BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Avalahalli, Doddaballapur Main Road, Bengaluru – 560064

DEPARTMENT OF CSE

17CS71 – Web WEB TECHNOLOGY AND ITS APPLICATIONS

[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2017 - 2018)
SEMESTER – VII

By,

Prof.Muneshwara M S **Prof. Shankar R**

Dept. of CSE, BMSIT&M

HTML 1: Overview

Objectives

1. HTML Defined and its History
2. HTML Syntax
3. Semantic Markup
4. Structure of HTML
5. Quick Tour of HTML
6. HTML Semantic Elements

Brief History of HTML

Did we mention that this will be brief?

- ARPANET of the late 1960s
- jump quickly to the first public specification of the HTML by Tim Berners-Lee in 1991
- HTML's codification by the World-Wide Web Consortium (better known as the **W3C**) in 1997.

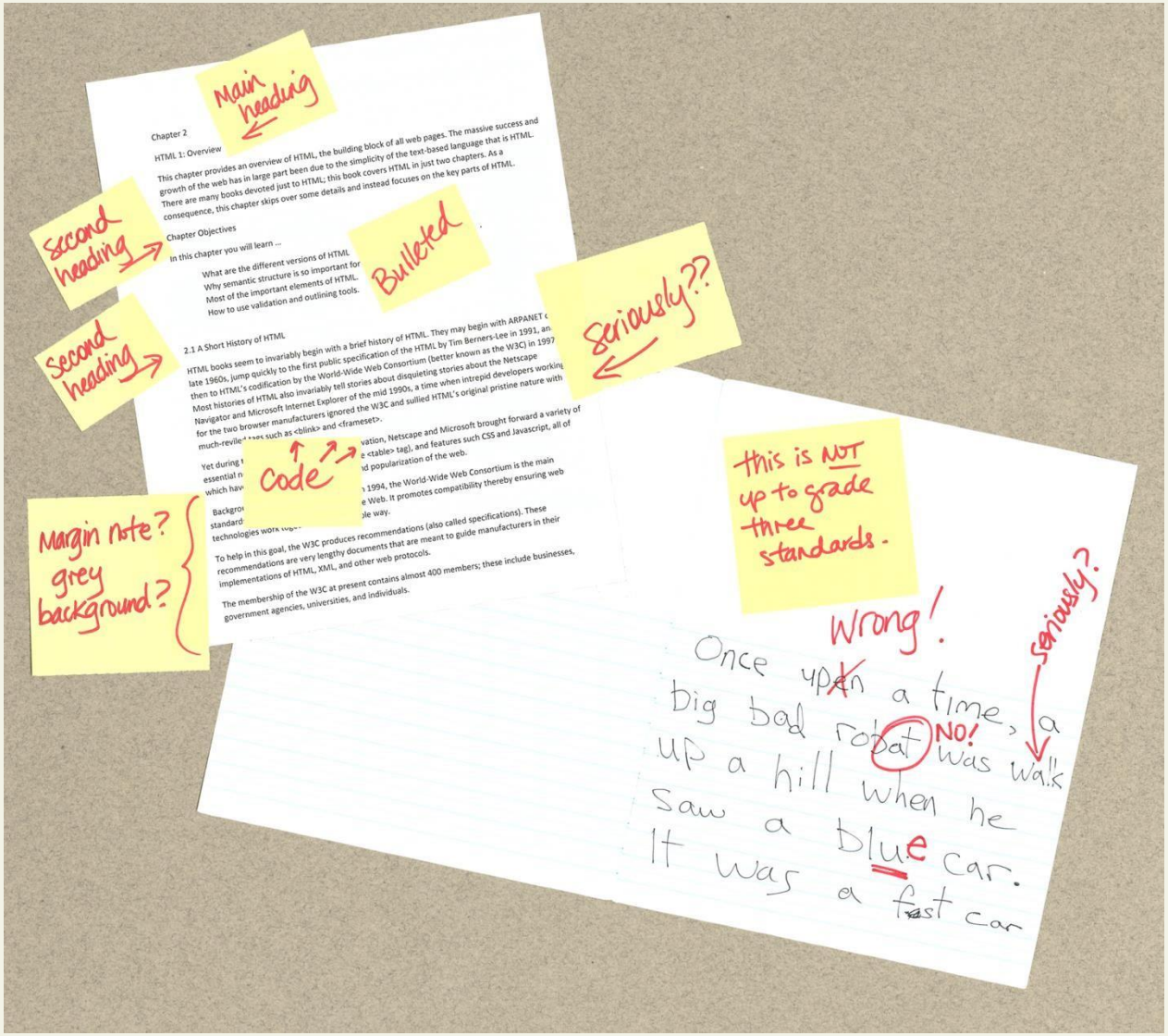
HTML Syntax

What is a markup language?

HTML is defined as a **markup language**.

- A markup language is simply a way of annotating a document in such a way to make the annotations distinct from the text being annotated.
- The term comes from the days of print, when editors would write instructions on manuscript pages that might be revision instructions to the author or copy editor.

Sample adhoc markup



Markup

What is itagain?

At its simplest, **markup** is a way to indicate *information about the content*

- This "information about content" in HTML is implemented via **tags** (aka elements).
- The markup in the previous slide consists of the red text and the various circles and arrows on the one page, and the little yellow sticky notes on the other.
- HTML does the same thing but uses textual tags.

What is the W3C?

Standards

The W3C is the main standards organization for the World Wide Web.

To promotes compatibility the W3C produces **recommendations** (also called **specifications**).

In 1998, the W3C turned its attention to a new specification called XHTML 1.0, which was a version of HTML that used stricter XML (Extensible Markup Language) syntax rules.

XHTML

Partying like it's1999

The goal of XHTML with its strict rules was to make page rendering more predictable by forcing web authors to create web pages without syntax errors.

XHTML

You too can be strict

The XML-based syntax rules for XHTML are pretty easy to follow.

The main rules are:

- lowercase tag names,
- attributes always within quotes,
- and all elements must have a closing element (or be self-closing).

XHTML

Two versions

To help web authors, two versions of XHTML were created:

XHTML 1.0 Strict and **XHTML 1.0 Transitional**.

- The **strict** version was meant to be rendered by a browser using the strict syntax rules and tag support described by the W3C XHTML 1.0 Strict specification.
- The **transitional** recommendation is a more forgiving flavor of XHTML, and was meant to act as a temporary transition to the eventual global adoption of XHTML Strict.

Standards Movement

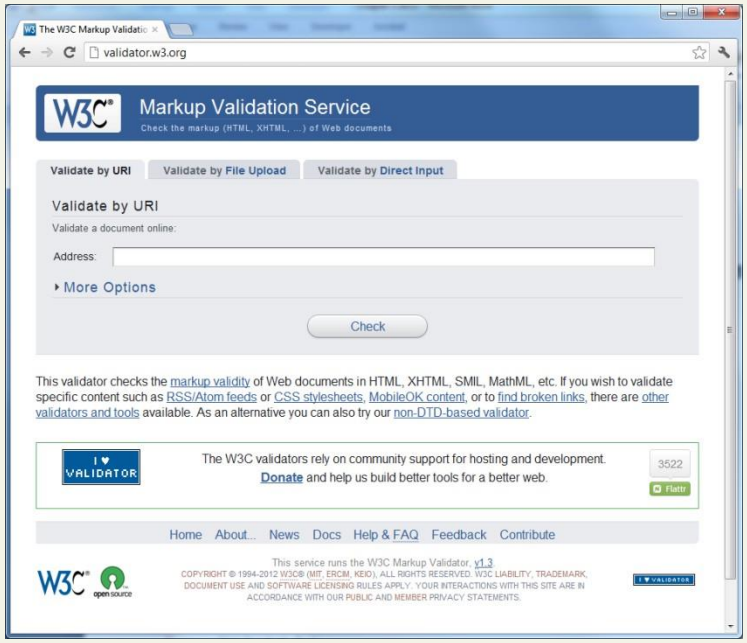
Following a standard is a good thing

During much of the 2000s, the focus in the professional web development community was on standards: that is, on limiting oneself to the W3C specification for XHTML.

Validators

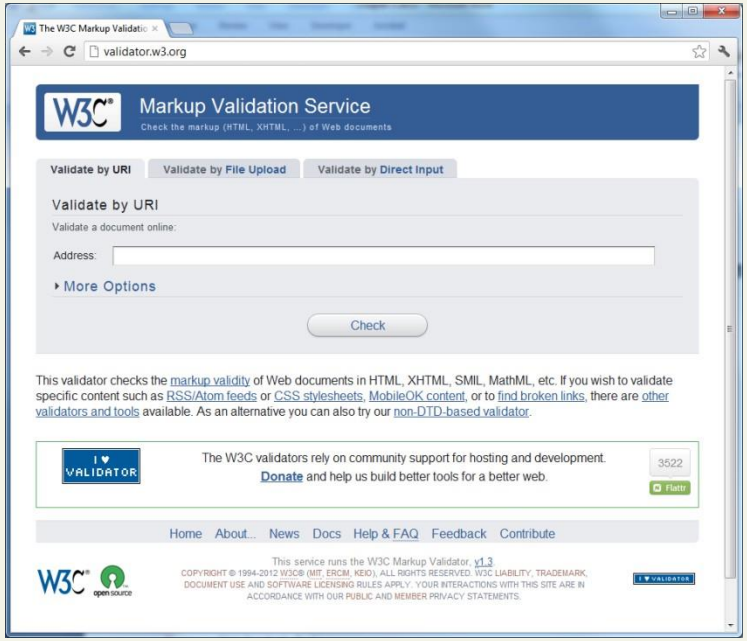
How to ensure your pages follow a standard

A key part of the standards movement in the web development community of the 2000s was the use of **HTML Validators** as a means of verifying that a web page's markup followed the rules for XHTML transitional or strict.



How about an example

Only if you have an internet connection



Open a web browser to the W3C validator and find a few websites to test.

Type the URL into the bar, and you can check if the home page is valid against various standards (or auto-detect)

XHTML 2.0 and WHATWG

Where did it go?

In the mid 2000s, XHTML 2.0 proposed a revolutionary and substantial change to HTML.

- backwards compatibility with HTML and XHTML 1.0 was dropped.
- Browsers would become significantly less forgiving of invalid markup.

At around the same time, a group of developers at Opera and Mozilla formed the **WHATWG** (Web Hypertext Application Technology Working Group) group within the W3C.

This group was not convinced that the W3C's embrace of XML and its abandonment of backwards-compatibility was the best way forward for the web.

HTML5

The new hotness

By 2009, the W3C stopped work on XHTML 2.0 and instead adopted the work done by WHATWG and named it HTML5.

HTML5

Three main aims

There are three main aims to HTML5:

- Specify unambiguously how browsers should deal with invalid markup.
- Provide an open, non-proprietary programming framework (via Javascript) for creating rich web applications.
- Be backwards compatible with the existing web.

HTML5

It evolves

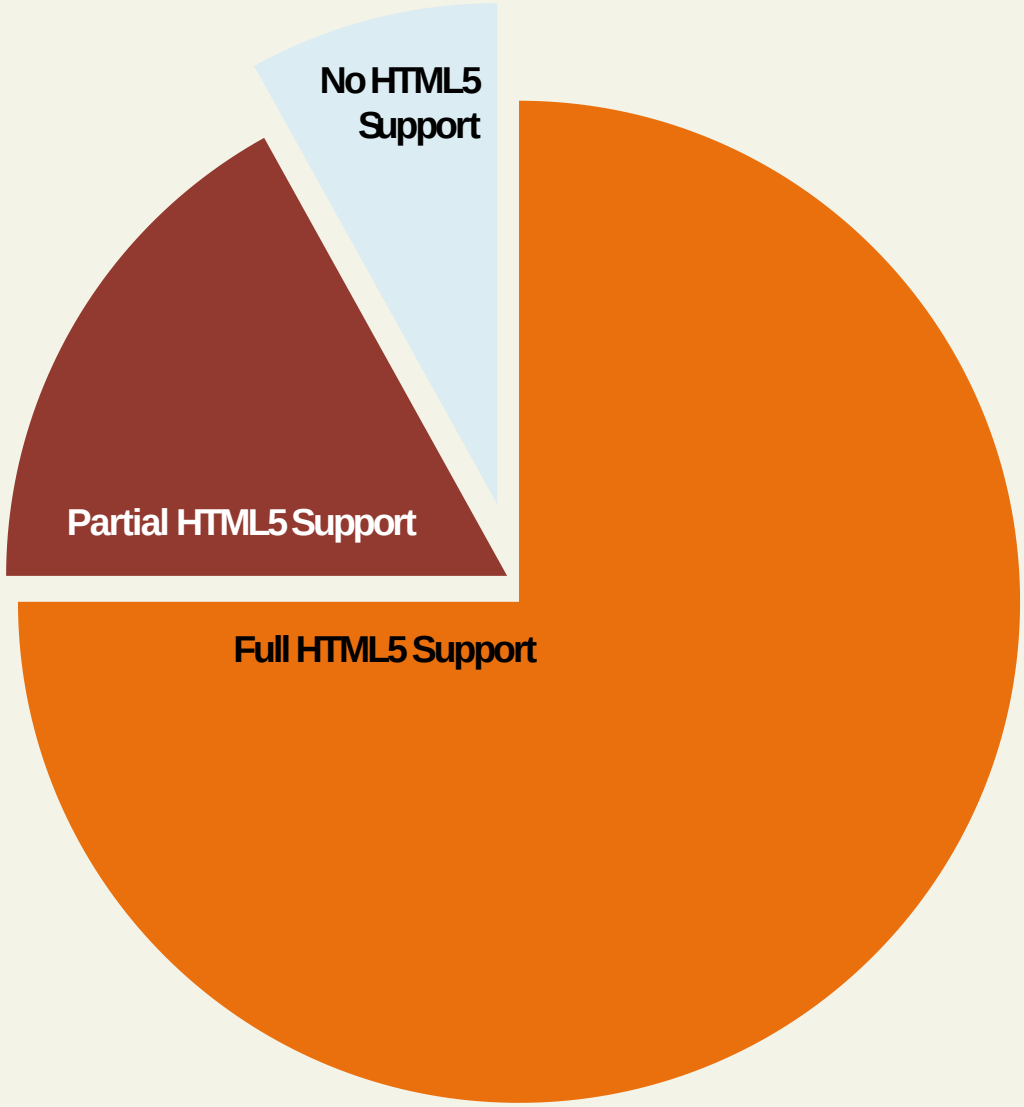
While parts of the HTML5 are still being finalized, all of the major browser manufacturers have at least partially embraced HTML5.

Certainly not all browsers and all versions support every feature of HTML5.

This is in fact by design. HTML in HTML5 is now a living language: that is, it is a language that evolves and develops over time.

As such, every browser will support a gradually increasing subset of HTML5 capabilities

HTML5 Support in Browsers



Section 2 of 6

HTML SYNTAX

Elements and Attributes

More syntax

HTML documents are composed of textual content and HTML elements.

An **HTML element** can contain text, other elements, or be empty. It is identified in the HTML document by tags.

HTML elements can also contain attributes. An **HTML attribute** is a name=value pair that provides more information about the HTML element.

In XHTML, attribute values had to be enclosed in quotes; in HTML5, the quotes are optional.

What HTML lets you do

- Insert images using the **<img>** tag
- Create links with the **<a>** tag
- Create lists with the **<ul>, ** and **<li>** tags
- Create headings with **<H1>, <H2>, …, <H6>**
- Define metatdata with **<meta>** tag
- And much more…

Elements and Attributes

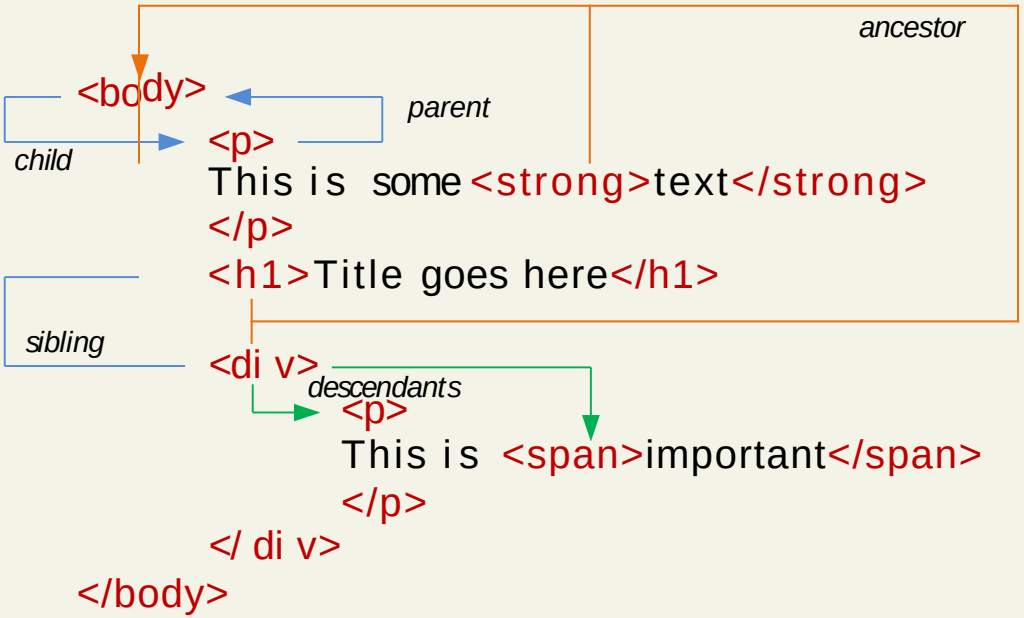
Nesting HTML elements

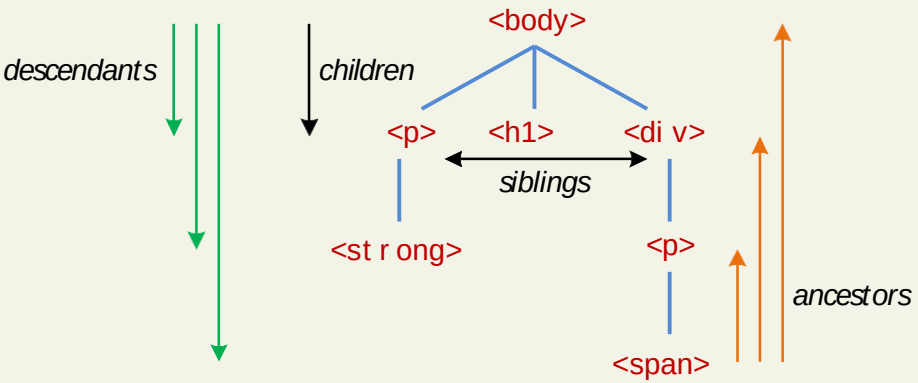
Often an HTML element will contain other HTML elements.

