# MODULE1

# Chapter1: Introduction to HTML

1 What is HTML and Where Did It Come From?
2 HTML Syntax
3 Semantic Markup
4 Structure of HTML Documents
5 Quick Tour of HTML Elements
6 HTML5 Semantic Structure Elements.

## 1.1 What Is HTML and Where Did It Come from?

- HTML is defined as a **markup language**. A markup language is simply a way of annotating a document in such a way as to make the annotations distinct from the text being annotated.
- Markup languages such as HTML, Tex, XML, and XHTML allow users to control how text and visual elements will be laid out and displayed. 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.
- At its simplest, **markup** is a way to indicate *information about the content* that is distinct from the content. This "information about content" in HTML is implemented via **tags.**
- Markup languages are able to encode information how to display the content for the end user.
- The combining semantic markup with presentation markup is no longer permitted in HTML5, "formatting the content" for display remains a key reason why HTML was widely adopted.

**XHTML**

- Instead of growing HTML, the W3C turned its attention in the late 1990s to a new specification called XHTML 1.0, which was a version of HTML that used stricter **XML** (extensible markup language) syntax rules.
- 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**.

- To help web authors, two versions of XHTML were created: **XHTML 1.0Strict** 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 flavour of XHTML, and was meant to act as a temporary transition to the eventual global adoption of XHTML Strict.

