup {
   position: absolute;
   top: 0;
   left: 100px;

   /* display image upon hover */
   visibility: visible;
}
```

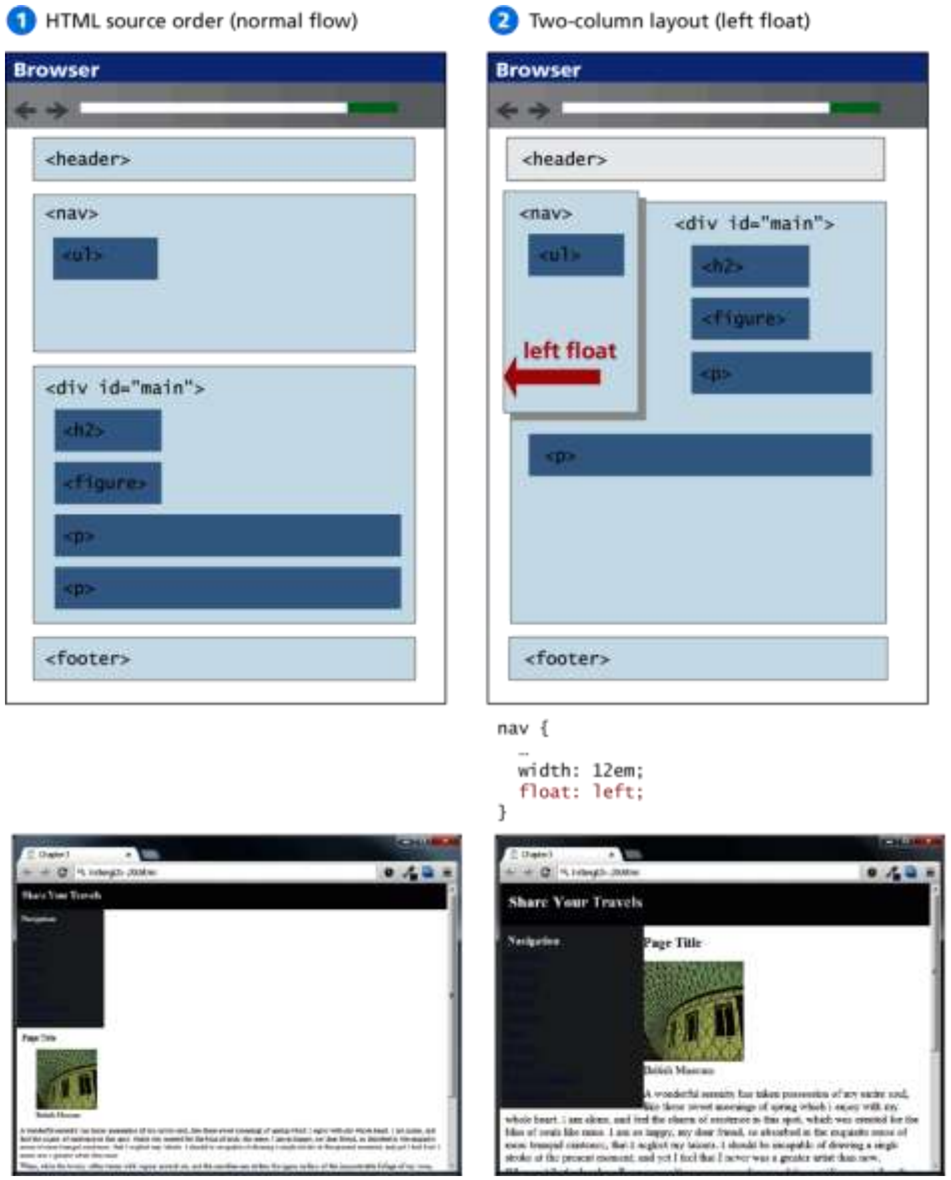
## Constructing Multicolumn Layouts

## Constructing Multicolumn Layouts

- There is unfortunately no simple and easy way to create robust multicolumn page layouts. There are tradeoffs with each approach:
    - Using Floats to Create Columns
    - Using Positioning to Create Columns

- The most common way to create columns of content is **using floats**.
    - **The first step** is to float the content container that will be on the left-hand side. Remember that the floated container needs to have a width specified.