In such a case, the container element is said to be a parent of the contained, or child, element.

Any elements contained within the child are said to be **descendents** of the parent element; likewise, any given child element, may have a variety of **ancestors**.

Hierarchy of elements



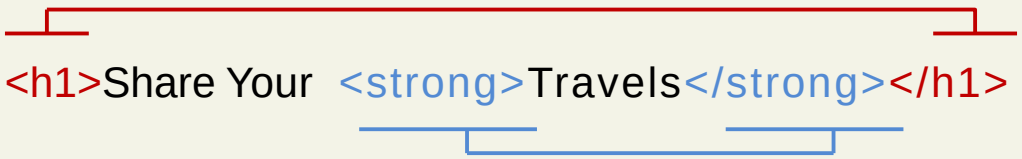


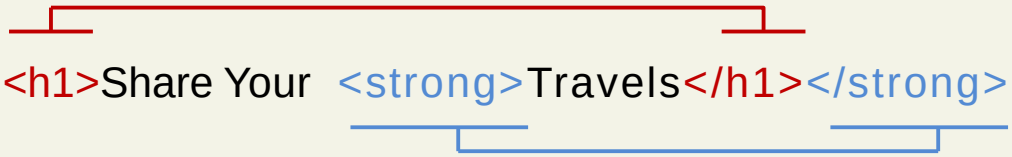
Nesting HTML elements

In order to properly construct a hierarchy of elements, your browser expects each HTML nested element to be properly nested.

That is, a child's ending tag must occur before its parent's ending tag.

Correct Nesting





Incorrect Nesting

Section 3 of 6

SEMANTIC MARKUP

Semantic Markup

What does itmean?

Over the past decade, a strong and broad consensus has grown around the belief that HTML documents should **only** focus on the structure of the document.

Information about how the content should look when it is displayed in the browser is best left to CSS (Cascading Style Sheets).

Semantic Markup

As a consequence, beginning HTML authors are often counseled to create **semantic HTML** documents.

That is, an HTML document should not describe how to visually present content, but only describe its content's structural semantics or meaning.

Structure

Structure is a vital way of communicating information in paper and electronic documents.

All of the tags that we will examine in this presentation are used to describe the basic structural information in a document, such as articles, headings, lists, paragraphs, links, images, navigation, footers, and so on.

Semantic Markup

Its advantages

Eliminating presentation-oriented markup and writing semantic HTML markup has a variety of important advantages:

Maintainability. Semantic markup is easier to update and change than web pages that contain a great deal of presentation markup.

Faster. Semantic web pages are typically quicker to author and faster to download.

Accessibility. Visiting a web page using voice reading software can be a very frustrating experience if the site does not use semantic markup.

Search engine optimization. Semantic markup provides better instructions for search engines: it tells them what things are important content on the site.

Section 4 of 6

STRUCTURE OF HTML

Simplest HTML document

```
<!DOCTYPE html>
<title>A Very Small Document</title>
<p>This is a simple document with not much content</p>
```

The <title> element (Item 1) is used to provide a broad description of the content. The title is not displayed within the browser window. Instead, the title is typically displayed by the browser in its window and/or tab.

A more complete document

1 DOCTYPE

(short for **Document Type Definition**)

Tells the browser (or any other client software that is reading this HTML document) what type of document it is about to process.

Notice that it does not indicate what version of HTML is contained within the document: it only specifies that it contains HTML.

HTML, Head, and Body

HTML5 does not require the use of the <html>, <head>, and <body>.

However, in XHTML they were required, and most web authors continue to use them.

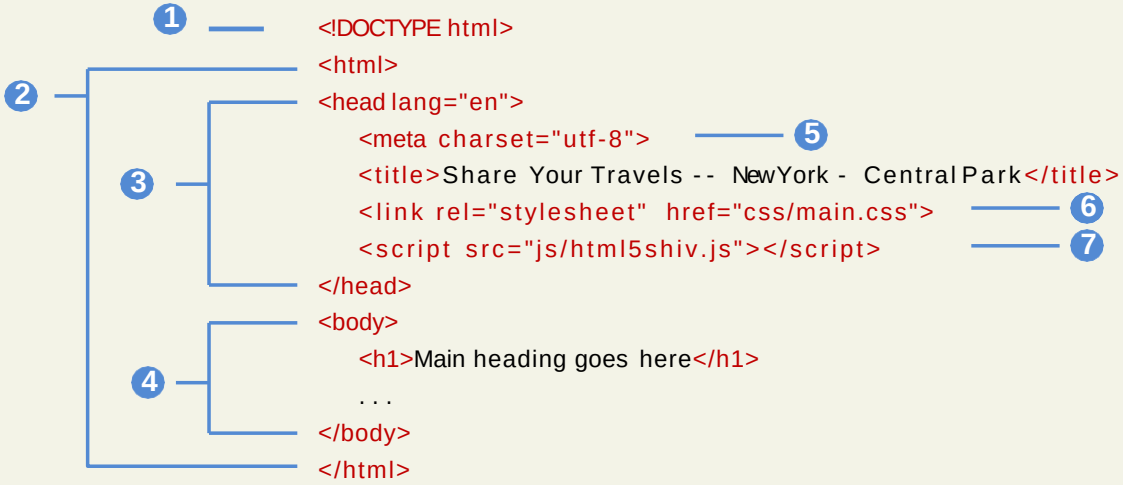
2. The <html> element is sometimes called the **root element** as it contains all the other HTML elements in the document.

Head and Body

HTML pages are divided into two sections: the **head** and the **body**, which correspond to the <head> and <body> elements.

3. The head contains descriptive elements *about* the document

4. The body contains content that will be displayed by the browser.

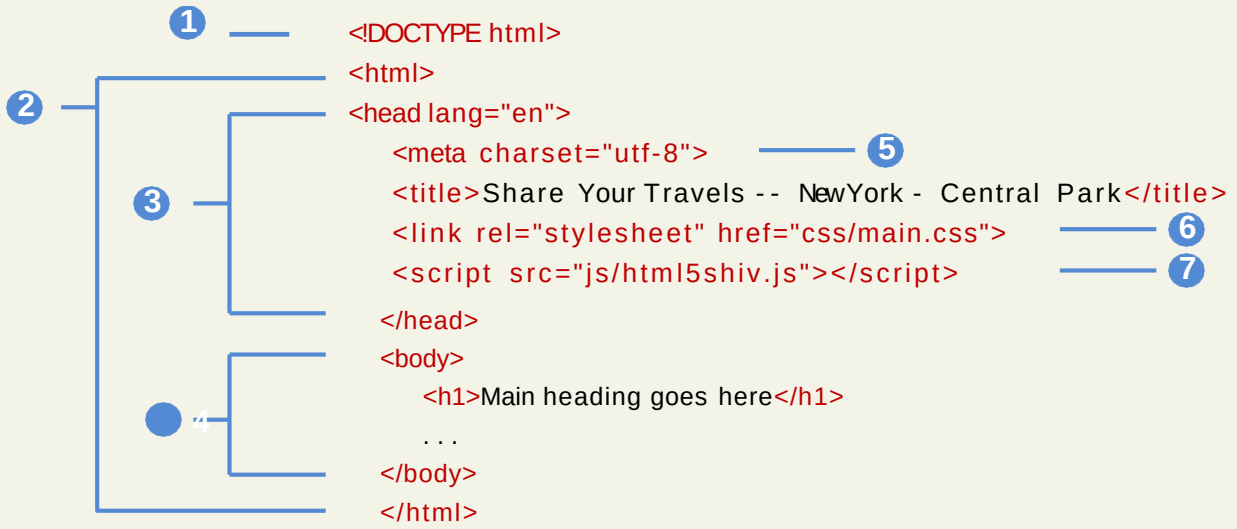


Inside the head

There are no brains

You will notice that the <head> element contains a variety of additional elements.

5. The first of these is the <meta> element. Our example declares that the character encoding for the document is UTF-8.



Inside the head

No brains but metas, styles and javascripts

6. Our example specifies an external CSS style sheet file that is used with this document.

7. It also references an external Javascript file.

Section 5 of 6

QUICK TOUR OF HTML

Why a quick tour?

HTML5 contains many structural and presentation elements – too many to completely cover in this presentation.

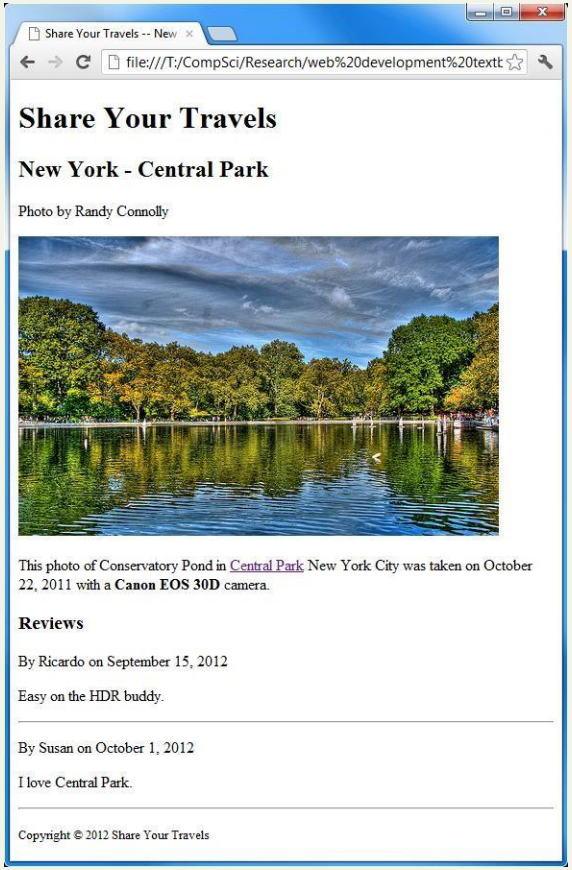
Rather than comprehensively cover all these elements, this presentation will provide a quick overview of the most common elements.

Sample Document

```
<body>
    <h1>Share Your Travels</h1>
    <h2>New York - Central Park</h2>
    <p>Photo by Randy Connolly</p>
    <p>This photo of Conservatory Pond in
        <a href="http://www.centralpark.com/">Central Park</a>
        New York City was taken on October 22, 2011 with a
        <strong>Canon EOS 30D</strong> camera.
    </p>
    <img src="images/central-park.jpg" alt="Central Park" />

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

    <div>
        <p>By Susan on <time>October 1, 2012</time></p>
        <p>I love Central Park.</p>
    </div>
    <p><small>Copyright &amp;copy; 2012 Share Your Travels</small></p>
</body>
```

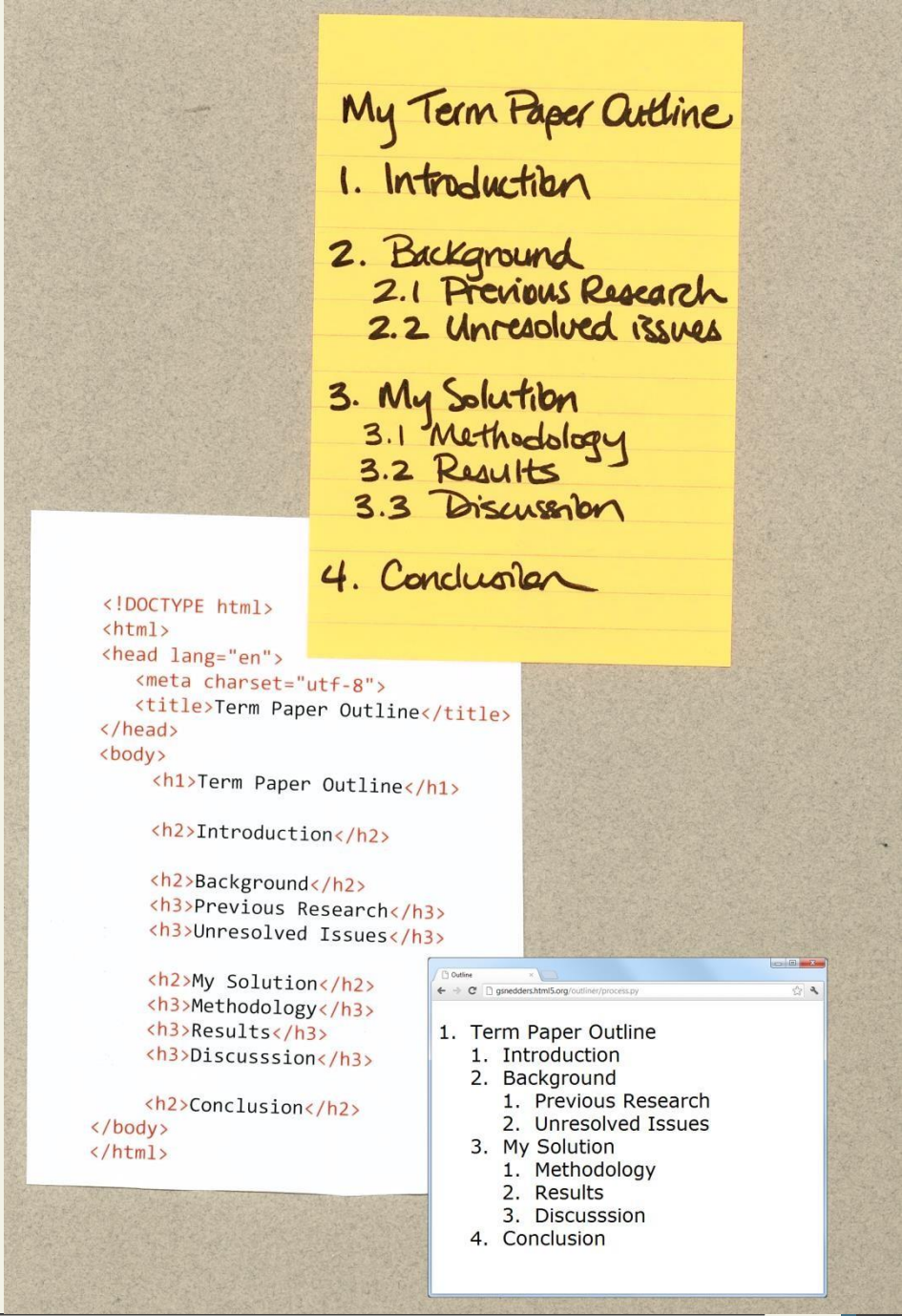


1 Headings

<h1>, <h2>, <h3>, etc

HTML provides six levels of heading (**h1, h2, h3**, …), with the higher heading number indicating a heading of less importance.

Headings are an essential way for document authors use to show their readers the structure of the document.



Headings

The browser has its own default styling for each heading level.

However, these are easily modified and customized via CSS.

Headings

Be semantically accurate

In practice, specify a heading level that is semantically accurate.

Do not choose a heading level because of its default presentation

- e.g., choosing <h3> because you want your text to be bold and 16pt

Rather, choose the heading level because it is appropriate

- e.g., choosing <h3> because it is a third level heading and not a primary or secondary heading

2 Paragraphs

<p>

Paragraphs are the most basic unit of text in an HTML document.

Notice that the **<p>** tag is a container and can contain HTML and other **inline HTML elements**

inline HTML elements refers to HTML elements that do not cause a paragraph break but are part of the regular "flow" of the text.

6 Divisions

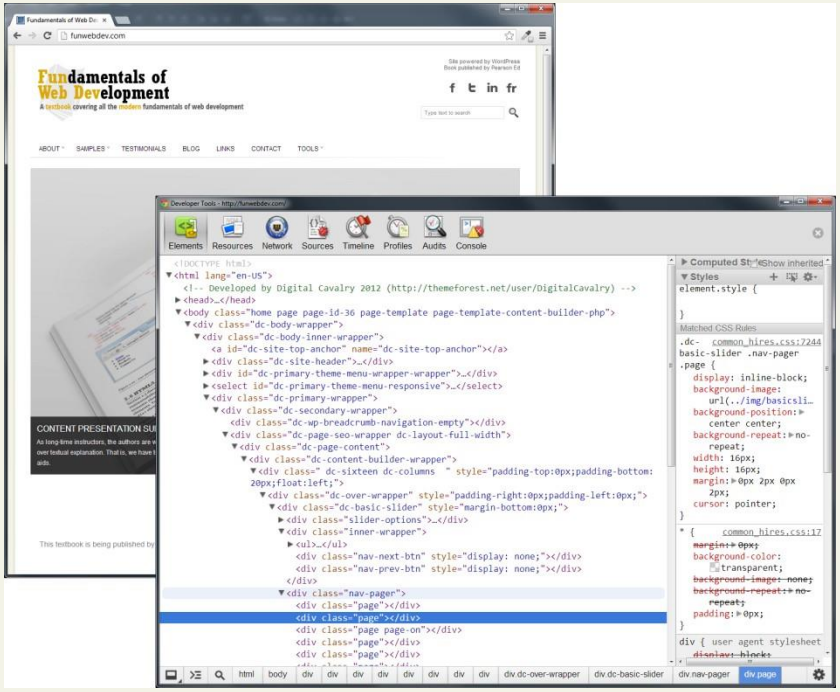
This ** ** tag is also a container element and is used to create a logical grouping of content

- The element has no intrinsic presentation.
- It is frequently used in contemporary CSS-based layouts to mark out sections.