## XML

- XML is a more general markup language than HTML. It is used to mark up any type of data. XML-based data formats (called **schemas** in XML) are almost everywhere.
- The following is an example of a simple XML document:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<art>
  <painting id="290">
    <title>Balcony</title>
    <artist>
      <name>Manet</name>
      <nationality>France</nationality>
    </artist>
    <year>1868</year>
    <medium>Oil on canvas</medium>
  </painting>
</art>
```

- The XML-based syntax rules for XHTML are pretty easy to follow. The main rules are:
    - There must be a single root element.
    - Element names are composed of any of the valid characters (most punctuation symbols and spaces are not allowed) in XML.
    - Element names can't start with a number.
    - Element and attribute names are case sensitive.
    - Attributes must always be within quotes.
    - All elements must have a closing element (or be self-closing).

- XML also provides a mechanism for validating its content. It can check, for instance, whether an element name is valid, or elements are in the correct order, or that the elements follow a proper nesting hierarchy.
- **HTML validators** means verifying that a web page's markup followed the rules for XHTML Transitional or Strict. Web developers often placed proud images on their sites

to tell the world at large that their site followed XHTML rules (and also to communicate their support for web standards).

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

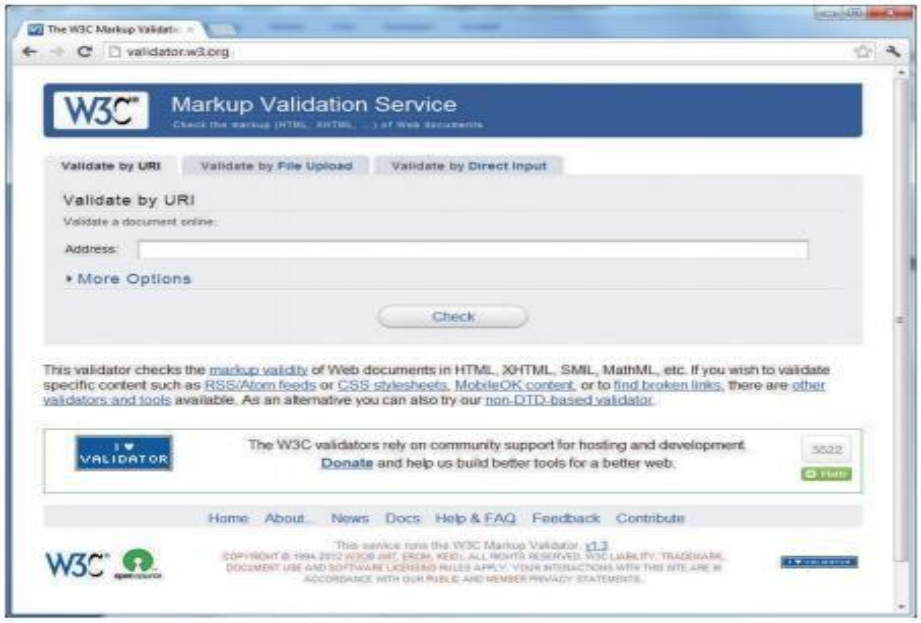



**Figure 1.1** W3C XHTML validation service

## HTML5

- A group of developers at Opera and Mozilla formed the **WHATWG** (Web Hypertext Application Technology Working Group) group within the W3C.
- There are three main aims to HTML5:
    1. Specify unambiguously how browsers should deal with invalid markup.
    2. Provide an open, nonproprietary programming framework (via JavaScript) for creating rich web applications.
    3. Be backwards compatible with the existing web.
- 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.

## 1.2 HTML Syntax

### Elements and Attributes

- HTML documents are composed of textual content and HTML elements.
- The term **HTML element** is often used interchangeably with the term **tag**.
- An HTML element is identified in the HTML document by tags.
- A tag consists of the element name within angle brackets.
- The element name appears in both the beginning tag and the closing tag, which contains a forward slash followed by the element's name, again all enclosed within angle brackets.
- The closing tag acts like an off-switch for the on-switch that is the start tag.
- 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.
- Some HTML attributes expect a number for the value. These will just be the numeric value; they will never include the unit.


**Figure 1.2** The parts of an HTML element

- Figure 1.2 illustrates the different parts of an HTML element, including an example of an empty HTML element.
- An **empty element** does not contain any text content, instead, it is an instruction to the browser to do something. Perhaps the most common empty element is <img>, the image element.
- In XHTML, empty elements had to be terminated by a trailing slash, the trailing slash in empty elements is optional.

**Nesting HTML Elements**

- HTML element will contain other HTML elements. 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 **descendants** of the parent element; likewise, any given child element, may have a variety of **ancestors**.
- In XHTML, all HTML element names and attribute names had to be lowercase.
- HTML5 and HTML 4.01 does not care whether you use upper or lowercase for element or attribute name.

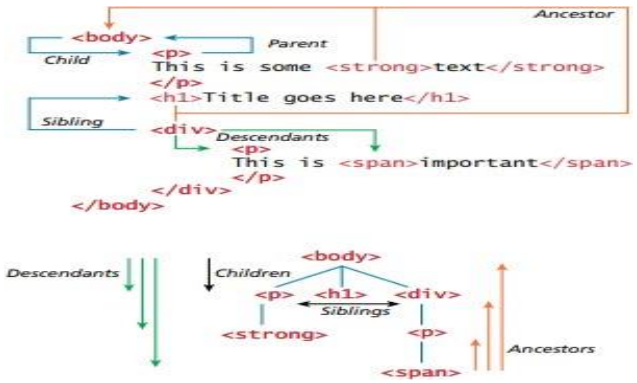



**Figure 1.3:** HTML document outline

- This underlying family tree or hierarchy of elements **Cascading Style Sheets** (CSS) and JavaScript programming and parsing.
- In order to properly construct this hierarchy of elements, our 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, as shown in Figure 1.4

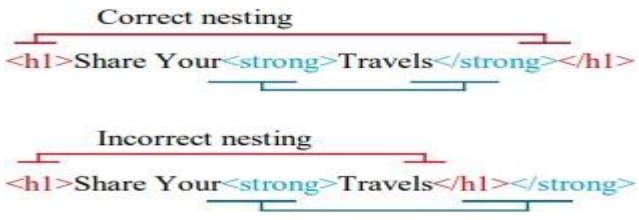



**Figure 1.4:** The proper nesting of HTML elements

## 1.3 Semantic Markup

- 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).
- 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.
- That structure (and to be honest the presentation as well) makes it easier for the reader to quickly grasp the hierarchy of importance as well as the broad meaning of the information in the document.
- Structure is a vital way of communicating information in paper and electronic documents that are used to describe the basic structural information in a document, such as headings, lists, paragraphs, links, images, navigation, footers, and so on.

**Advantages**

1. **Maintainability**. Semantic markup is easier to update and change than webpages that contain a great deal of presentation markup.