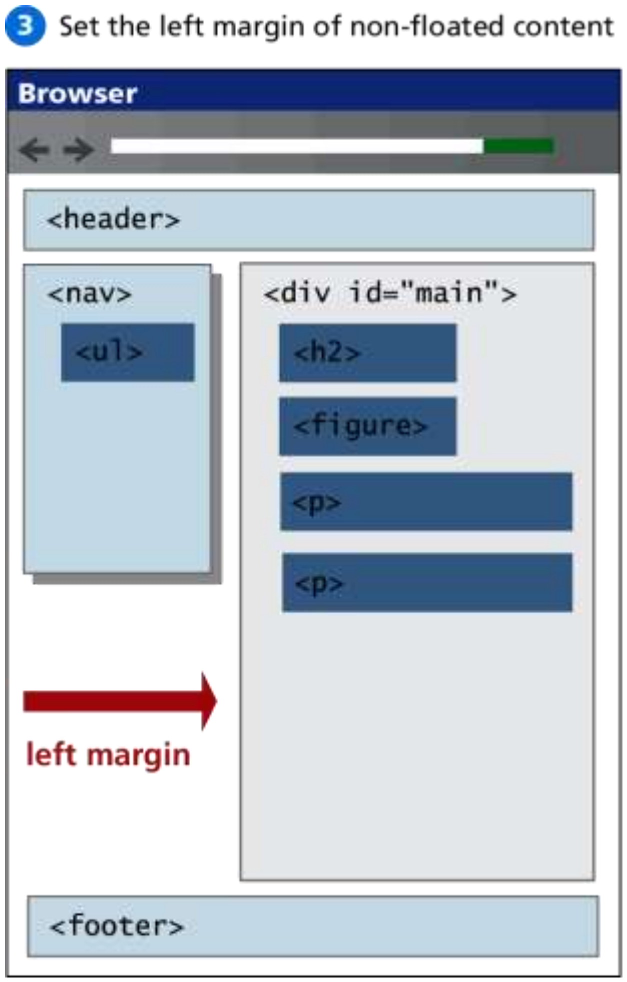
# Using Floats to create Columns

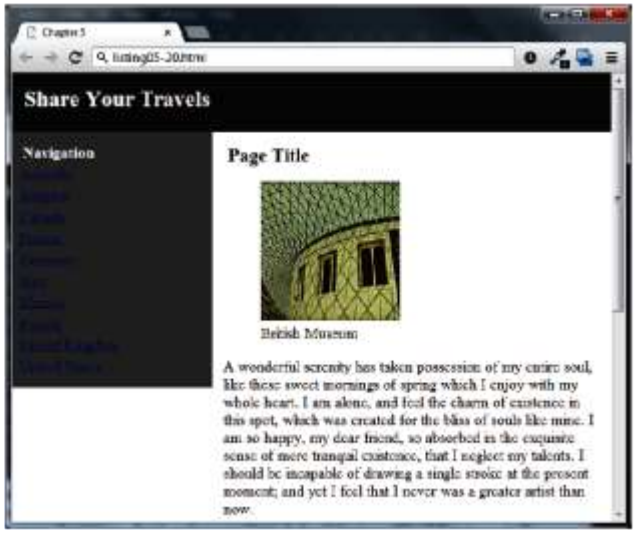


```
nav {
   …
   width: 12em;
   float: left;
}
```

## Using Floats to create Columns

- **The other key step** is changing the left-margin so that it no longer flows back under the floated content.