Using div elements

Can you say “div-tastic”

HTML5 has a variety of new semantic elements (which we will examine later) that can be used to reduce somewhat the confusing mass of div within divs within divs that is so typical of contemporary web design.



3 Links

<a>

Links are created using the **<a>** element (the "a" stands for anchor).

A link has two main parts: the destination and the label.

```
<a href="http://www.centralpark.com">Central Park</a>
```

Destination

Label (text)

```
<a href="index.html"><img src="logo.gif" /></a>
```

Label (image)

Types of Links

You can use the anchor element to create a wide range of links:

- Links to external sites (or to individual resources such as images or movies on an external site).
- Links to other pages or resources within the current site.
- Links to other places within the current page.
- Links to particular locations on another page.
- Links that are instructions to the browser to start the user's email program.
- Links that are instructions to the browser to execute a Javascript function.

Different link destinations

Link to external site

```
<a href="http://www.centralpark.com">Central Park</a>
```

Link to resource on external site

```
<a href="http://www.centralpark.com/logo.gif">Central Park</a>
```

Link to another page on same site as this page

```
<a href="index.html">Home</a>
```

Link to another place on the same page

```
<a href="#top">Go to Top of Document</a>
```

Link to specific place on another page

```
<a href="productX.html#reviews">Reviews for product X</a>
```

Link to email

```
<a href="mailto://person@somewhere.com">Someone</a>
```

Link to javascript function

```
<a href="javascript://OpenAnnoyingPopup();">See This</a>
```

Link to telephone (automatically dials the number when user clicks on it using a smartphone browser)

```
<a href="tel:+18009220579">Call toll free (800) 922-0579</a>
```

Link Text

Some guidance … or … don't "Click Here"

Links with the label "Click Here" were once a staple of the web.

Today, such links are frowned upon, since:

- they do not tell users where the link will take them
- as a verb "click" is becoming increasingly inaccurate when one takes into account the growth of mobile browsers.

Instead, textual link labels should be descriptive.

~~"Click here to see the race results"~~

"Race Results" or **"See Race Results".**

URL Absolute Referencing

For external resources

When referencing a page or resource on an external site, a full **absolute reference** is required: that is,

- the protocol (typically, http://),
- the domain name,
- any paths, and then finally
- the file name of the desired resource.

URL Relative Referencing

An essential skill

We also need to be able to successfully reference files within our site.

This requires learning the syntax for so-called **relative referencing**.

When referencing a resource that is on the same server as your HTML document, then you can use briefer relative referencing. If the URL does not include the “http://” then the browser will request the current server for the file.

URL Relative Referencing

If all the resources for the site reside within the same **directory** (also referred to as a **folder**), then you can reference those other resources simply via their filename.

However, most real-world sites contain too many files to put them all within a single directory.

For these situations, a relative pathname is required along with the filename.

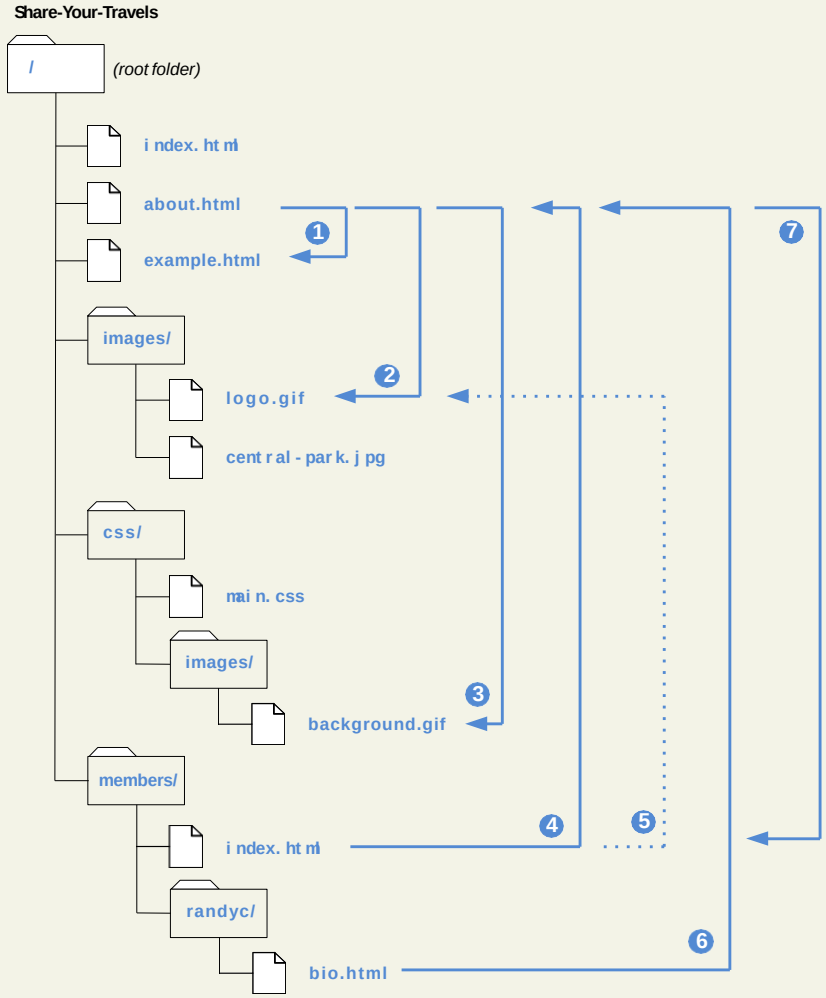
The **pathname** tells the browser where to locate the file on the server.

Pathnames

Pathnames on the web follow Unix conventions.

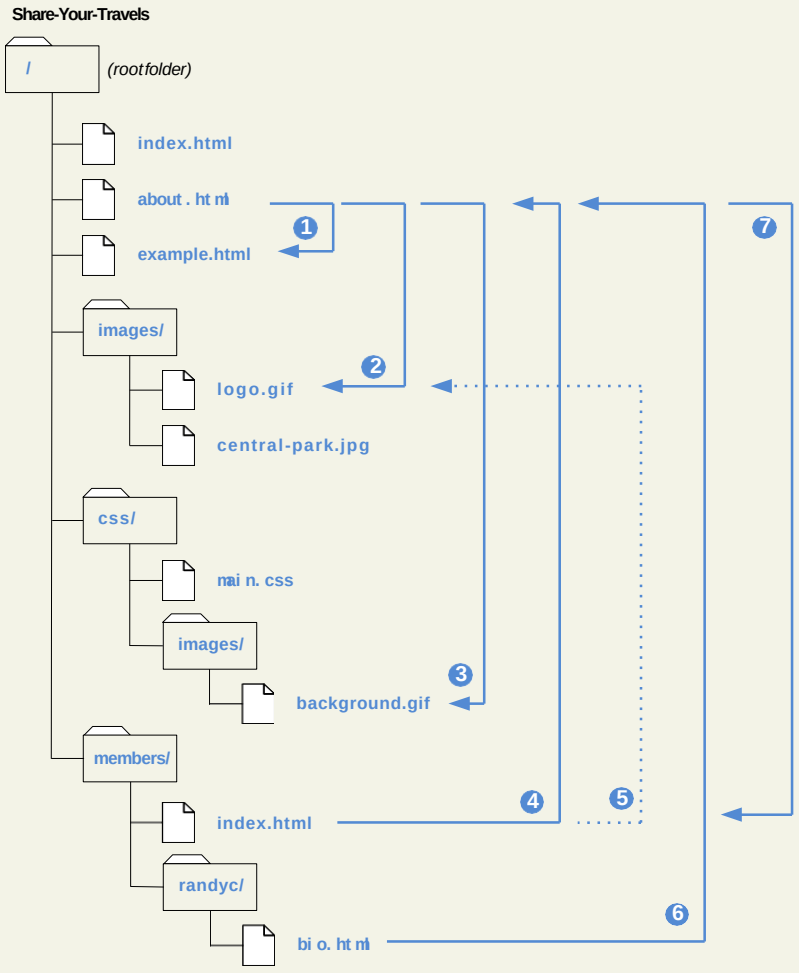
- Forward slashes (“/”) are used to separate directory names from each other and from file names.
- Double-periods (“..”) are used to reference a directory “above” the current one in the directory tree.

URL Relative Referencing



	Relative Link Type	Example
1	**Same Directory** To link to a file within the same folder, simply use the file name.	To link to example.html from about.html (in Figure 2.18), use: <a href="example.html">
2	**Child Directory** To link to a file within a subdirectory, use the name of the subdirectory and a slash before the file name.	To link to logo.gif from about.html, use: <a href="images/logo.gif">
3	**Grandchild/Descendant Directory** To link to a file that is multiple subdirectories *below* the current one, construct the full path by including each subdirectory name (separated by slashes) before the file name.	To link to background.gif from about.html, use: <a href="css/images/background.gif">
4	**Parent/Ancestor Directory** Use "../" to reference a folder *above* the current one. If trying to reference a file several levels above the current one, simply string together multiple "../".	To link to about.html from index.html in members, use: <a href="../about.html"> To link to about.html from bio.html, use: <a href="../../about.html">

URL Relative Referencing



5 Sibling Directory

Use "../" to move up to the appropriate level, and then use the same technique as for child or grandchild directories.

To link to logo.gif from index.html in members, use:

```
<a href="../images/about.html">
```

To link to background.gif from bio.html, use:

```
<a href="../../css/images/background.gif">
```

6 Root Reference

An alternative approach for ancestor and sibling references is to use the so-called **root reference** approach. In this approach, begin the reference with the root reference (the "/") and then use the same technique as for child or grandchild directories. **Note that these will only work on the server! That is, they will not work when you test it out on your local machine.**

To link to about.html from bio.html, use:

```
<a href="/about.html">
```

To link to background.gif from bio.html, use:

```
<a href="/images/background.gif">
```

7 Default Document

Web servers allow references to directory names without file names. In such a case, the web server will serve the default document, which is usually a file called index.html (apache) or default.html (IIS). **Again, this will only generally work on the web server.**

To link to index.html in members from about.html, use either:

```
<a href="members">
```

Or

```
<a href="/members">
```

Inline Text Elements

Do not disrupt the flow

Inline elements do not disrupt the flow of text (i.e., cause a line break).

HTML5 defines over 30 of these elements.

e.g., <a>,
, ,

Images

While the **<img>** tag is the oldest method for displaying an image, it is not the only way.

For purely decorative images, such as background gradients and patterns, logos, border art, and so on, it makes semantic sense to keep such images out of the markup and in CSS where they more rightly belong.

But when the images are content, such as in the images in a gallery or the image of a product in a product details page, then the <img> tag is the semantically appropriate approach.

Images

Specifies the URL of the image to display (note: uses standard relative referencing)

Text in title attribute will be displayed in a popup tool tip when user moves mouse over image.

```
<img src="images/central-park.jpg" alt="Central Park" title="Central Park" width="80" height="40" />
```

Text in alt attribute provides a brief description of image's content for users who are unable to see it.

Specifies the width and height of image in pixels.

Lists

HTML provides three types of lists

Unordered lists. Collections of items in no particular order; these are by default rendered by the browser as a bulleted list.

Ordered lists. Collections of items that have a set order; these are by default rendered by the browser as a numbered list.

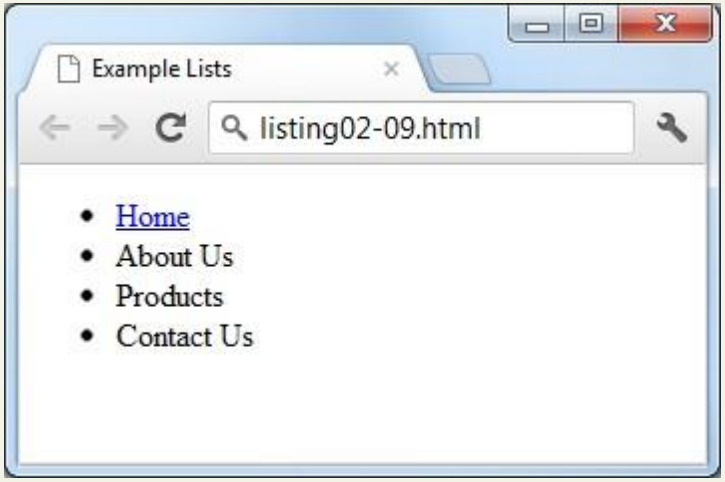
Definition lists. Collection of name and definition pairs. These tend to be used infrequently. Perhaps the most common example would be a FAQ list.

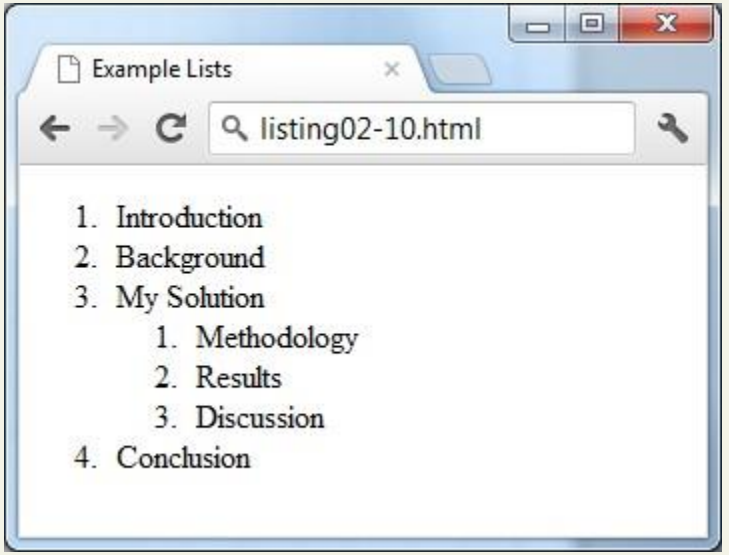
Lists

Notice that the list item element can contain other HTML elements

```
<ul>
    <li><a href="index.html">Home</a></li>
    <li>About Us</li>
    <li>Products</li>
    <li>Contact Us</li>
</ul>
```

```
<ol>
    <li>Introduction</li>
    <li>Background</li>
    <li>My Solution</li>
    <li>
        <ol>
            <li>Methodology</li>
            <li>Results</li>
            <li>Discussion</li>
        </ol>
    </li>
    <li>Conclusion</li>
</ol>
```





Character Entities

These are special characters for symbols for which there is either no way easy way to type in via a keyboard (such as the copyright symbol or accented characters) or which have a reserved meaning in HTML (for instance the “<” or “>” symbols).

They can be used in an HTML document by using the entity name or the entity number.

e.g.,   and ©

Section 6 of 6

HTML SEMANTIC ELEMENTS

HTML5 Semantic Elements

Why are theyneeded?

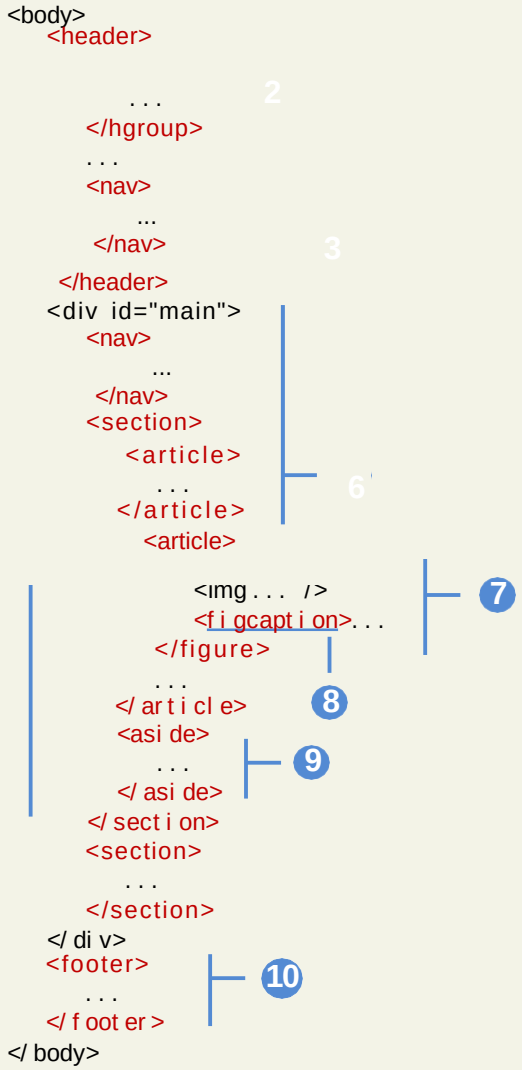
One substantial problem with modern, pre-HTML5 semantic markup:

> most complex web sites are absolutely packed solid with elements.

Unfortunately, all these elements can make the resulting markup confusing and hard to modify.

Developers typically try to bring some sense and order to the chaos by using id or class names that provide some clue as to their meaning.