2. **Faster**. Semantic web pages are typically quicker to author and faster to download.
3. **Accessibility**. Not all web users are able to view the content on web pages. Users with sight disabilities experience the web using voice reading software.
    - Visiting a web page using voice reading software can be a very frustrating experience if the site does not use semantic markup.
    - For instance, the United States government has its own Section 508 Accessibility Guidelines (**http://www.section508.gov**).
4. **Search engine optimization**. For many site owners, the most important users of a website are the various search engine crawlers. Semantic markup provides better instructions for these crawlers: it tells them what things are important content on the site.

## 1.4 Structure of HTML Documents

- The <title> element (Item **1** in Figure 1.5) 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, as shown in the example in Figure 1.5.

- The title is used by the browser for its bookmarks and its browser history list. The operating system might also use the page's title, for instance, in the Windows taskbar or in the Mac dock. And even search engines will typically use the page's title as the linked text in their search engine result pages.
- Figure 1.6 illustrates a more complete HTML5 document that includes these other structural elements as well as some other common HTML elements.

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


**Figure 1.5:** One of the simplest possible HTML5 documents

## DOCTYPE

- Item **1** in Figure 1.6 points to the DOCTYPE (short for **Document Type Definition**) element, which tells the browser what type of document it is about to process.
- In HTML5 doctype is quite short in comparison to one of the standard doctype specifications for XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

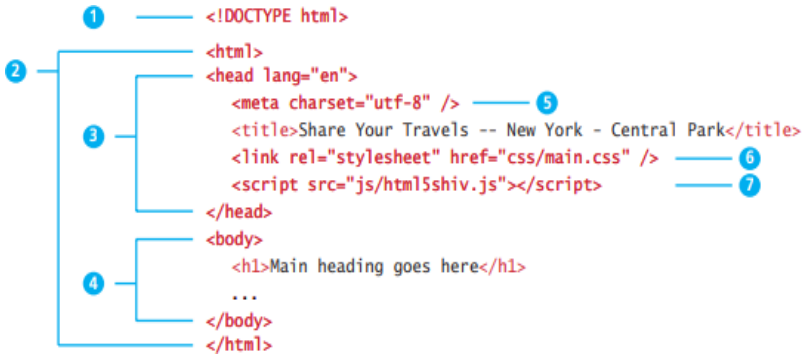



**Figure 1.6:** Structure elements of an HTML5 document

- The doctype
- was used to tell the browser to render an HTML document.
- Document Type Definitions (DTD) define a document's type for markup languages such as HTML and XML.
- In both these markup languages, the DTD must appear near the beginning of the document. DTDs have their own syntax that defines allowable element names and their order.

**Head and Body**

- HTML5 does not require the use of the <html>, <head>, and <body> elements. However, in XHTML they were required, and most web authors continue to use them.
- The <html> element is sometimes called the **root element** as it contains all the other HTML elements in the document.
- It also has a **lang** attribute. This optional attribute tells the browser the natural language that is being used for textual content in the HTML document, which is English in this example.
- This doesn't change how the document is rendered in the browser; rather, search engines and screen reader software can use this information. HTML pages are divided into two sections: the **head** and the **body**, which correspond to the <head> and <body> elements.
- The head contains descriptive elements *about* the document, such as its title, any style sheets or JavaScript files it uses, and other types of meta information used by search engines and other programs.
- The body contains content (both HTML elements and regular text) that will be displayed by the browser.
- Character encoding refers to which character set standard is being used to encode the characters in the document.
- **UTF-8** is a more complete variable-width encoding system that can encode all 110,000 characters in the Unicode character set (which in itself supports over 100 different language scripts).
- Item 6 in Figure 1.6 specifies an external CSS style sheet file that is used with this document. Virtually all commercial web pages make use of style sheets to define the visual look of the HTML elements in the document.

- Styles can also be defined within an HTML document for consistency's sake, most sites place most or all of their style definitions within one or more external style sheet files.
- Finally, Item 7 in Figure 1.6 references an external JavaScript file. Most modern commercial sites use at least some JavaScript. Like with style definitions, JavaScript code can be written directly within the HTML or contained within an external file.

## 1.5 Quick Tour of HTML Elements

### Headings

- Item 1 in Figure 1.7 defines two different headings. HTML provides six levels of heading (h1 through h6), with the higher heading number indicating a heading of less importance.
- In the real-world documents we know that headings are an essential way for document authors to show their readers the structure of the document.
- Headings are also used by the browser to create a **document outline** for the page. Every web page has a document outline. This outline is not something that we see.