Share Your Travels

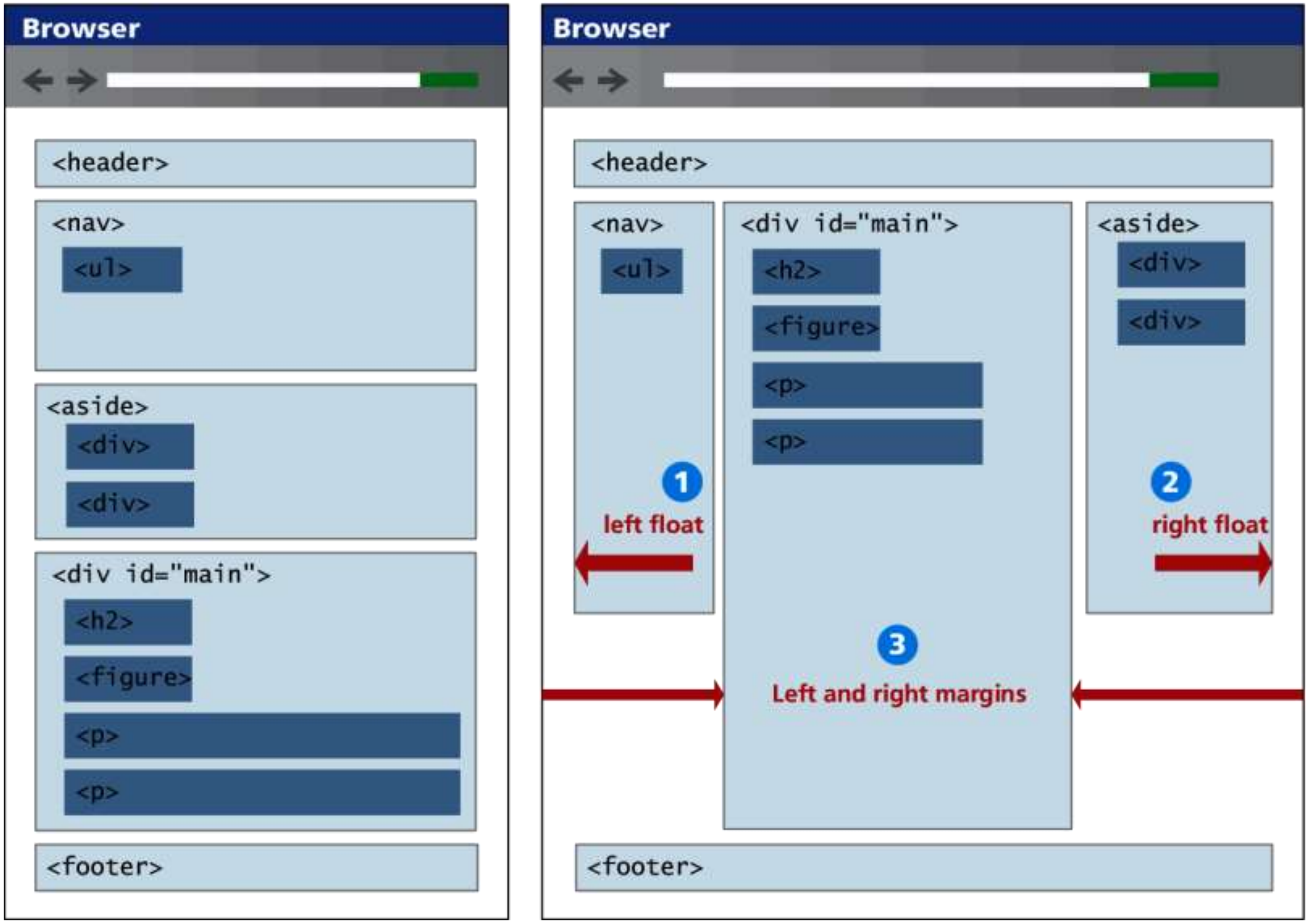
Page Title

British Museum

A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart. I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single stroke at the present moment; and yet I feel that I never was a greater artist than now.