XHTML versus HTML5



Header and Footer

<header> <footer>

Most web site pages have a recognizable header and footer section.

Typically the **header** contains

- the site logo
- title (and perhaps additional subtitles or taglines)
- horizontal navigation links, and
- perhaps one or two horizontal banners.

Header and Footer

<header> <footer>

The typical footer contains less important material, such as

- smaller text versions of the navigation,
- copyright notices,
- information about the site's privacy policy, and
- perhaps twitter feeds or links to other social sites.

Header and Footer

Both the HTML5 <header> and <footer> element can be used not only for *page* headers and footers, they can also be used for header and footer elements within other HTML5 containers, such as <article> or <section>.

```
<header>
    <img src="logo.gif" alt="logo" />
    <h1>Fundamentals of Web Development</h1>
    ...
</header>
<article>
    <header>
        <h2>HTML5 Semantic Structure Elements </h2>
          <p>By <em>Randy Connolly</em></p>
          <p><time>September 30, 2012</time></p>
    </header>
    ...
</article>
```

2 Heading Groups

<hgroup>

The <hgroup> element can be used to group related headings together within one container.

```
<header>
    <hgroup>
        <h1>Chapter Two: HTML 1</h1>
        <h2>An Introduction</h2>
    </hgroup>
</header>
<article>
    <hgroup>
        <h2>HTML5 Semantic Structure Elements </h2>
        <h3>Overview</h3>
    </hgroup>
</article>
```

3 Navigation

<nav>

The **<nav>** element represents a section of a page that contains links to other pages or to other parts within the same page.

Like the other new HTML5 semantic elements, the browser does not apply any special presentation to the <nav> element.

The <nav> element was intended to be used for major navigation blocks, presumably the global and secondary navigation systems.

Navigation

```
<header>
      <img src="logo.gif" alt="logo" />
      <h1>Fundamentals of Web Development</h1>
      <nav role="navigation">
            <ul>
                  <li><a href="index.html">Home</a></li>
                  <li><a href="about.html">About Us</a></li>
                  <li><a href="browse.html">Browse</a></li>
            </ul>
      </nav>
</header>
```

Articles and Sections

<article> <section>

The **<article>** element represents a section of content that forms an independent part of a document or site; for example, a magazine or newspaper article, or a blog entry.

The **<section>** element represents a section of a document, typically with a title or heading.

Articles and Sections

According to the W3C, **<section>** is a much broader element, while the **<article>** element is to be used for blocks of content that could potentially be read or consumed independently of the other content on the page.

Sections versus Divs

How to decide which to use

The WHATWG specification warns readers that the <section> element is **not** a generic container element. HTML already has the element for such uses.

When an element is needed only for styling purposes or as a convenience for scripting, it makes sense to use the element instead.

Another way to help you decide whether or not to use the <section> element is to ask yourself if it is appropriate for the element's contents to be listed explicitly in the document's outline.

If so, then use a <section>; otherwise use a .

Figure and FigureCaptions

<figure> <figcaption>

The W3C Recommendation indicates that the <figure> element can be used not just for images but for any type of *essential* content that could be moved to a different location in the page or document and the rest of the document would still make sense.

Figure and FigureCaptions

Note however …

The **<figure>** element should **not** be used to wrap every image.

For instance, it makes no sense to wrap the site logo or non-essential images such as banner ads and graphical embellishments within <figure> elements.

Instead, only use the <figure> element for circumstances where the image (or other content) has a caption and where the figure is essential to the content but its position on the page is relatively unimportant.

Figure and FigureCaptions

Figure could be moved to a different location in document …
But it has to exist in the document (i.e., the figure isn't optional)

```
<p>This photo was taken on October 22, 2011 with a Canon EOS 30D camera.</p>
<figure>
  <img src="images/central-park.jpg" alt="Central Park" /><br/>
  <figcaption>Conservatory Pond in Central Park</figcaption>
</figure>
<p>
It was a wonderfully beautiful autumn Sunday, with strong sunlight and
expressive clouds. I was very fortunate that myone day in New York was
blessed with such weather!
</p>
```

Aside

<aside>

The **<aside>** element is similar to the **<figure>** element in that it is used for marking up content that is separate from the main content on the page.

But while the **<figure>** element was used to indicate important information whose location on the page is somewhat unimportant, the **<aside>** element "represents a section of a page that consists of content that is tangentially related to the content around the aside element."

The **<aside>** element could thus be used for sidebars, pull quotes, groups of advertising images, or any other grouping of non-essential elements.

What You've Learned

1. HTML Defined and its History
2. HTML Syntax
3. Semantic Markup
4. Structure of HTML
5. Quick Tour of HTML
6. HTML Semantic Elements

CSS 1: Introduction

Chapter 3

Objectives

1. What is CSS?
2. CSS Syntax
3. Location of Styles
4. Selectors
5. The Cascade: How Styles Interact
6. The Box Model
7. CSS Text Styling

Section 1 of 7

WHAT IS CSS?

What is CSS?

You be styling soon

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

With CSS, we can assign

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

What is CSS?

You be styling soon

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

CSS can be added directly to any HTML element (via the style attribute), within the **<head>** element, or, most commonly, in a separate text file that contains only CSS.

Benefits of CSS

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

- The degree of formatting control in CSS is significantly better than that provided in HTML.
- Web sites become significantly more maintainable because all formatting can be centralized into one, or a small handful, of CSS files.
- CSS-driven sites are more accessible.
- A site built using a centralized set of CSS files for all presentation will also be quicker to download because each individual HTML file will contain less markup.
- CSS can be used to adopt a page for different output mediums.

CSS Versions

Let's just say there's more than 1

- W3C published the CSS Level 1 Recommendation in 1996.
- A year later, the CSS Level 2 Recommendation (also more succinctly labeled simply as CSS2) was published.
- Even though work began over a decade ago, an updated version of the Level 2 Recommendation, CSS2.1, did not become an official W3C Recommendation until June 2011.
- And to complicate matters even more, all through the last decade (and to the present day as well), during the same time the CSS2.1 standard was being worked on, a different group at the W3C was working on a CSS3 draft.

Browser Adoption

Insert obligatory snide comment about Internet Explorer 6 here

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

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

CSS has a reputation for being a somewhat frustrating language.

- this reputation is well deserved!

Section 2 of 7

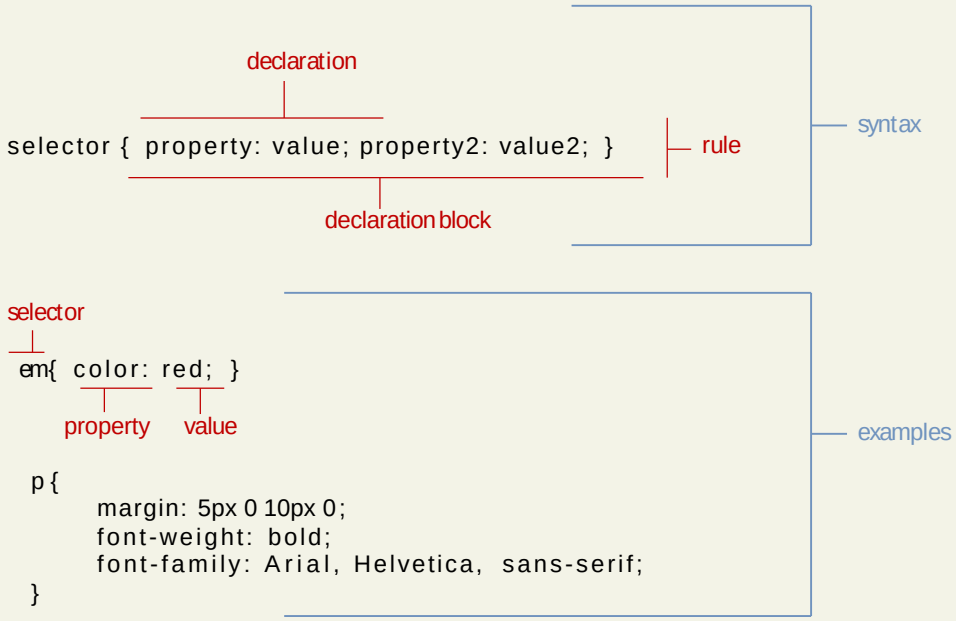
CSS SYNTAX

CSS Syntax

Rules, properties, values, declarations

A CSS document consists of one or more **style rules**.

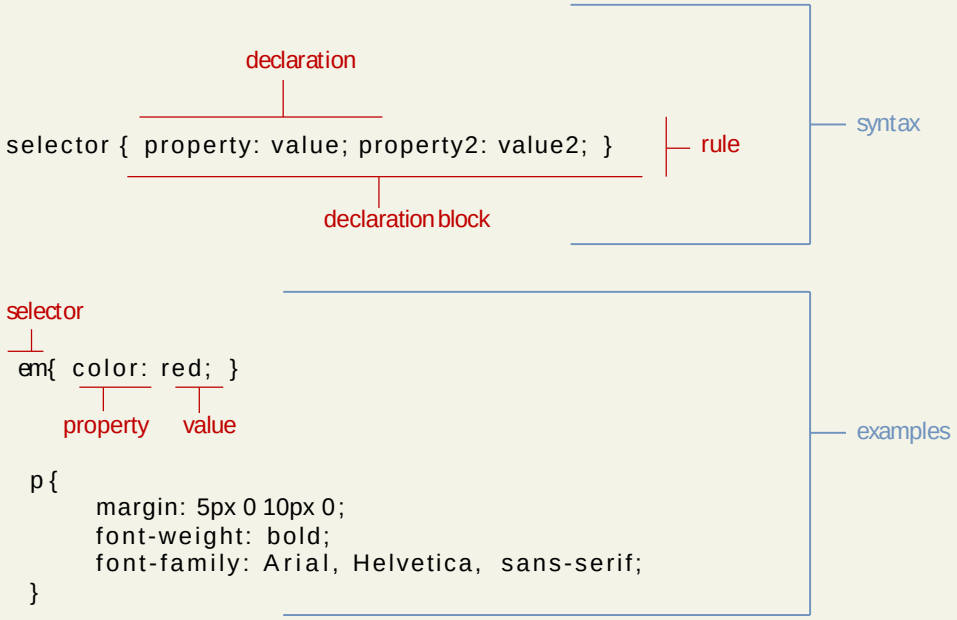
A rule consists of a selector that identifies the HTML element or elements that will be affected, followed by a series of **property** and **value** pairs (each pair is also called a **declaration**).



Declaration Blocks

The series of declarations is also called the **declaration block**.

- A declaration block can be together on a single line, or spread across multiple lines.
- The browser ignores white space
- Each declaration is terminated with a semicolon.



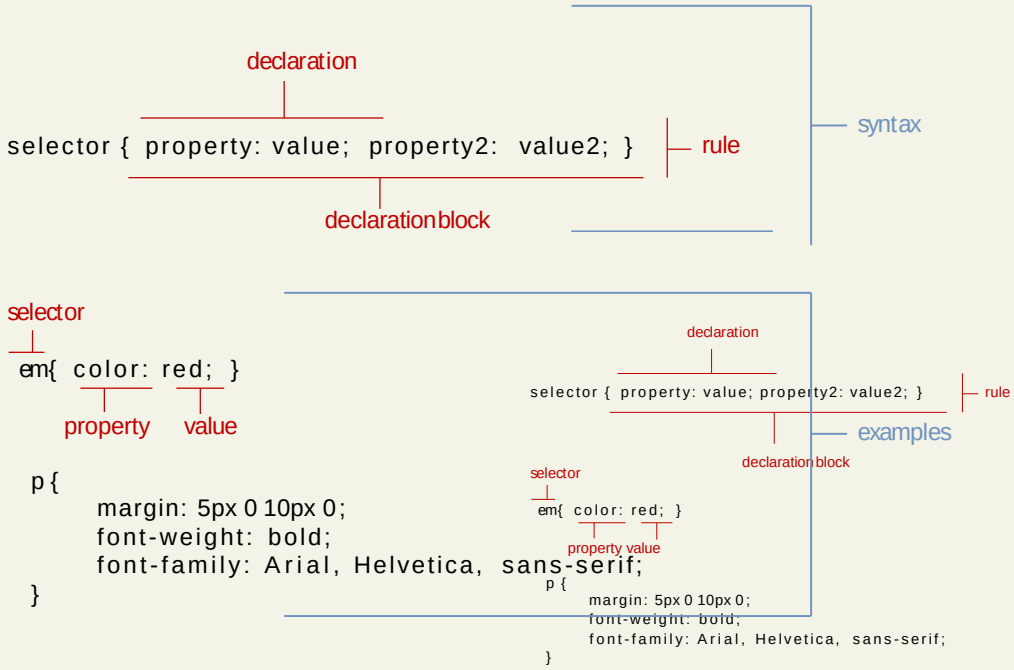
Selectors

Which elements

Every CSS rule begins with a **selector**.

The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule.

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.



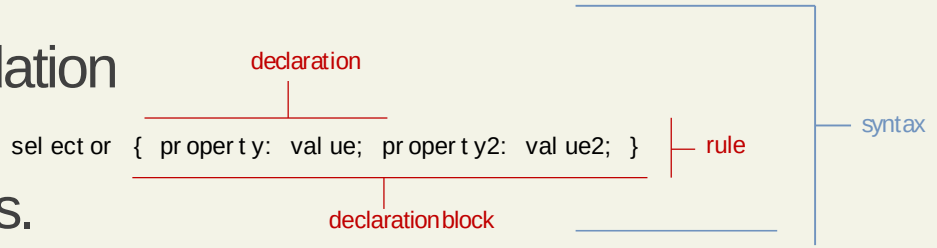
Properties

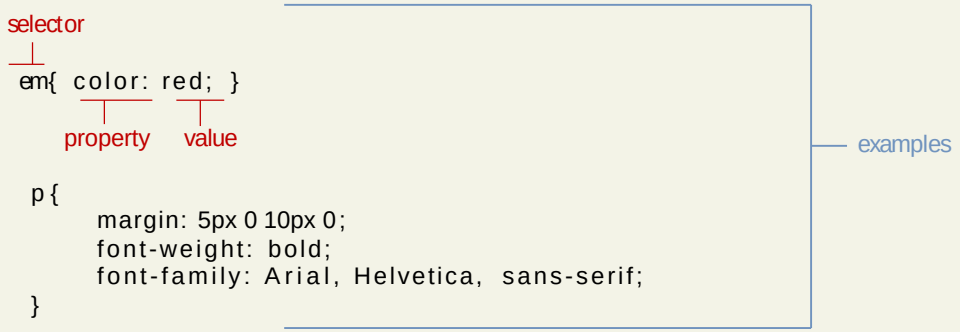
Which style properties of the selected elements

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

These property names are predefined by the CSS standard.

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





Properties

Common CSS properties

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

Properties

Common CSS properties continued.

Property Type	Property
Spacing	padding padding-bottom, padding-left, padding-right, padding-top margin margin-bottom, margin-left, margin-right, margin-top
Sizing	height max- height max-width min-height min-width width
Layout	bottom, left, right, top clear display float overflo w position visibility z-index
Lists	list-style list-style- image list- style-type

Values

What style value for the properties

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

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

Color Values

CSS supports a variety of different ways of describing color

Method	Description	Example
Name	Use one of 17 standard color names. CSS3 has 140 standard names.	color: red; color: hotpink; /* CSS3 only */
RGB	Uses three different numbers between 0 and 255 to describe the Red, Green, and Blue values for the color.	color: rgb(255,0,0); color: rgb(255,105,180);
Hexadecimal	Uses a six-digit hexadecimal number to describe the red, green, and blue value of the color; each of the three RGB values is between 0 and FF (which is 255 in decimal). Notice that the hexadecimal number is preceded by a hash or pound symbol (#).	color: #FF0000; color: #FF69B4;
RGBa	Allows you to add an alpha, or transparency, value. This allows a background color or image to "show through" the color. Transparency is a value between 0.0 (fully transparent) and 1.0 (fully opaque).	color: rgb(255,0,0, 0.5);
HSL	Allows you to specify a color using Hue Saturation and Light values. This is available only in CSS3. HSLA is also available as well.	color: hsl(0,100%,100%); color: hsl(330,59%,100%);

Units of Measurement

There are multiple ways of specifying a unit of measurement in CSS

Some of these are **relative units**, in that they are based on the value of something else, such as the size of a parent element.

Others are **absolute units**, in that they have a real-world size.

> Unless you are defining a style sheet for printing, it is recommended to avoid using absolute units.
>
> Pixels are perhaps the one popular exception (though as we shall see later there are also good reasons for avoiding the pixel unit).

Relative Units

Unit	Description	Type
px	Pixel. In CSS2this is a relative measure, while in CSS3it is absolute (1/96 of an inch).	Relative (CSS2) Absolute (CSS3)
em	Equal to the computed value of the font-size property of the element on which it is used. When used for font sizes, the em unit is in relation to the font size of the parent.	Relative
%	A measure that is always relative to another value. The precise meaning of %varies depending upon which property it is being used.	Relative
ex	A rarely used relative measure that expresses size in relation to the x-height of an element's font.	Relative
ch	Another rarely used relative measure; this one expresses size in relation to the width of the zero ("0") character of an element's font.	Relative (CSS3 only)
rem	Stands for root em, which is the font size of the root element. Unlike em, which may be different for each element, the rem is constant throughout the document.	Relative (CSS3 only)
vw, vh	Stands for viewport width and viewport height. Both are percentage values (between 0 and 100) of the viewport (browser window). This allows an item to change size when the viewport is resized.	Relative (CSS3 only)

Absolute Units

Unit	Description	Type
in	Inches	Absolute
cm	Centimeters	Absolute
mm	Millimeters	Absolute
pt	Points (equal to 1/72 of aninch)	Absolute
pc	Pica (equal to 1/6 of aninch)	Absolute

Comments in CSS

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

/* comment goes here */

Section 3 of 7

LOCATION OF STYLES

Actually there are three…

Different types of stylesheet

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

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

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

Style Locations

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

CSSstyle rules can be located in three different locations.