```
<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, 2015 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, 2015</time></p>
     <p>Easy on the HDR buddy.</p>
  </div>

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

  <p><small>Copyright &copy; 2015 Share Your Travels</small></p>
</body>
```

**Figure 1.7:** Sample HTML5 document

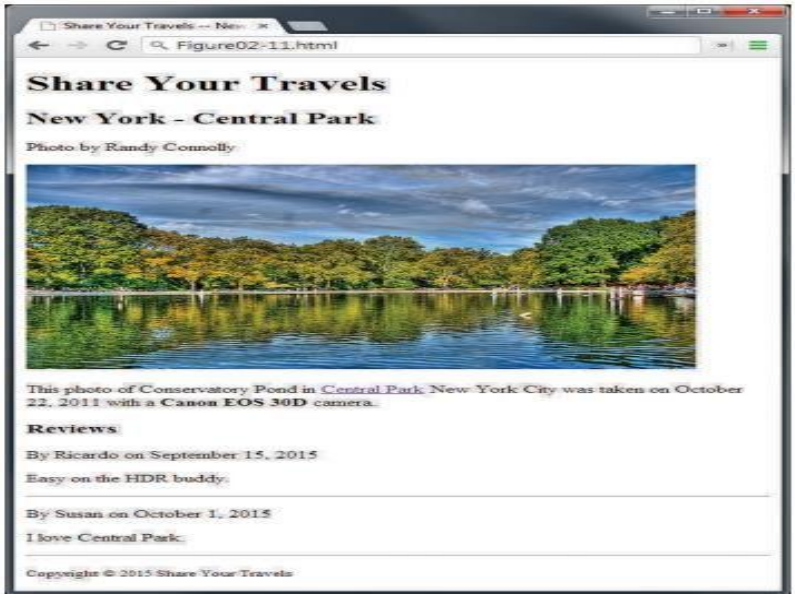



**Figure 1.8:** figure 1.7 in the browser

- This document outline is constructed from headings and other structural tags in our content. There is a variety of web-based tools that can be used to see the document outline.
- Specify a heading level that is semantically accurate; do not choose a heading level because of its default presentation. 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).

**Paragraphs and Divisions**

- Item 2 in Figure 1.7 defines two paragraphs, the most basic unit of text in an HTML document.
- <p> tag is a container and can contain HTML and other **inline HTML elements** (the  and <a> elements in Figure 1.7).
- The line break element (br /) forces a line break. It is suitable for text whose content belongs in a single paragraph but which must have specific line breaks: for example, addresses and poems.
- Item 6 in Figure 1.7 illustrates the definition of a  element. This element is also a container element and is used to create a logical grouping of content (text and other HTML elements, including containers such as <p> and other elements). The 

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

**Links**

- Item 3 in Figure 1.7 defines a hyperlink. Links are an essential feature of all web pages. Links are created using the <a> element (the "a" stands for anchor).
- A link has two main parts: the destination and the label. As can be seen in Figure 1.9, the label of a link can be text or another HTML element such as an image.

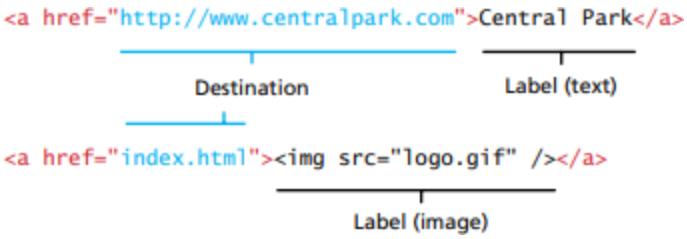



**Figure 1.9:** Two parts of a link

- We can use the anchor element to create a wide range of links. These include:
  - 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 (whether on the same site or on an external site).
  - 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.
  - Links that are instructions to the mobile browser to make a phone call.
  - Figure 1.10 illustrates the different ways to construct link destinations


**Figure 1.10:** Different link destinations

## URL Relative Referencing

- When referencing a page or a resource is on different site or different server then, it is referred as **absolute** referencing.
  - the protocol (typically, http://),
  - the domain name,
  - any paths, and then finally
  - the file name of the desired resource.
- When referencing a page or a resource is on the same site or same server as our HTML document, it is referred as **relative** referencing.
  - If the URL does not include the "**http://**" then the browser will request the current server for the file.
  - If all the resources for the site reside within the same **directory** (folder), then we can reference those other resources simply via their file name.

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

➢ Figure 1.11 illustrates the file structure of an example site. Table 1.1 provides additional explanations and examples of the different types of URL referencing.

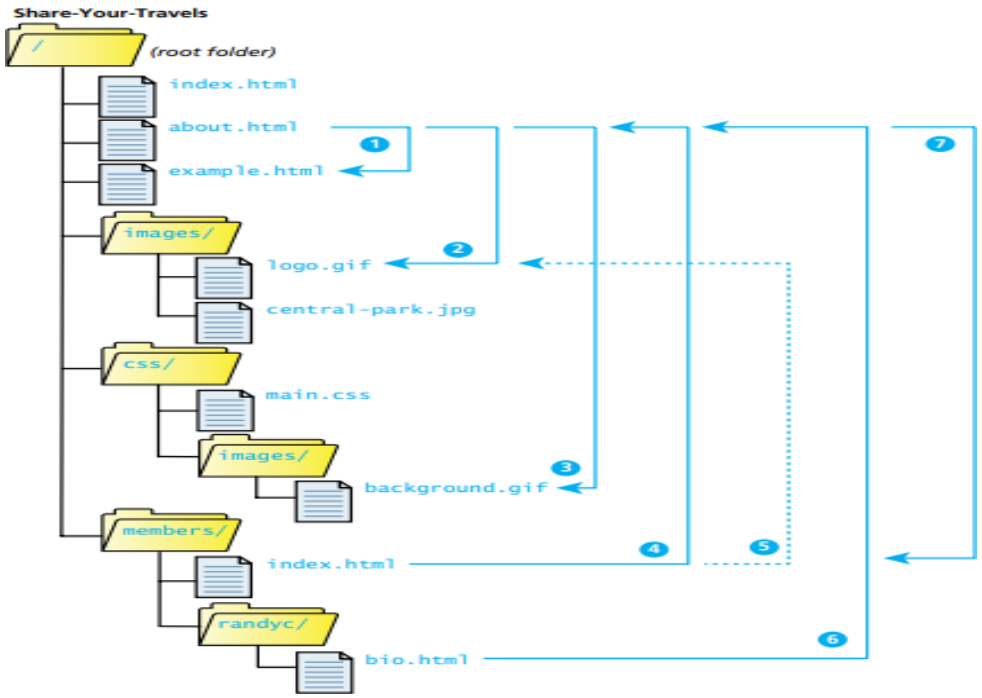



**Figure 1.11:** Example site directory tree

## Inline Text Elements

➢ The html elements which do not disrupt the flow of text (i.e., cause a line break) are referred as inline elements. HTML defines over 30 of these elements. Table 1.2 lists some of the most commonly used of these elements.

| Relative Link Type | Example |
|---|---|
| **1 Same Directory**<br>To link to a file within the same folder, simply use the file name. | To link to example.html from about.html (in Figure 2.17), use:<br><a href="example.html"> |
| **2 Child Directory**<br>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:<br><a href="images/logo.gif"> |
| **3 Grandchild/Descendant Directory**<br>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:<br><a href="css/images/background.gif"> |
| **4 Parent/Ancestor Directory**<br>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:<br><a href="../about.html"><br>To link to about.html from bio.html, use:<br><a href="../../about.html"> |
| **5 Sibling Directory**<br>Use "../" to move up to the appropriate level, and then use the same technique as for child or grandchild directories. | To link to about.html from index.html in members, use:<br><a href="../images/about.html"><br>To link to background.gif from bio.html, use:<br><a href="../../css/images/background.gif"> |