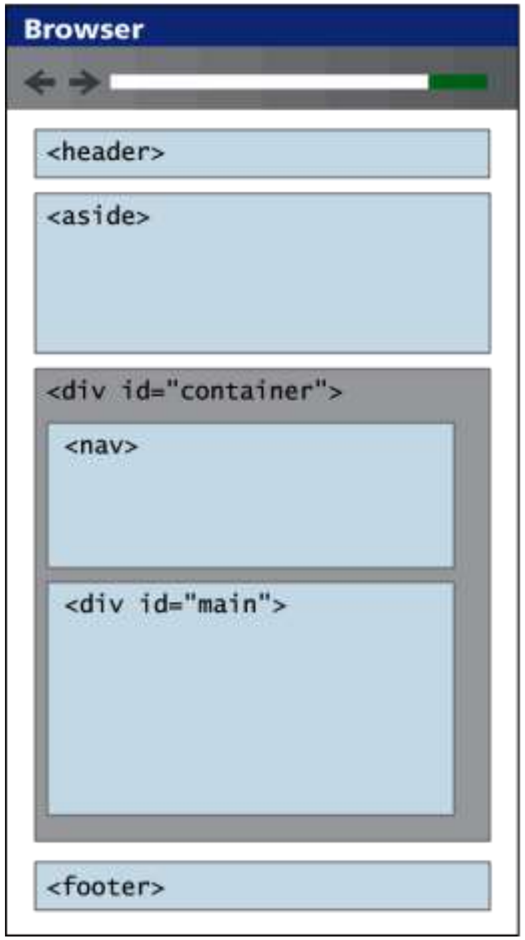
```
div#main {
   …
   margin-left: 220px;
}
```

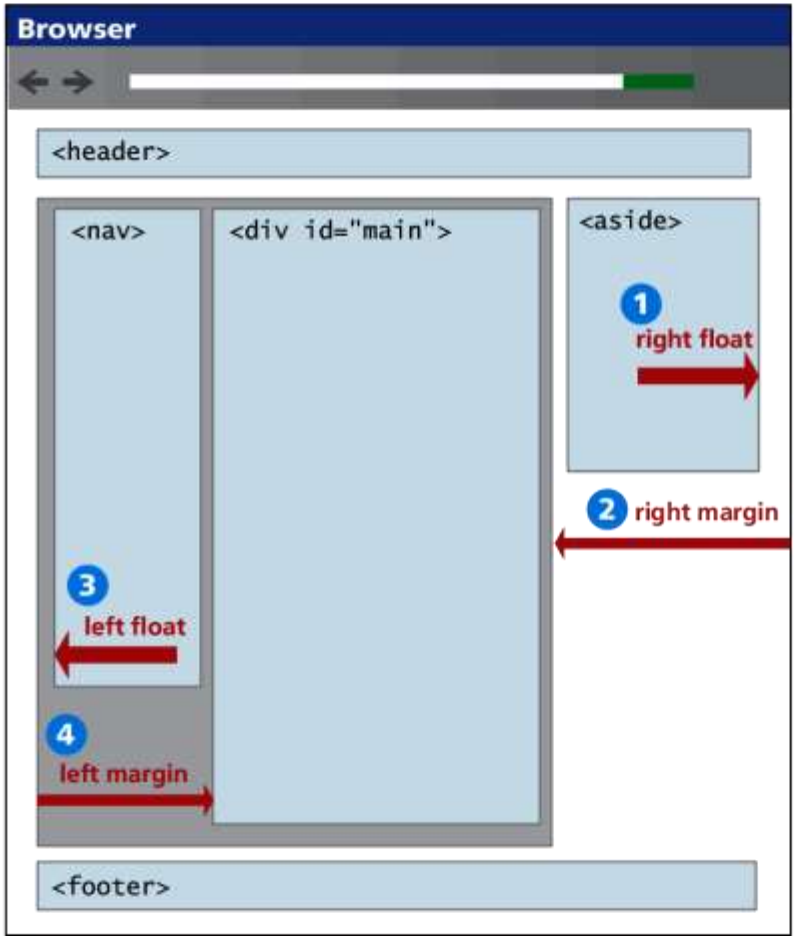
## Using Floats for Columns



## Using Positioning to Create Columns

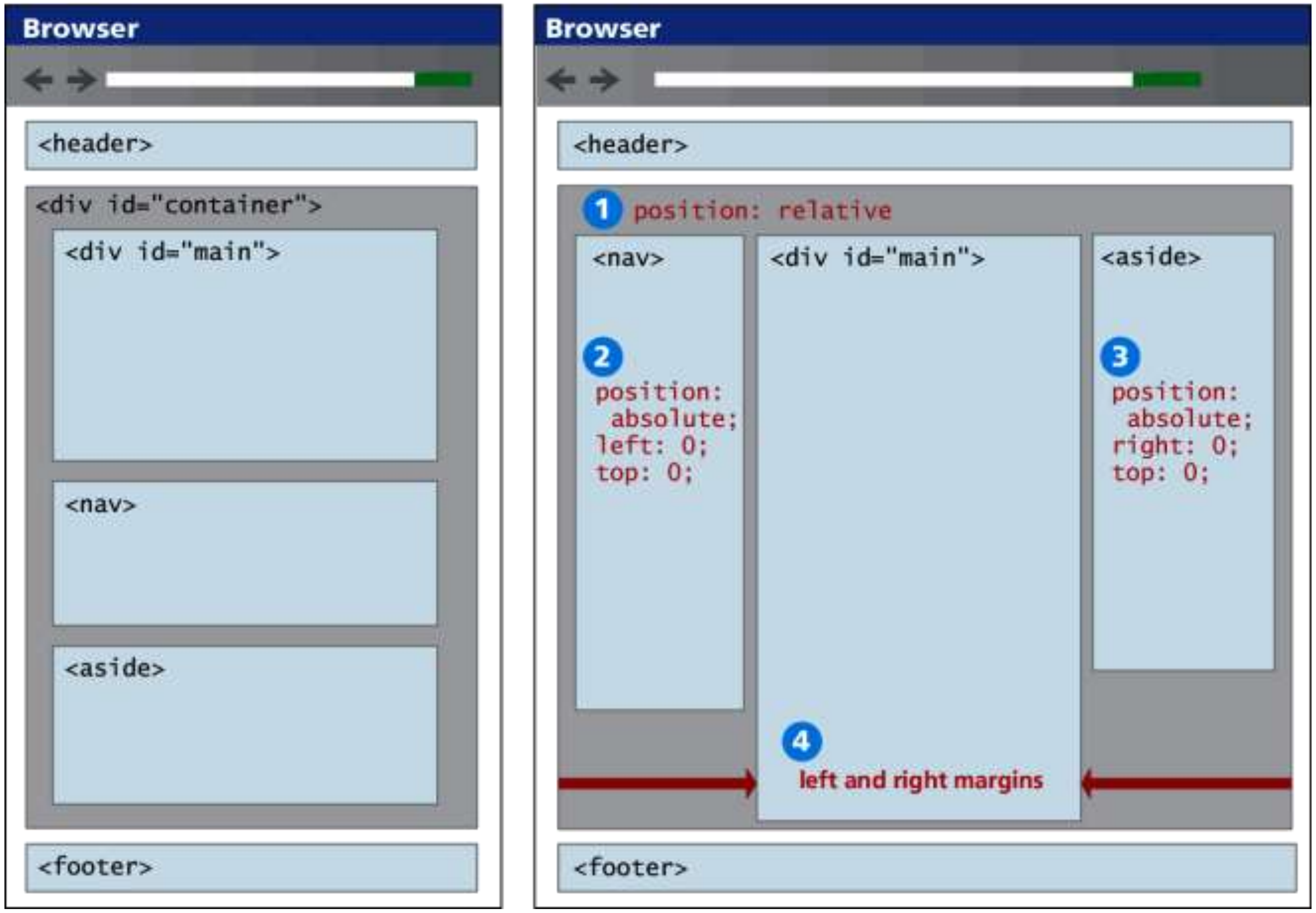
- Positioning can also be used to create a multicolumn layout. Typically, the approach will be to absolute position the elements that were floated in the examples from the previous section





## Positioning discussion

- Notice that with positioning it is easier to construct our source document with content in a more SEO-friendly manner; in this case, the main <div> can be placed first.