- Inline
- Embedded
- External

You can combine all 3!

Inline Styles

Style rules placed within an HTML element via the style attribute

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

LISTING 3.1 Internal styles example

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

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

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

Embedded StyleSheet

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

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

LISTING 3.2 Embedded styles example

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

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

External Style Sheet

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

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

LISTING 3.3 Referencing an external style sheet

This is by far the most common place to locate style rules because it provides the best maintainability.

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

Section 4 of 7

SELECTORS

Selectors

Things that make your life easier

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

CSS selectors allow you to select

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

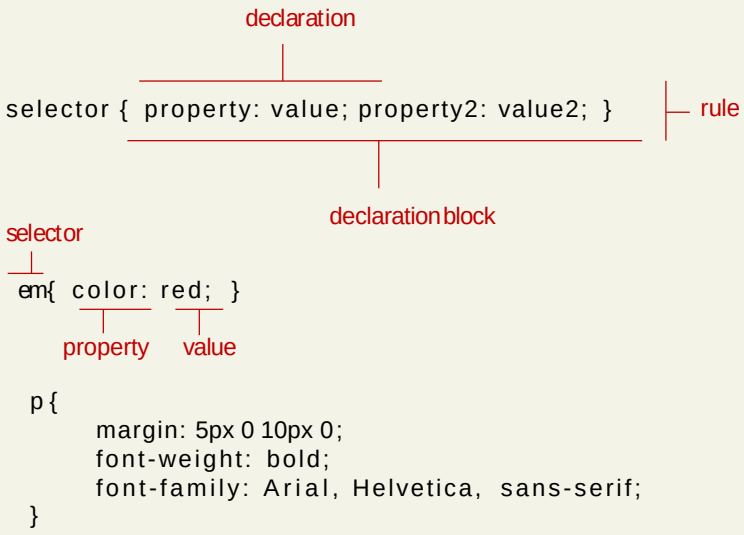
There are a number of different selector types.

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.




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

Grouped Selectors

Selecting multiple things

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

LISTING 3.4 Sample grouped selector

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

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

Reset

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

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

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

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

Class Selectors

Simultaneously target different HTML elements

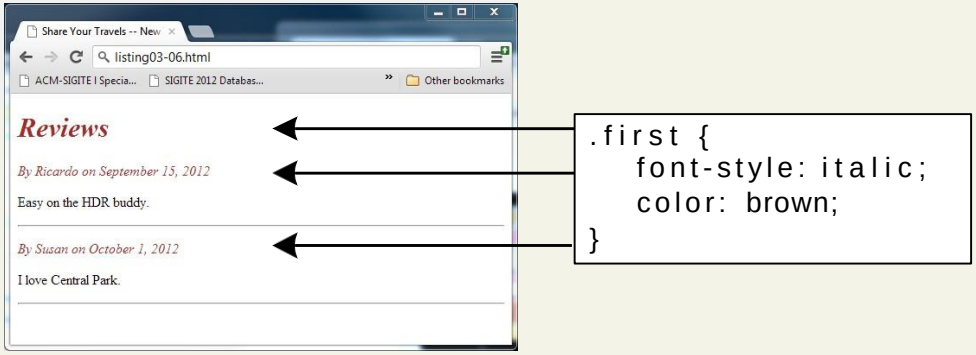
A **class selector** allows you to simultaneously target different HTML elements regardless of their position in the document tree.

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

Class Selectors

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

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



Id Selectors

Target a specific element by its id attribute

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

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

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

Id Selectors

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

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


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

Id versus Class Selectors

How to decide

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

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

Attribute Selectors

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

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

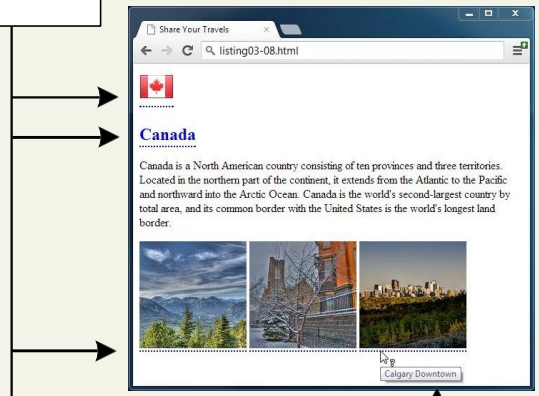
This can be a very powerful technique, but because of uneven support by some of the browsers, not all web authors have used them.

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

Attribute Selectors

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

```
<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>
```



Pseudo Selectors

Select something that does not exist explicitly as an element

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

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

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

Pseudo Selectors

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

LISTING 3.8 Styling a link using pseudo-class selectors

Contextual Selectors

Select elements based on their ancestors, descendants, or siblings

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

That is, it selects elements based on their context or their relation to other elements in the document tree.

Contextual Selectors

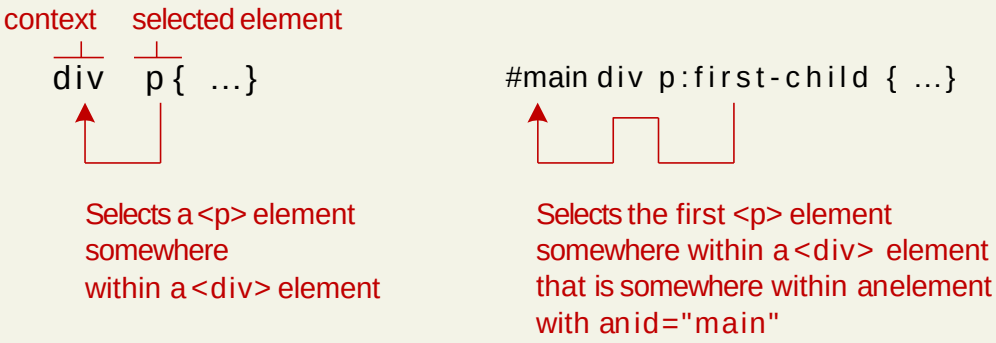
Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element	div p Selects a <p> element that is contained somewhere within a element. That is, the <p> can be any descendant, not just a child.
Child	A specified element that is a direct child of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

Descendant Selector

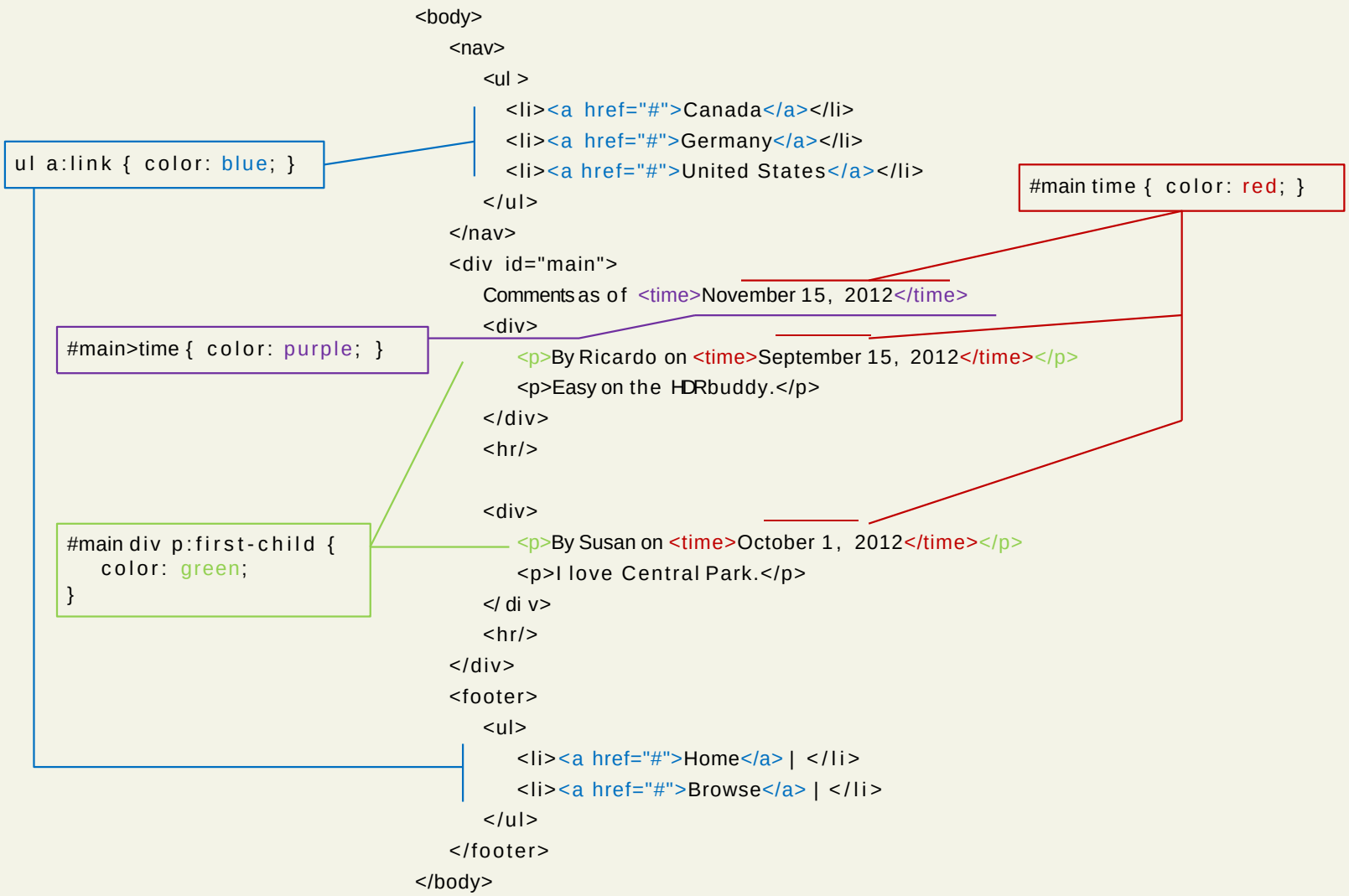
Selects all elements that are contained within another element

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

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



Contextual Selectors in Action



Section 5 of 7

THE CASCADE: HOW STYLES INTERACT

Why Conflict Happens

In CSS that is

Because

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

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

Cascade

How conflicting rules are handled in CSS

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

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

The downward movement of water down a cascade is meant to be analogous to how a given style rule will continue to take precedence with child elements.