| **6 Root Reference**<br>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:<br><a href="/about.html"><br>To link to background.gif from bio.html, use:<br><a href="/images/background.gif"> |
| **7 Default Document**<br>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:<br><a href="members"><br>Or<br><a href="/members"> |

**Table 1.1:** Sample Relative Referencing

| Element | Description |
|---|---|
| <a> | Anchor used for hyperlinks. |
| <abbr> | An abbreviation |
| <br> | Line break |
| <cite> | Citation (i.e., a reference to another work). |
| <code> | Used for displaying code, such as markup or programming code. |
|  | Emphasis |
| <mark> | For displaying highlighted text |
| <small> | For displaying the fine-print, i.e., "non-vital" text, such as copyright or legal notices. |
|  | The inline equivalent of the  element. It is generally used to mark text that will receive special formatting using CSS. |
|  | For content that is strongly important. |
| <time> | For displaying time and date data |

**Table 1.2:** Common Text-Level Semantic Elements

## Images

- Item 5 in Figure 1.7 defines an image. While the <img> tag is the oldest method for displaying an image, it is not the only way. In fact, it is very common for images to be added to HTML elements via the background-image property in CSS, a technique.

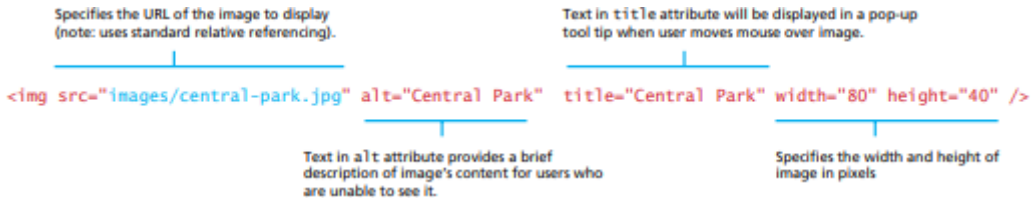



**Figure 1.12:** The <img> element

## Character Entities

- Item 9 in Figure 1.7 illustrates the use of a **character entity**.
- These are special characters for symbols for which there is either no easy way to type them via a key board (such as the copyright symbol or accented characters) or which have are reserved meaning in HTML (for instance the "<" or ">" symbols).
- There are many HTML character entities which can be used in an HTML document by using the entity name or the entity number. Some of the most common are listed in Table 1.3.

| Entity Name | Entity Number | Description |
|---|---|---|
|   |   | Nonbreakable space. **The browser ignores multiple spaces in the source HTML file. If you need to display multiple spaces, you can do so using the nonbreakable space entity.** |
| &lt; | &#60; | Less than symbol ("<"). |
| &gt; | &#62; | Greater than symbol (">"). |
| &copy; | &#169; | The © copyright symbol. |
| &euro; | &#8364; | The € euro symbol. |
| &trade; | &#8482; | The ™ trademark symbol. |
| &uuml; | &#252; | The ü—i.e., small u with umlaut mark. |

**Table 1.3:** Common Character Entities

## Lists

A Collection of items forms a lists. HTML provides three types of lists:

1. **Unordered lists(ul)**. Collections of items in no particular order; these are by default rendered by the browser as a bulleted list.
2. **Ordered lists(ol)**. Collections of items that have a set order; these are by default rendered by the browser as a numbered list.
3. **Definition lists(dl)**. Collection of name and definition pairs. These tend to be used infrequently. The various list elements are container element containing list item elements (<li>). Other HTML elements can be included with in the <li> container, as shown in the first list item of the unordered list in Figure 1.13.

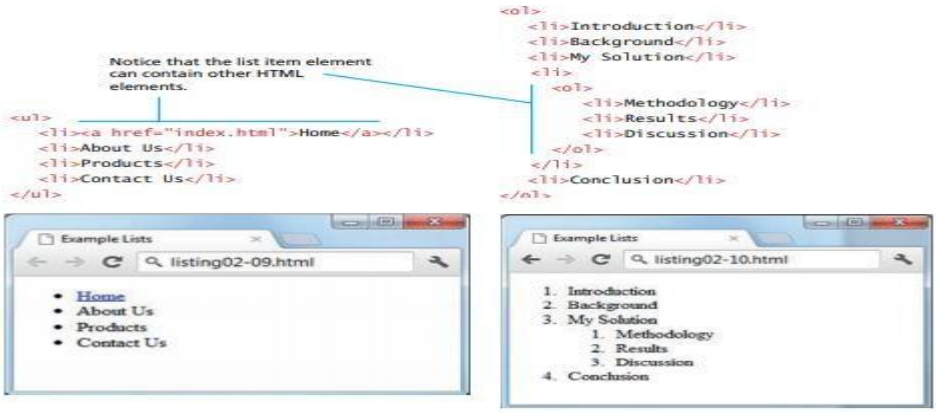



**Figure 1.13:** List elements and their default rendering

## 1.6 HTML5 Semantic Structure Elements

- Semantic structural elements describes its meaning to both the browser and the developer.
- Most complex websites are absolutely packed solid with elements. Most of these are marked with different id or class attributes.
- HTML5 proposes new semantic block structuring elements. The idea behind using these elements is that our markup will be easier to understand because we will be able to replace some of your  sprawl with cleaner and more self-explanatory HTML5 elements.

### Header and Footer

- Most website pages have a recognizable header and footer section. Typically the **header** contains the site logo and title, horizontal navigation links, and perhaps one or two horizontal banners.
- 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.

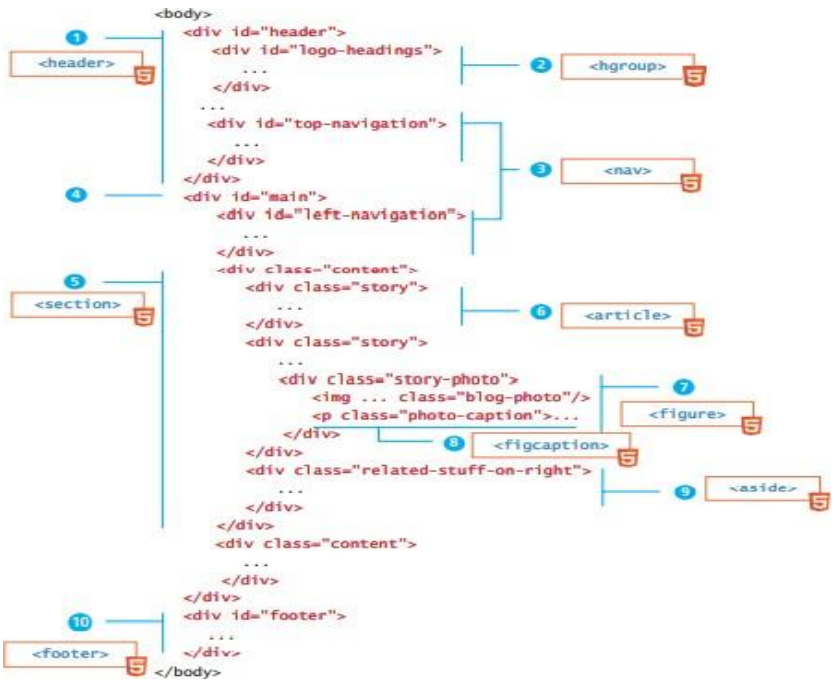



**Figure 1.14:** Sample  based XHTML layout (with HTML5 equivalents)

- Both the HTML5 <header> and <footer> element can be used not only for *page* headers and footers, but also for header and footer elements within other HTML5 containers, such as <article> or <section>.
- The header element typically contains the headings for a section (an h1–h6 element or hgroup element), along with content such as introductory material or navigational aids for the section.
- The browser really doesn't care how one uses these HTML5 semantic structure elements. Just like with the  element, there is no predefined presentation for these tags.