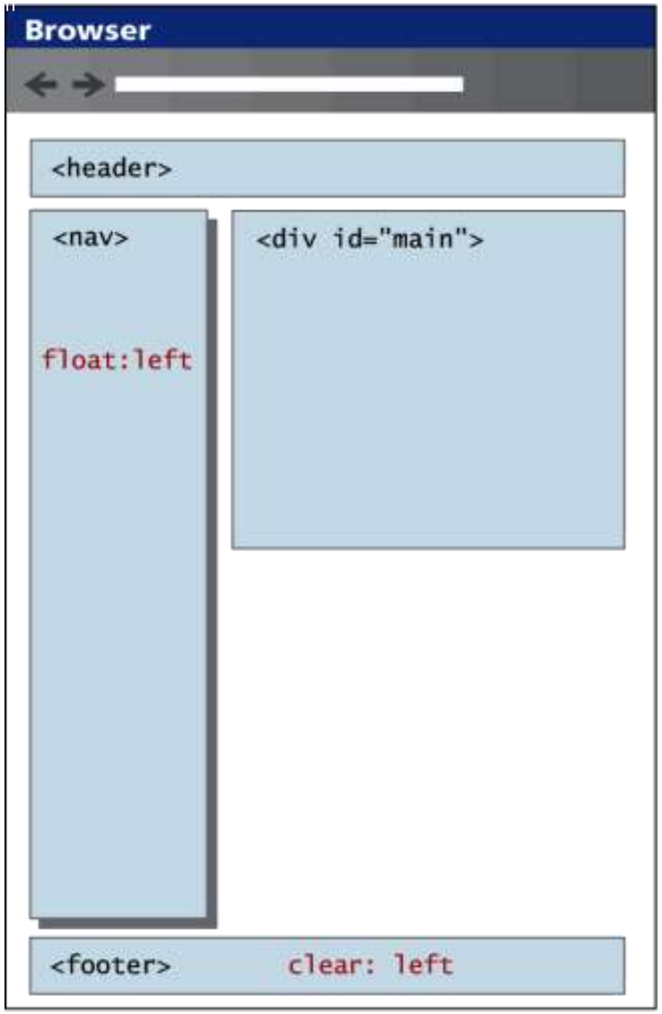


## Positioning discussion

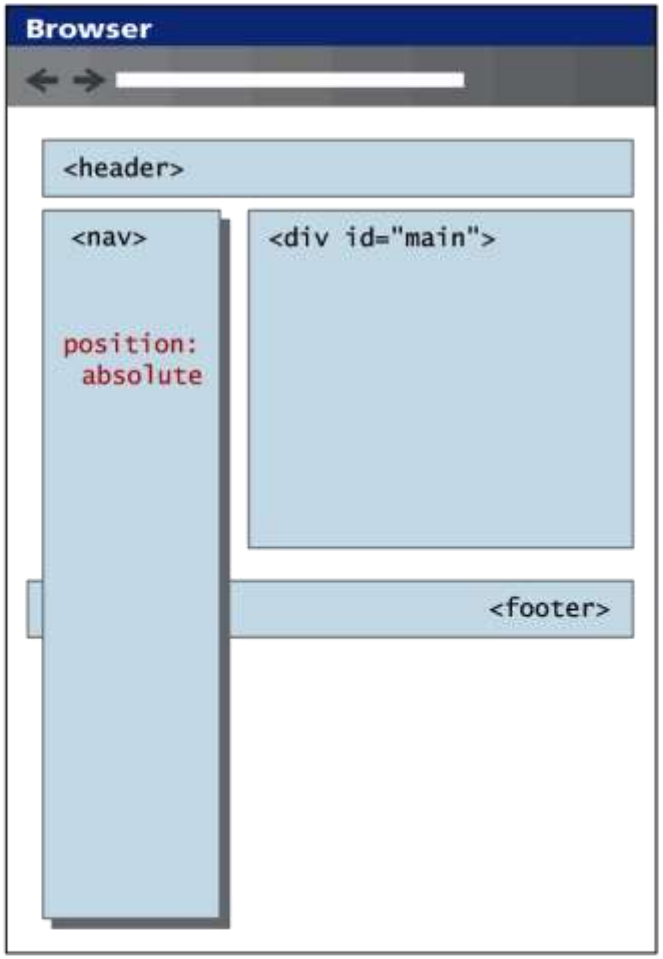
- What would happen if one of the sidebars had a lot of content and was thus quite long?
  - In the floated layout, this would not be a problem at all, because when an item is floated, blank space is left behind.
  - But when an item is positioned, it is removed entirely from normal flow, so subsequent items will have no “knowledge” of the positioned item.

## Positioning Discussion

- Problem illustration



Elements that are floated leave behind space for them in the normal flow. We can also use the `clear` property to ensure later elements are below the floated element.



Absolute positioned elements are taken completely out of normal flow, meaning that the positioned element may overlap subsequent content. The `clear` property will have no effect since it only responds to floated elements.