Cascade Principles

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

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

Inheritance

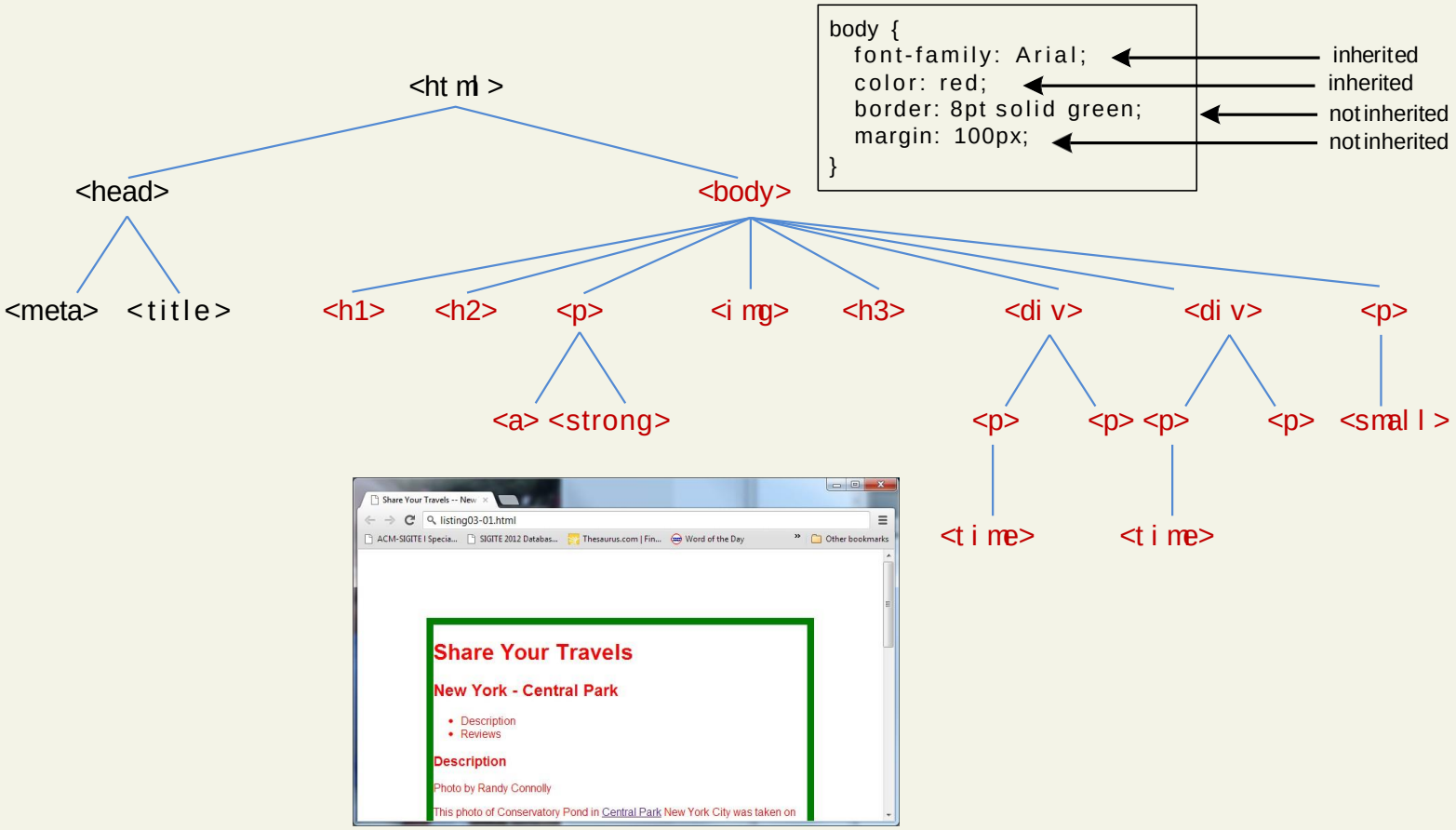
Cascade Principle #1

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

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

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

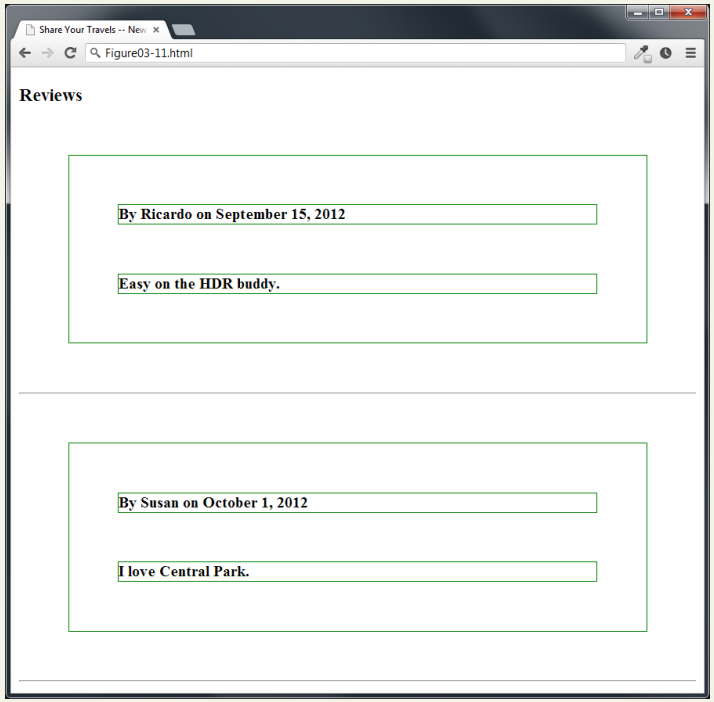
Inheritance



Inheritance

How to force inheritance

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

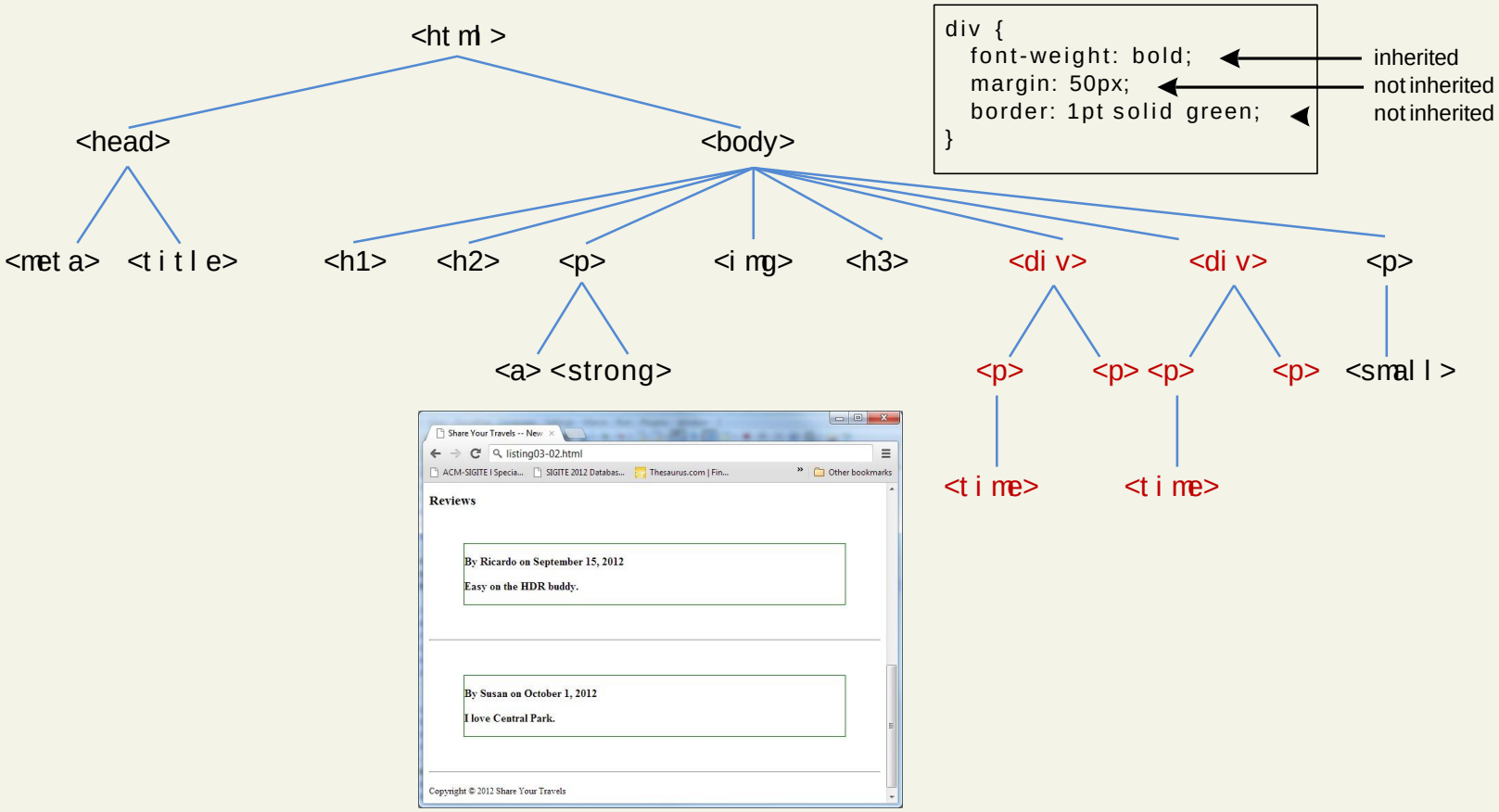



```
div {
  font-weight: bold;
  margin: 50px;
  border: 1pt solid green;
}
p {
  border: inherit;
  margin: inherit;
}
```

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

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

Inheritance



Specificity

Cascade Principle #2

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

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

Specificity

How it works

The way that specificity works in the browser is that the browser assigns a weight to each style rule.

When several rules apply, the one with the greatest weight takes precedence.

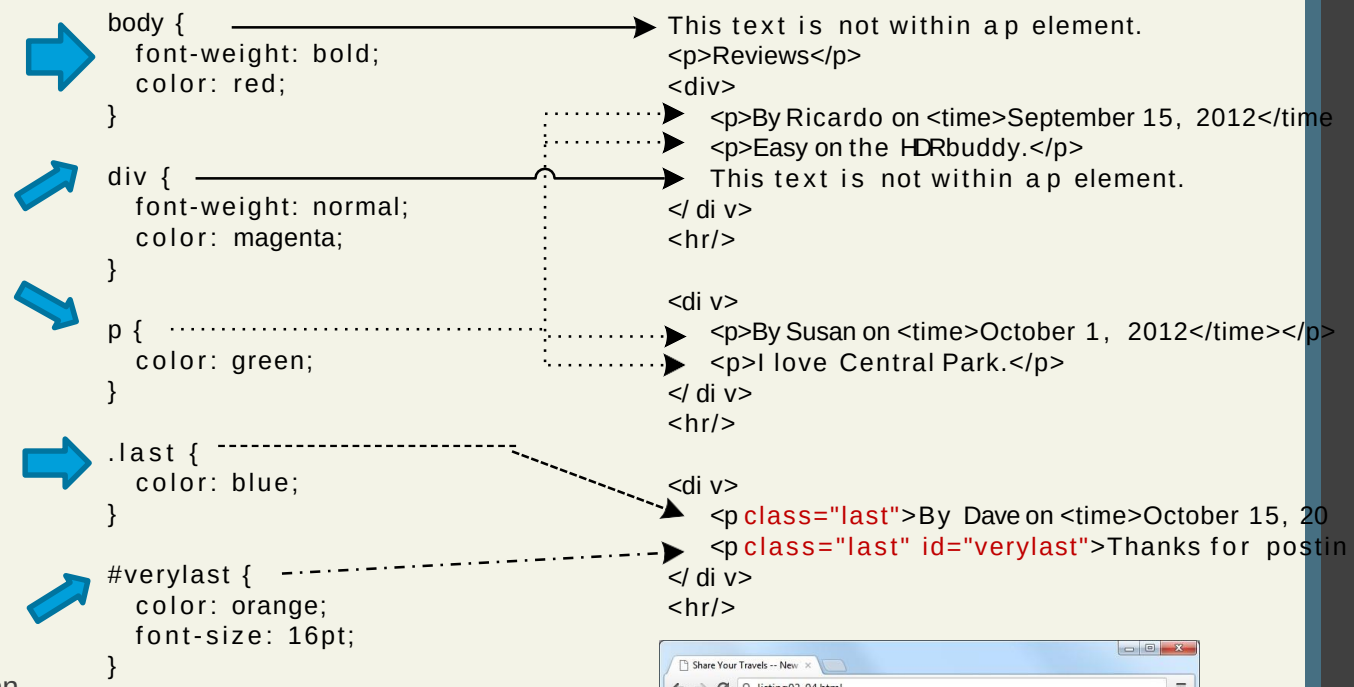
Specificity

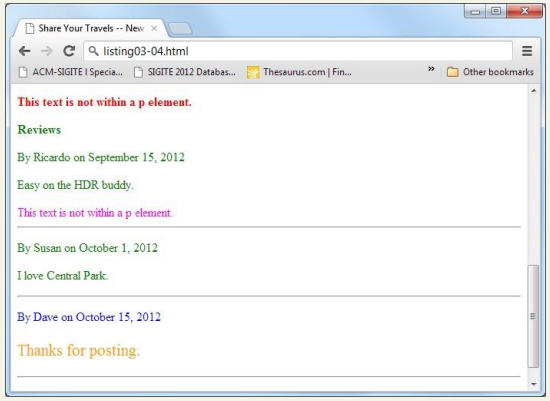
These color and font-weight properties are inheritable and thus potentially applicable to all the child elements contained within the body.

However, because the 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.





Specificity Algorithm

The algorithm that is used to determine specificity is :

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

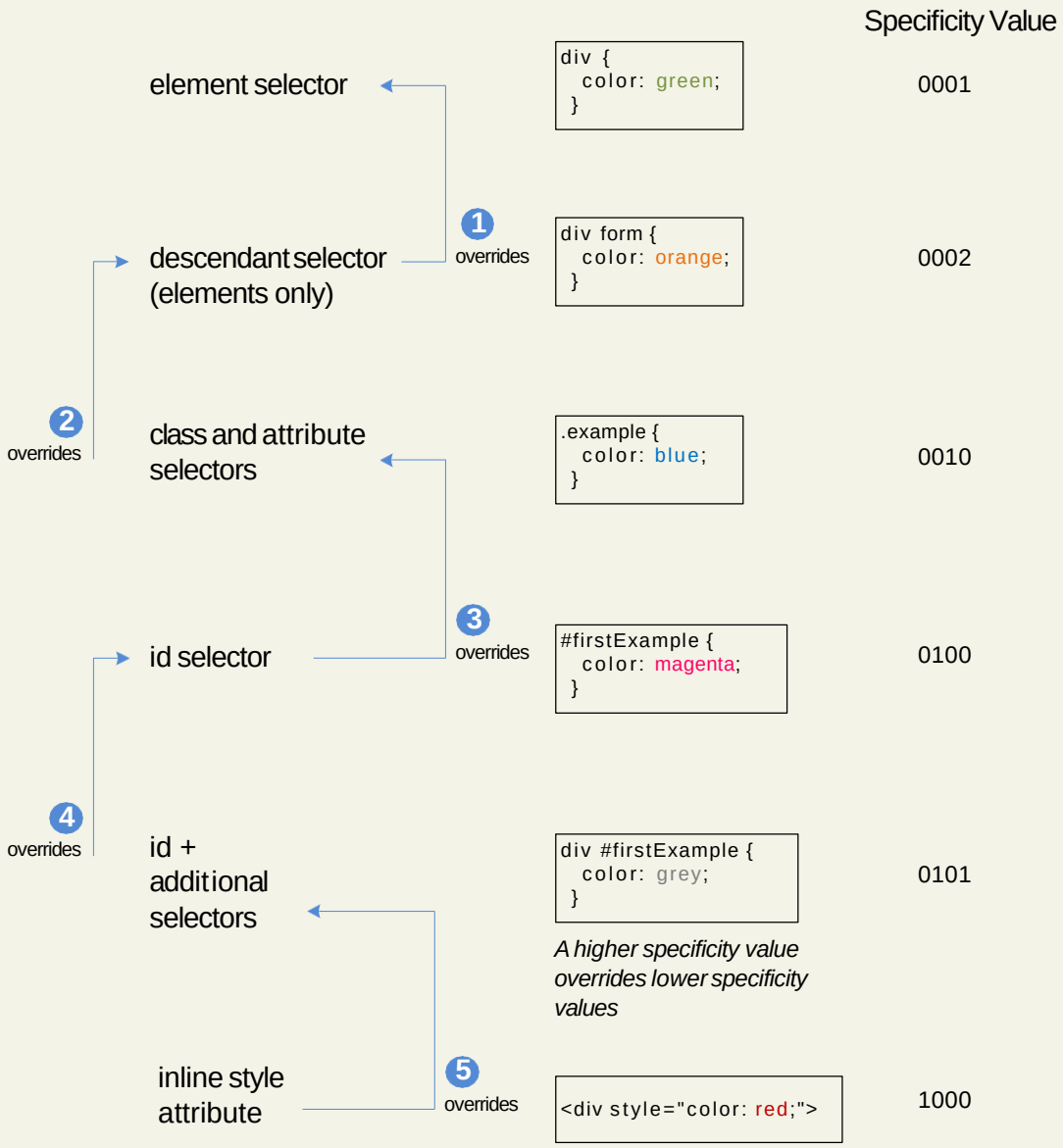
Count the number of ID attributes in the selector (let that value = b).

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

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

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

Specificity Algorithm



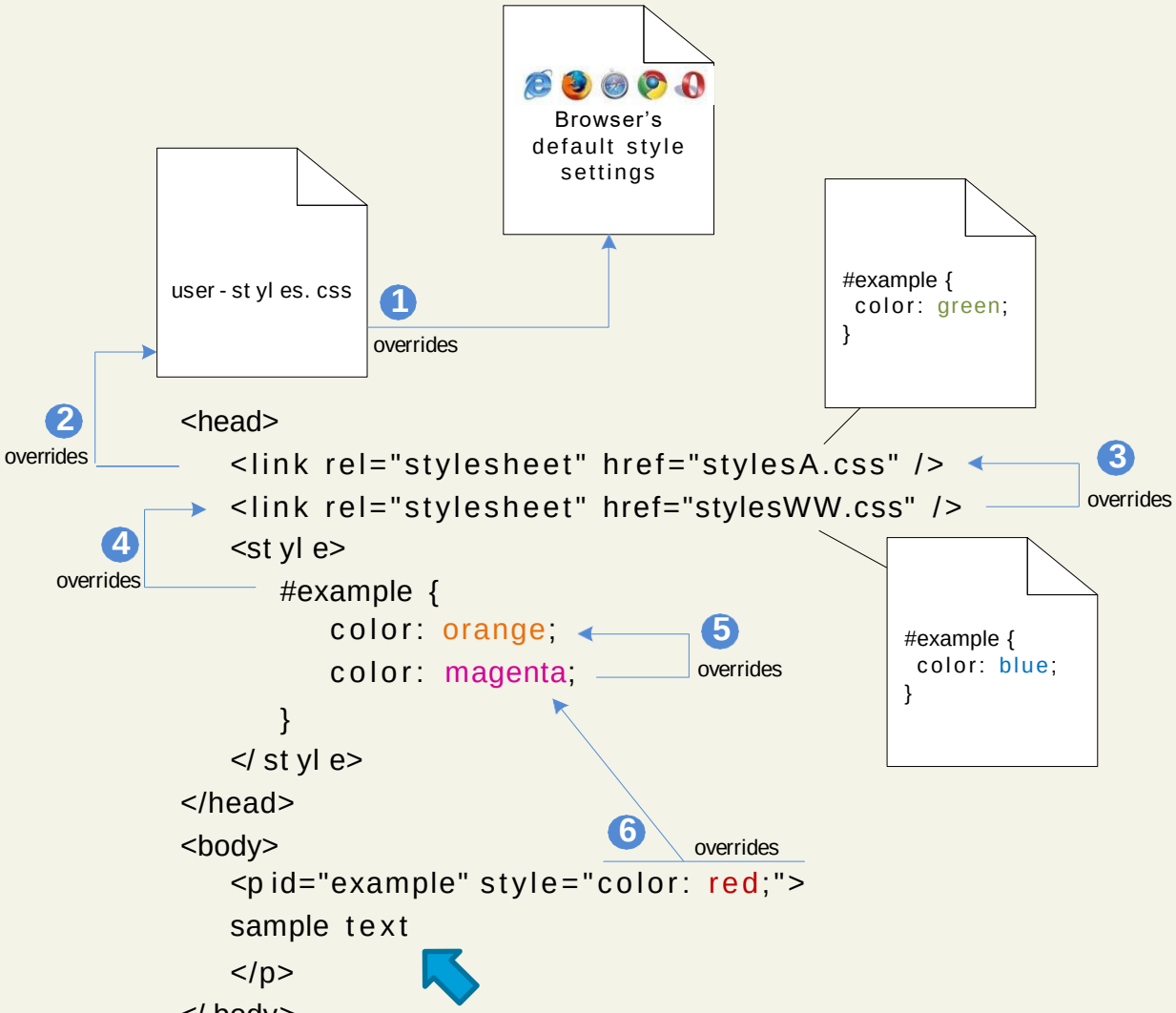
Location

Cascade Principle #3

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

The principle of location is that when rules have the same specificity, then the latest are given more weight.

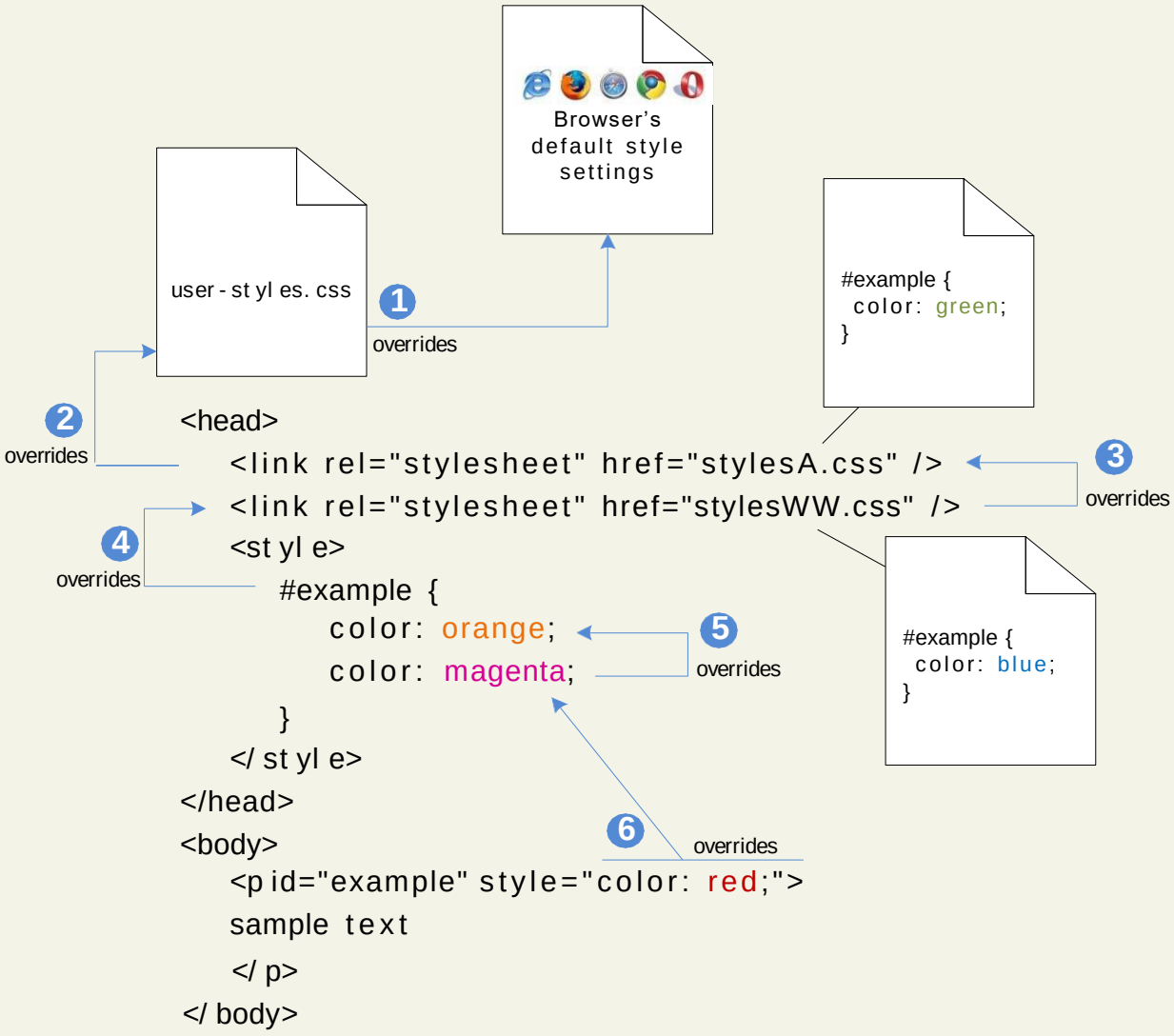
Location



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

Location

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



Location

There's always an exception

There is one exception to the principle of location.