```
<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, 2015</time></p>
    </header>
    ...
</article>
```

**Listing 1.1:** Heading example

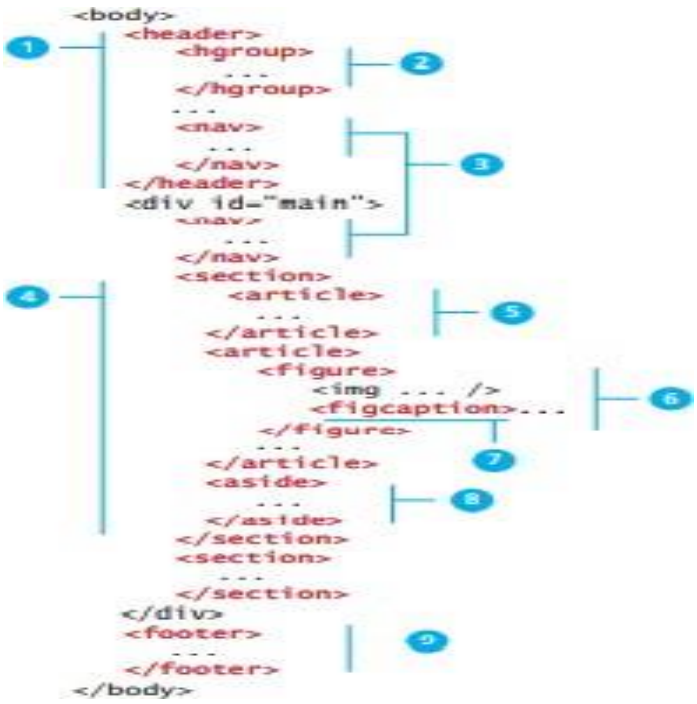



**Figure 1.15:** Sample layout using new semantic structure elements

## Heading Groups

- The <hgroup> element (item 2 in Figure 1.15) can be used to group heading tags together within one container.
- The <hgroup> element can be used in contexts other than a header. For instance, one could also use an <hgroup> within an <article> or a <section> element as well.
- The <hgroup> element can *only* contain <h1>, <h2>, etc., elements. Listing 2.2illustrates two example usages of the <hgroup> element.