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

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

Section 6 of 7

THE BOX MODEL

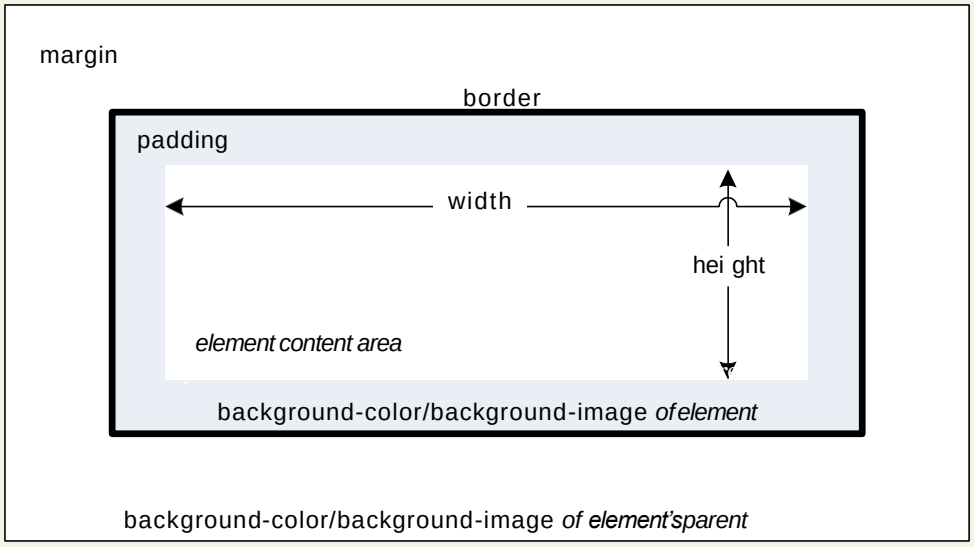
The Box Model

Time to think inside the box

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

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

The Box Model



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

Background

Box Model Property #1

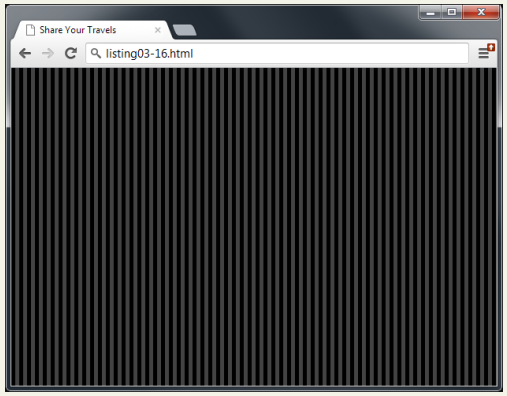
The background color or image of an element fills an element out to its border (if it has one that is).

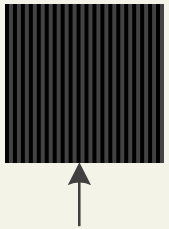
In contemporary web design, it has become extremely common too use CSS to display purely presentational images (such as background gradients and patterns, decorative images, etc) rather than using the <img> element.

Background Properties

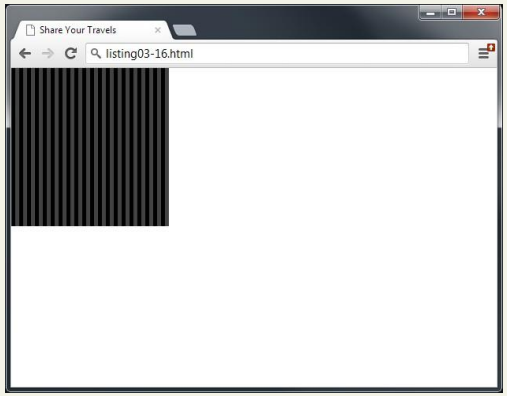
Property	Description
background	A combined short-hand property that allows you to set the background values in one property. While you can omit properties with the short-hand, do remember that any omitted properties will be set to their default value.
background-attachment	Specifies whether the background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

Background Repeat

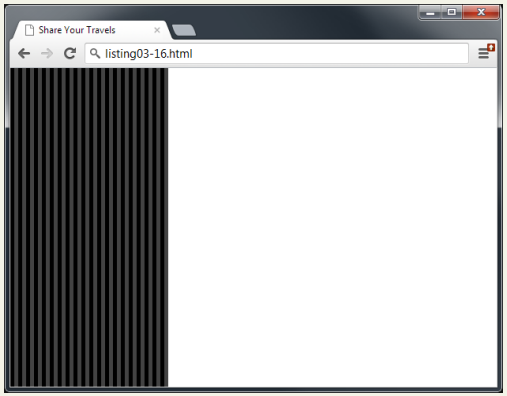




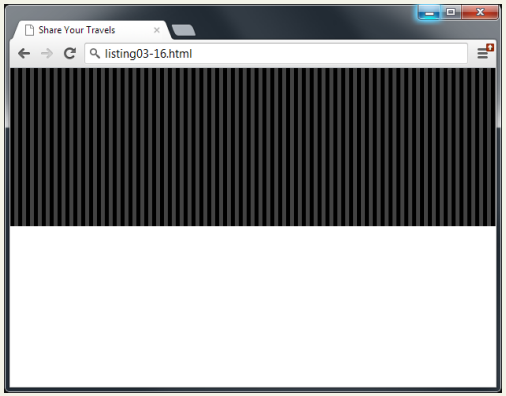
background-image: url(../images/backgrounds/body-background-tile.gif);
background-repeat: repeat;



background-repeat: no-repeat;

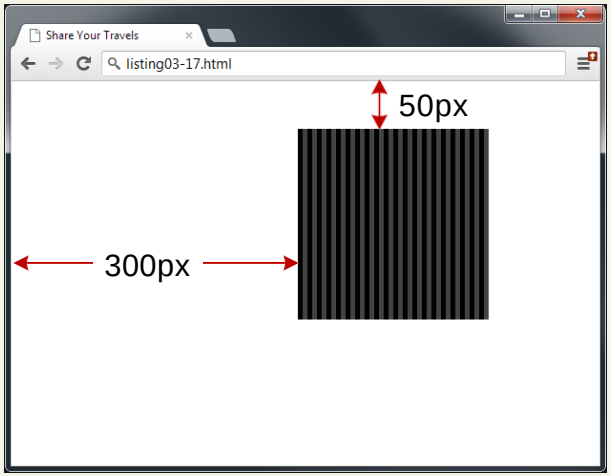


background-repeat: repeat-y;



background-repeat: repeat-x;

Background Position




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

Borders

Box Model Property #2

Borders provide a way to visually separate elements.

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

Borders

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

Shortcut notation

TRBL

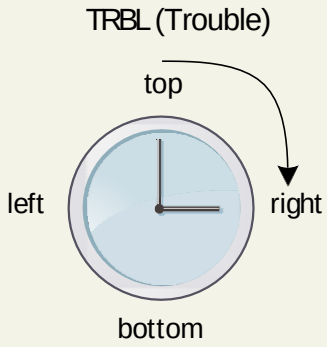
With border, margin, and padding properties, there are long-form and shortcut methods to set the 4 sides

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

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

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

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

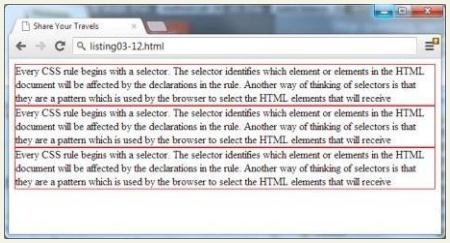



```
border-color: top right bottom left;
```

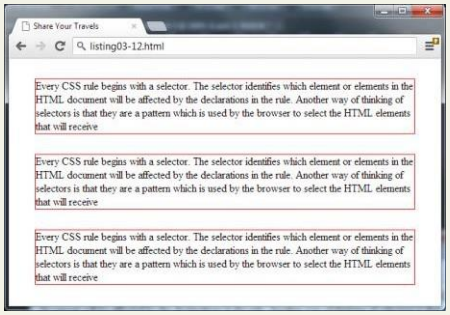
```
border-color: red green orange blue;
```

Margins and Padding

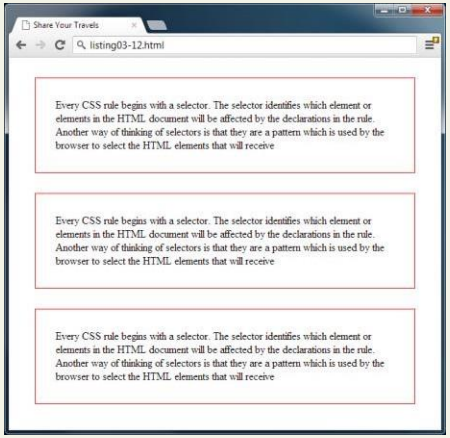
Box Model Properties #3 and #4




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



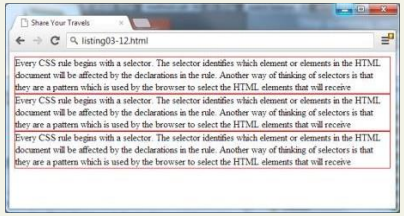

```
p {
    border: solid 1pt red;
    margin: 30px;
    padding: 0;
}
```




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

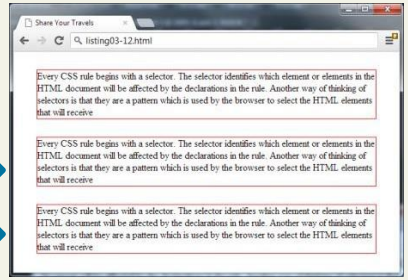
Margins

Why they will cause you trouble.



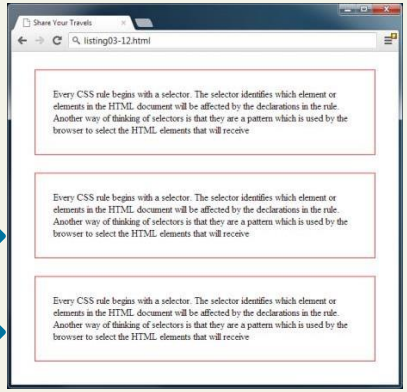

```
p{
    border: solid 1pt red;
    margin: 0;
    padding: 0;
}
```

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?



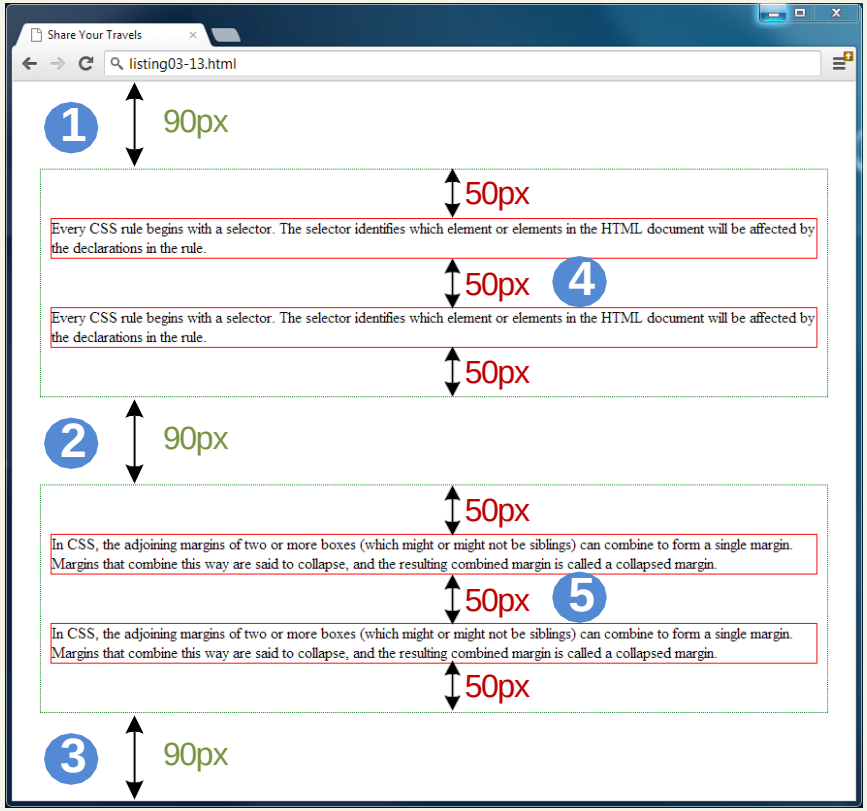

```
p{
    border: solid 1pt red;
    margin: 30px;
    padding: 0;
}
```

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




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

Collapsing Margins




```
<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 + 90 pixels for the top margin of the second), while the margins 4 and 5 for would be 100px.

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

Collapsing Margins

How it works

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

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

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

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

Width and Height

Box Model Properties #5 and #6

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

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

Width and Height

Potential Problem #1

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

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

For text,

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

For images,

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

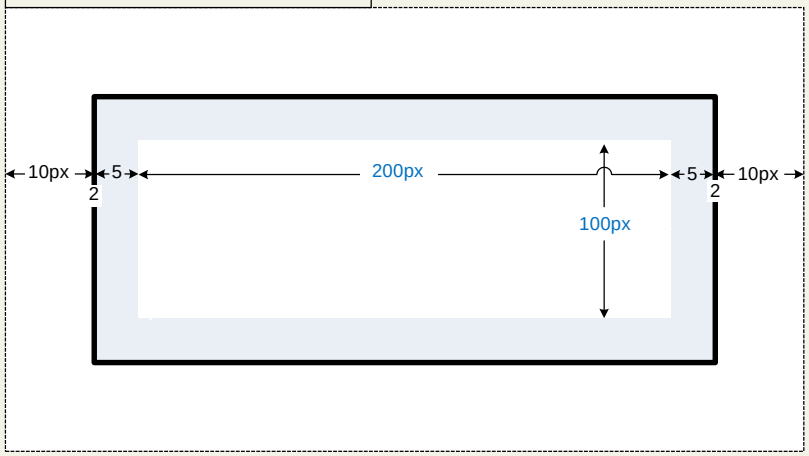
Width and Height

Potential Problem #2

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

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

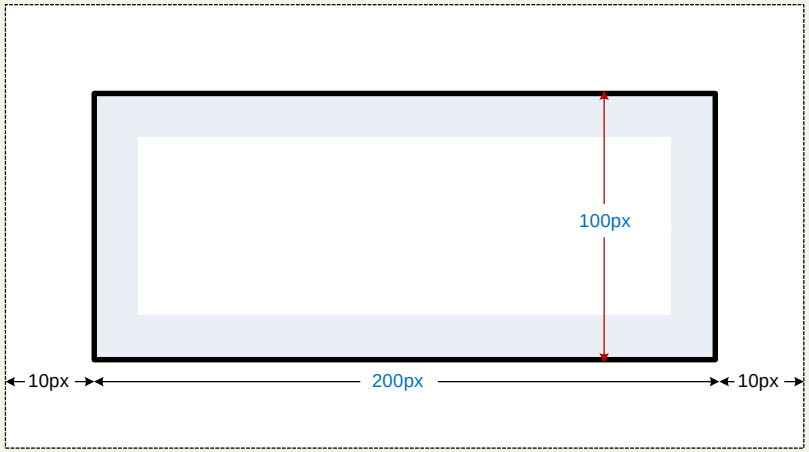
True element width = 10 + 2 + 5 + 200 + 5 + 2 + 10 = 234 px
True element height = 10 + 2 + 5 + 100 + 5 + 2 + 10 = 134 px



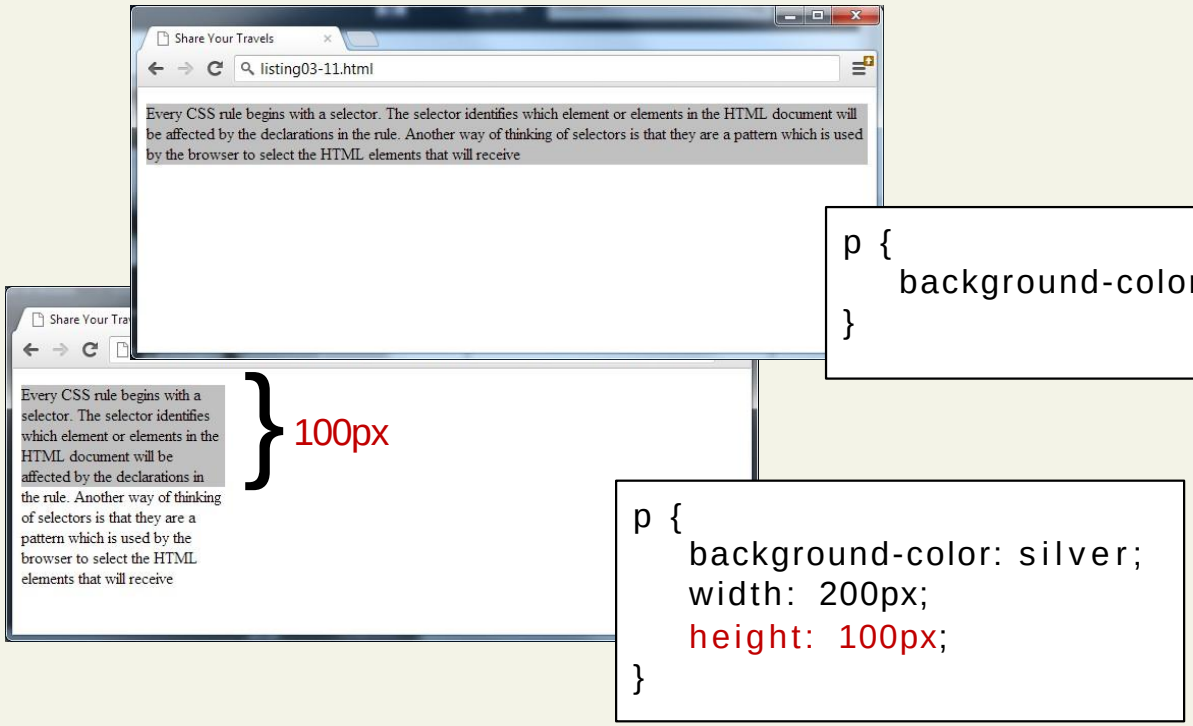
Default

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

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



Width and Height




```
p {
    background-color: silver;
}
```

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

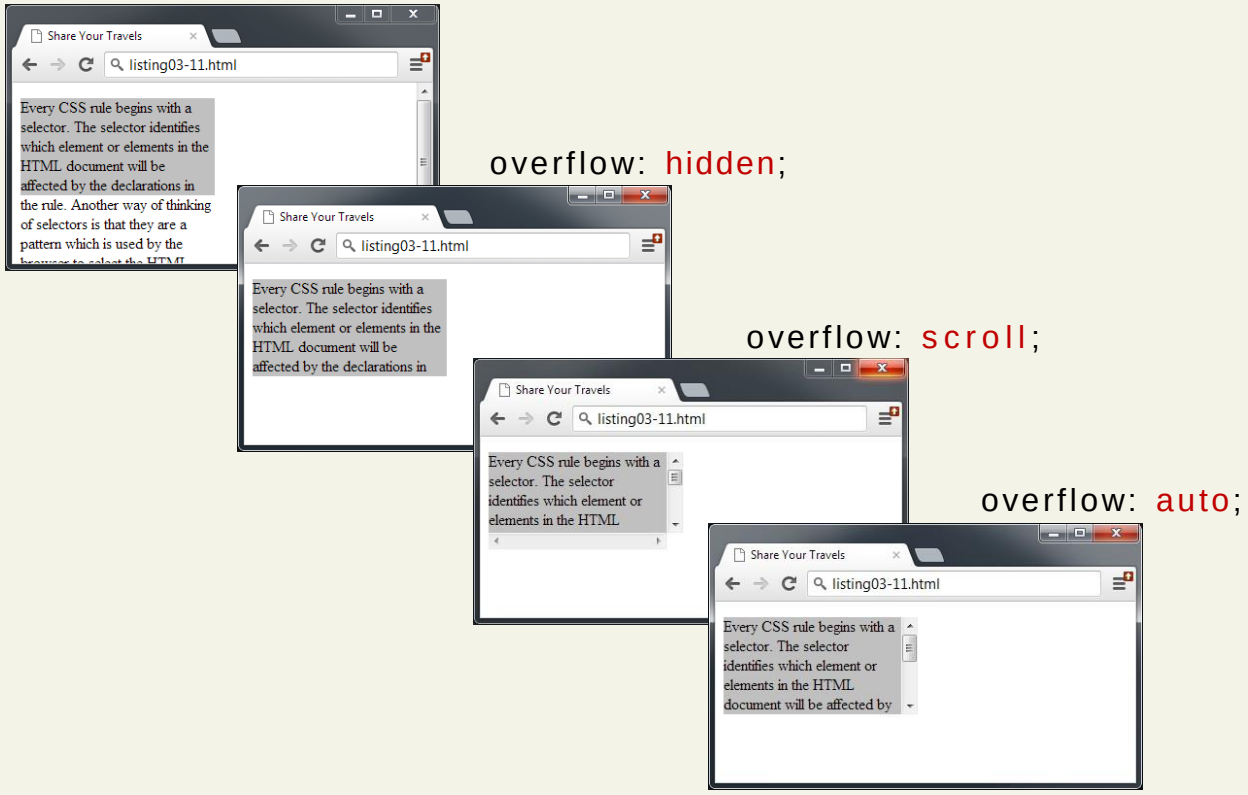
Overflow Property

overflow: visible;

overflow: hidden;

overflow: scroll;

overflow: auto;



Sizing Elements

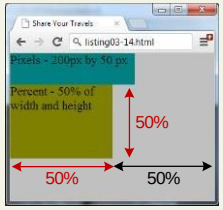
Time to embrace ems and percentages

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

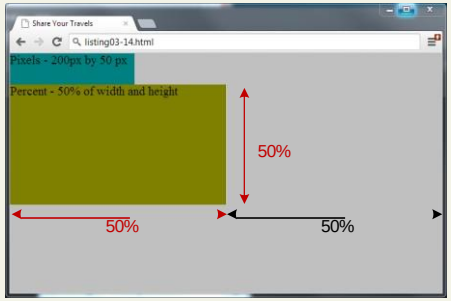
- When you use percentages, the size is relative to the size of the parent element.
- When you use ems, the size of the box is relative to the size of the text within it.

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

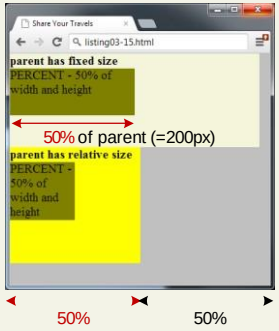
```
<style>
  html,body {
      margin:0;
      width:100%;
      height:100%;
      background: silver;
  }
  .pixels {
      width:200px;
      height:50px;
      background: teal;
  }
  .percent {
      width:50%;
      height:50%;
      background: olive;
  }
```




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




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




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

Developer Tools

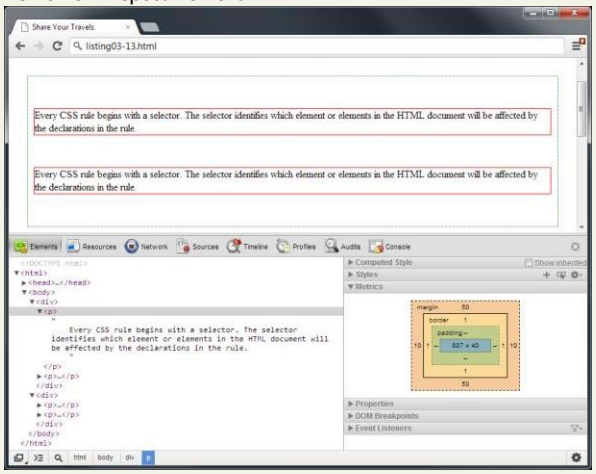
Help is on the way

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

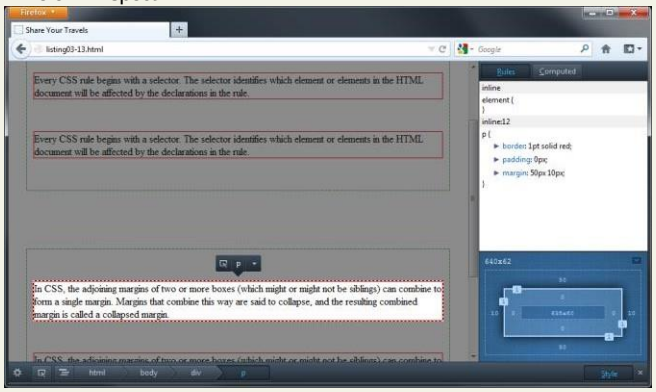
You can use the various browsers' CSS inspection tools to examine, for instance, the box values for a selected element.

Developer Tools

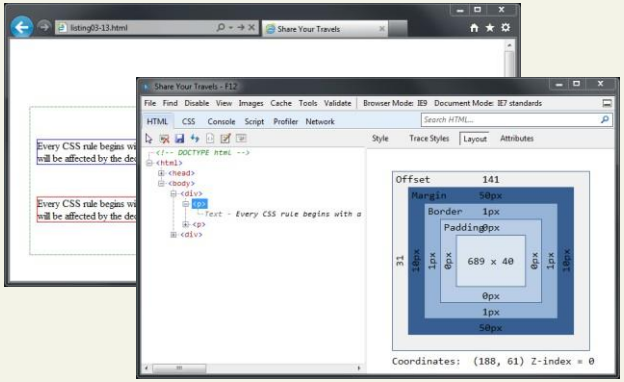
Chrome – Inspect Element



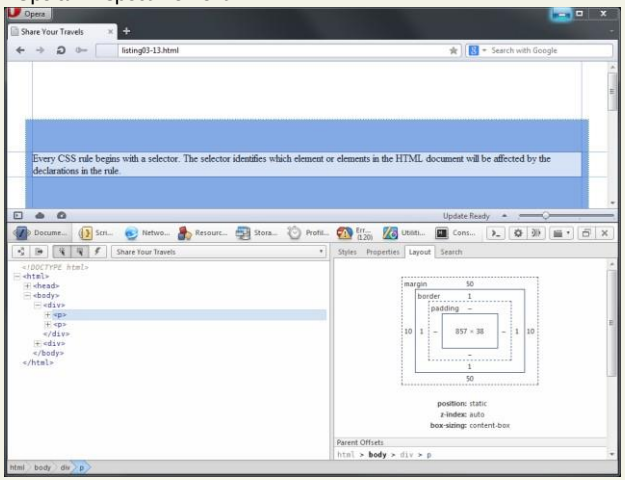
Firefox – Inspect



Internet Explorer – Developer Tools



Opera – Inspect Element



Section 7 of 7

TEXT STYLING

Text Properties

Two basic types

CSS provides two types of properties that affect text.

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

Font-Family

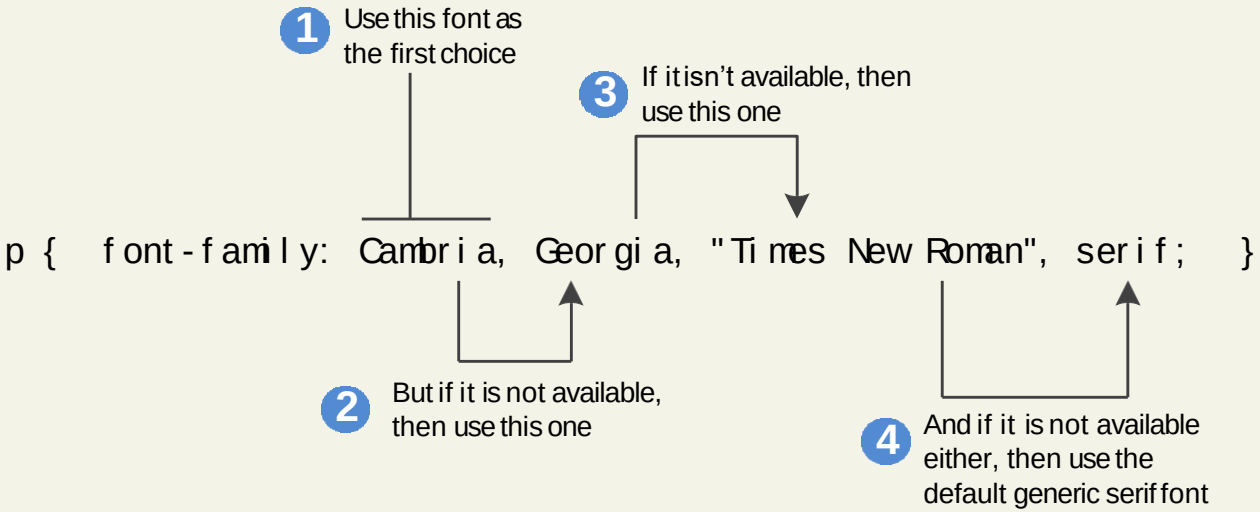
A few issues here

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

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

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

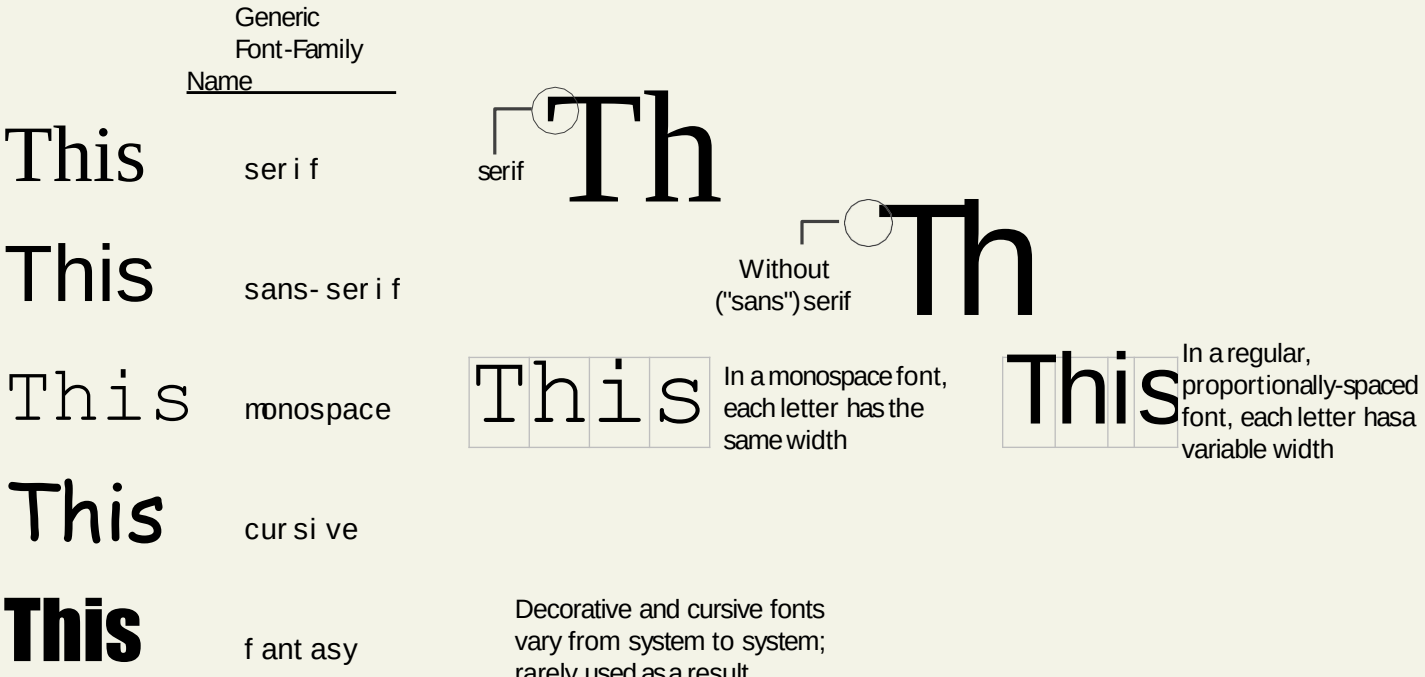
Specifying the Font-Family



Generic Font-Family

The font-family property supports five different generic families.

The browser supports a typeface from each family.



@font-face

The future is now

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

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

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

Font Sizes

Mo control, mo problems

The issue of font sizes is unfortunately somewhat tricky.

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

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

Somewhat surprisingly, pixels are also a problematic unit.

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

Font Sizes

Welcome ems and percents again

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

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

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

How to use ems and percents

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

How to use ems and percents

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

```
/* using 16px scale */

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

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

How to use ems and percents

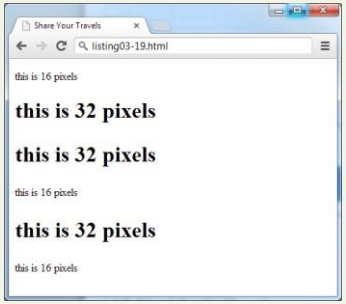
It might seem easy … but it's not …

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

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

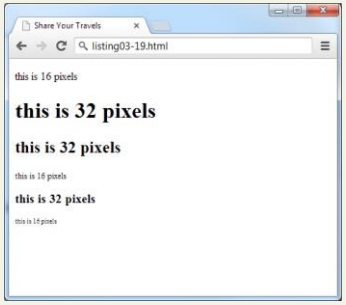
ems and percents

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




```
/* using 16px scale */

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




```
/* using 16px scale */

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

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

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

The rem unit

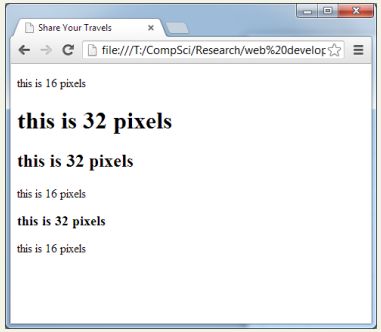
Solution to font sizing hassles?

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

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

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

The rem unit




```
/* using 16px scale */

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

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

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

What you've learned

1. What is CSS?
2. CSS Syntax
3. Location of Styles
4. Selectors
5. The Cascade: How Styles Interact
6. The Box Model
7. CSS Text Styling