```
<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>
```

**Listing1.2:** hgroup example

## Navigation

- The <nav> element (item 3 in Figure 1.15) represents a section of a page that contains links to other pages or to other parts within the same page. The <nav> element was intended to be used for major navigation blocks.
- Not all groups of links on a page need to be in a nav element—the element is primarily intended for sections that consist of major navigation blocks.
- In particular, it is common for footers to have a short list of links to various pages of a site, such as the terms of service, home page, and a copyright page. The <footer> alone is sufficient for such cases; while a nav element such cases, it is usually unnecessary

```
<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>
```

**Listing 1.3:** nav example

## Articles and Sections

### Articles

- The item 5 in figure 1.15 indicates the article element. 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 article element represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content

### Sections

- The item 4 in figure 1.15 indicates the section element. The section element represents a section of a document, typically with a title or heading.
- The section element represents a generic section of a document or application. A section, is a thematic grouping of content, typically with a heading. Examples of sections would

be chapters, the various tabbed pages in a tabbed dialog box, or the numbered sections of a thesis. A Website's home page could be split into sections for an introduction, news items, and contact information.

**Figure and Figure Captions**

- In HTML5 we can instead use the more obvious <figure> and <figcaption>elements (items 6 and 7 in Figure1.15).
- The 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.
- The figure element represents some flow content, optionally with a caption, that is self-contained and is typically referenced as a single unit from the main flow of the document.
- The element can thus be used to annotate illustrations, diagrams, photos, code listings, etc, that are referred to from the main content of the document, but that could, without affecting the flow of the document, be moved away from that primary content, e.g. to the side of the page, to dedicated pages, or to an appendix.


**Figure 1.16:** The figure and fig caption elements in the browser

**Aside**

- The <aside> element (item 8 in Figure 1.15) 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.
- 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.

# Chapter 2: Introduction to CSS

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

## 2.1 What Is CSS?

- HTML should not describe the formatting or presentation of documents. Instead that presentation task is best performed using **Cascading Style Sheets (CSS)**.
- CSS is a W3C standard
  - For describing the appearance of HTML elements.
  - To describe CSS's function is to say that CSS is used to define the **presentation** of HTML documents.
- With CSS, we can assign font properties, colors, sizes, borders, background images, and even position elements on the page.
- CSS can be added directly to any HTML element (via the style attribute), within the <head> element, or, most commonly, in a separate text file that contains only CSS.

**Benefits of CSS**

The benefits of CSS include:

- **Improved control over formatting.** CSS gives fine-grained control over the appearance of their web content.
- **Improved site maintainability.** Websites become significantly more maintainable because all formatting can be centralized into one CSS file. This allows us to make site-wide visual modifications by changing a single file.
- **Improved accessibility.** CSS-driven sites are more accessible. By keeping presentation out of the HTML, screen readers and other accessibility tools work better, thereby providing a significantly enriched experience for those reliant on accessibility tools.
- **Improved page download speed.** A site built using a centralized set of CSS files for all presentation will also be quicker to download because each individual HTML file will contain less style information and markup, and thus be smaller.
- **Improved output flexibility.** CSS can be used to adopt a page for different output media.

**CSS Versions**

- Netscape's proposal was one that required the use of JavaScript programming to perform style changes.
- The W3C decided to adopt CSS, and by the end of 1996 the CSS Level 1 Recommendation was published. A year later, the CSS Level 2 Recommendation (also more succinctly labeled simply as CSS2) was published.
- A different group at the W3C was working on a CSS3 draft.
- To make CSS3 more manageable for both browser manufacturers and web designers, theW3C has subdivided it into a variety of different **CSS3 modules**.

**Browser Adoption**

- Microsoft's Internet Explorer was an early champion of CSS (its IE3, released in 1996, was the first major browser to support CSS, and its IE5 for the Macintosh was the first browser to reach almost 100% CSS1 support in 2000).

- Its later versions (especially IE5, IE6, and IE7) for Windows had uneven support for certain parts of CSS2. However, all browsers have not implemented parts of the CSS2 Recommendation.
- CSS has a reputation for being a somewhat frustrating language. Since CSS was designed to be a styling language, text styling is quite easy.

## 2.2 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:value pairs** (each pair is also called a **declaration**), as shown in Figure 2.1.
- The series of declarations is also called the **declaration block**. A declaration block can be together on a single line, or spread across multiple lines.
- The browser ignores white space (i.e., spaces, tabs, and returns) between our CSS rules so we can format the CSS however we want. Each declaration is terminated with a semicolon.
- The semicolon for the last declaration in a block is in fact optional. However, it is sensible practice to also terminate the last declaration with a semicolon as well.

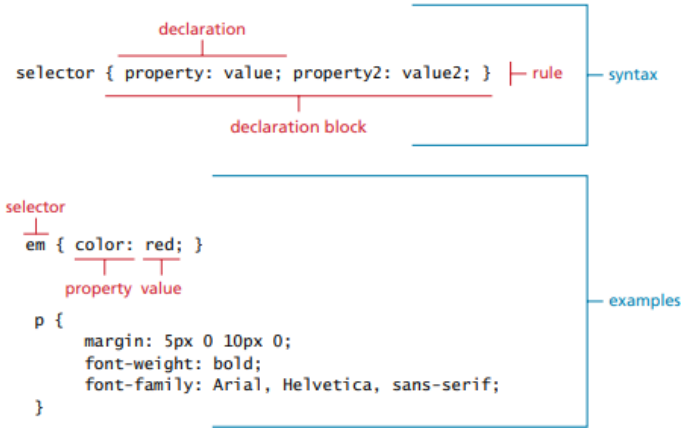



**Figure 2.1:** CSS Syntax

### Selectors

- 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.

## Properties

- Each individual CSS declaration must contain a property.
- These property names are predefined by the CSS standard.
- Table 2.1 lists many of the most commonly used CSS 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. |
| 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 |

**Table 2.1:** Common CSS Properties

## 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.
- Colors would seem at first glance to be the most clear of these units.

| 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 of 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** | This defines a partially transparent background color. The "a" stands for "alpha", which is a term used to identify a transparency that is a value between 0.0 (fully transparent) and 1.0 (fully opaque). | `color: rgb(255,0,0, 0.5);` |
| **HSL** | Allows you to specify a color using Hue Saturation and Light values. This is available only in CSS3. HSLA is also available as well. | `color: hsl(0,100%,100%);`<br>`color: hsl(330,59%,100%);` |

**Table 2.2:** Color Values

Table 2.2 lists the different ways we can describe a color value in CSS just as there are multiple ways of specifying color in CSS, so there are multiple ways of specifying a unit of measurement.

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

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

**Table 2.3:** Units of Measure Values

- Table 2.3 lists the different units of measure in CSS.
- **Relative units** are based on the value of something else or specifies a length relative to another length property, such as the size of a parent element.
- **Absolute units** have a real-world size or it is fixed in relation to each other.
- In general, most of the CSS that we will see uses either px, em, or % as a measure unit.

## 2.3 Location of Styles

- CSS style rules can be located in three different locations. These three are not mutually exclusive, in that we could place our style rules in all three.

### 1. Inline Styles

- **Inline styles** are style rules placed within an HTML element via the style attribute, as shown in Listing 2.1.
- An inline style only affects the element it is defined within and overrides any other style definitions for properties used in the inline style.
- A selector is not necessary with inline styles but semicolons are only required for separating multiple rules.

**Advantages**

- Inline styles are handy for quickly testing out a style change.
- We don't need to create and upload a separate document for css.

**Disadvantages**

- Adding CSS rules to every HTML element is time-consuming and make our HTML structure messy.
- Styling multiple elements can affect our page's size and download time.

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

**Listing 2.1:** Internal styles example

### 2. Embedded Style Sheet

- Embedded style sheets (also called **internal styles**) are style rules placed within the<style> element (inside the <head> element of an HTML document), as shown in Listing 2.2.

**Advantage**

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

**Disadvantages**

- 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.
- Adding the code to the HTML document can increase the page's size and loading time.

```
<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 2.2:** Embedded styles example

### 3. External Style Sheet

- External style sheets are style rules placed within an external text file with the **.css** extension.
- To reference an external style sheet, you must use a <link> element (within the<head> element), as shown in Listing 2.3.

**Advantages**

- The most common place to locate style rules because it provides the best maintainability.
- When you make a change to an external style sheet, all HTML documents that reference that style sheet will automatically use the updated version.
- The browser is able to cache the external style sheet, which can improve the performance of the site as well.
- We can link to several style sheets at a time; each linked style sheet will require its own <link> element.

**Disadvantage**

- Pages may not be rendered correctly until the external CSS is loaded.
- Uploading or linking to multiple CSS files can increase our site's download time

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

**Listing 2.3:** Referencing an external style sheet

## 2.4 Selectors

- When defining CSS rules, we need to first use a selector to tell the browser which elements will be affected by the property values.
- CSS selectors allow us to select individual or multiple HTML elements.
- There are now a variety of new selectors that are supported by most modern browsers.

### Element Selectors

- Element selectors select all instances of a given HTML element. The example CSS rules in Figure 2.1 illustrate two element selectors.
- We can select all elements by using the **universal element selector**, which is the * (asterisk) character.
- We can select a group of elements by separating the different element names with commas as shown in Listing 2.4.

```
/* 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 2.4:** Element/grouped selector

### Class Selectors

- A class selector allows us to simultaneously target different HTML elements regardless of their position in the document tree.
- If a series of HTML elements have been labeled with the same class attribute value, then we can target them for styling by using a class selector, which takes the form: period (.) followed by the class name.
- Listing 2.5 illustrates an example of styling using a class selector. The result in the browser is shown in Figure 2.2

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

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

**Listing 2.5:** Class selector example

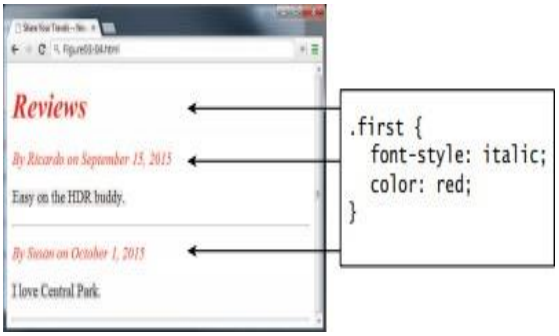



**Figure 2.2:** Class selector example in browser

## Id Selectors

- An **id selector** allows us 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 we can target it for styling by using an id selector, which takes the form: pound/hash (#) followed by the id name.
- Listing 2.6 illustrates an example of styling using an id selector. The result in the browser is shown in Figure2.3.

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

**Listing 2.6:** Id selector example

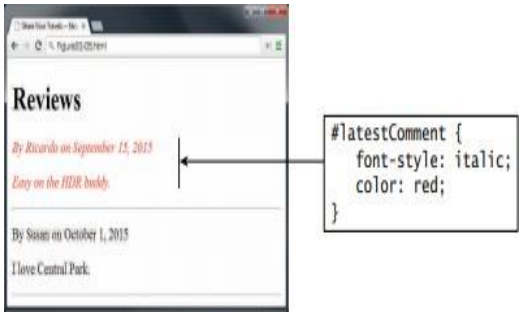



**Figure 2.3:** Id selector example in browser

## Attribute Selectors

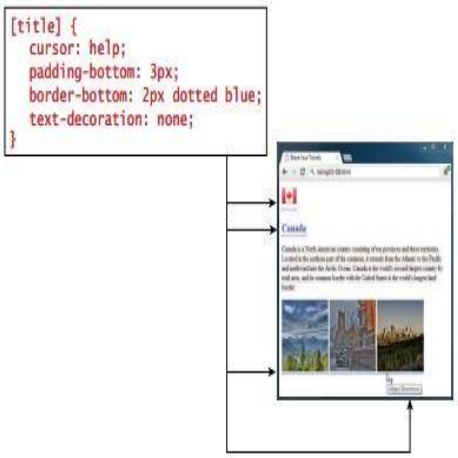
- An **attribute selector** provides a way to select HTML elements either by the presence of an element attribute or by the value of an attribute.

- Attribute selectors can be a very helpful technique in the styling of hyperlinks and images.
- We can do this by using the following attribute selector:

[title] { … }

- This will match any element in the document that has a title attribute.
- Table 2.4 summarizes some of the most common ways one can construct attribute selectors in CSS3. Listing 2.7, with the results in the browser shown in Figure 2.4.

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




**Listing 2.7:** Attribute selector example

**Figure 2.4:** Attribute selector example in browser

| Selector | Matches | Example |
|---|---|---|
| [] | A specific attribute. | `[title]`<br>Matches any element with a title attribute |
| [=] | A specific attribute with a specific value. | `a[title="posts from this country"]`<br>Matches any <a> element whose title attribute is exactly "posts from this country" |
| [~=] | A specific attribute whose value matches at least one of the words in a space-delimited list of words. | `[title~="Countries"]`<br>Matches any `title` attribute that contains the word "Countries" |
| [^=] | A specific attribute whose value begins with a specified value. | `a[href^="mailto"]`<br>Matches any <a> element whose `href` attribute begins with "mailto" |
| [*=] | A specific attribute whose value contains a substring. | `img[src*="flag"]`<br>Matches any <img> element whose `src` attribute contains somewhere within it the text "flag" |
| [$=] | A specific attribute whose value ends with a specified value. | `a[href$=".pdf"]`<br>Matches any <a> element whose `href` attribute ends with the text ".pdf" |

**Table 2.4:** Attribute Selectors

**Pseudo-Element and Pseudo-Class 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. For instance, we can select the first line or first letter of any HTML element using a pseudo-element selector.
- A **pseudo-class selector** does apply to an HTML element, but targets either a particular state or, in CSS3, a variety of family relationships.
- Table 2.5 lists some of the more common pseudo-class and pseudo element selectors. By default, the browser displays link text blue and visited text links purple.
- Listing 2.8 illustrates the use of pseudo-class selectors to style not only the visited and unvisited link colors, but also the hover color, which is the color of the link when the mouse is over the link.
- Note the syntax of pseudo-class selectors: the colon (:) followed by the pseudo-class selector name. Do be aware that a space is *not* allowed after the colon. The order of these pseudo-class elements is important.

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

```
<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 2.8:** 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.
- 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. Figure 2.5 illustrates the syntax and usage of the syntax of the descendant selector.
- Table 2.6 describes the other contextual selectors. Figure 2.6 illustrates some sample uses of a variety of different contextual selectors.

**Figure 2.5:** Syntax of a descendant selection

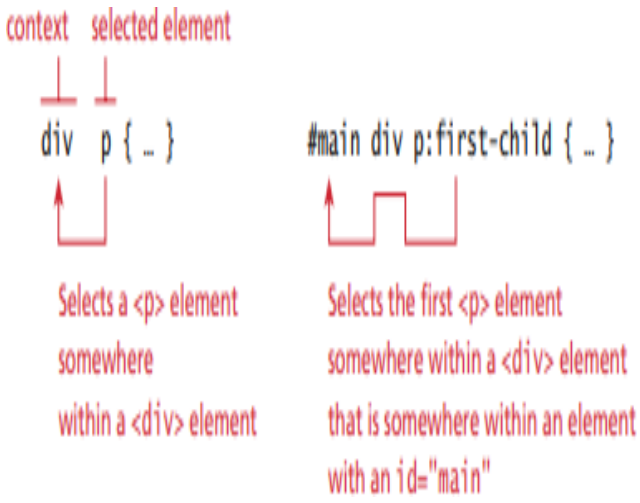



**Table 2.6:** Contextual Selectors

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

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

      <div>
         <p>By Susan on <time>October 1, 2015</time></p>
         <p>I love Central Park.</p>
      </div>
      <hr/>
   </div>
   <footer>
      <ul>
         <li><a href="#">Home</a> | </li>
         <li><a href="#">Browse</a> | </li>
      </ul>
   </footer>
</body>
```

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

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

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

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

**Figure 2.6:** Contextual selectors in action

## 2.5 The Cascade: How Styles Interact

- There are three different types of style sheets: author-created, user-defined, and the default browser style sheet. As well, it is possible within an author-created style sheet to define multiple rules for the same HTML element
- CSS has a system to help the browser determine how to display elements when different style rules conflict. The "Cascade" in CSS refers to how conflicting rules are handled.
- CSS uses the following cascade principles to help it deal with conflicts: inheritance, specificity, and location.

### Inheritance

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

- Figures 2.7 illustrate CSS inheritance. In the first example, only some of the property rules are inherited for the <body> element. That is, only the body element (thankfully!) will have a thick green border and the 100-px margin.

- In the second example in Figure 2.8, we can assume there is no longer the body styling but instead we have a single style rule that styles *all* the  elements. The <p> and<time> elements within the  inherit the bold font-weight property but not the margin or border styles.

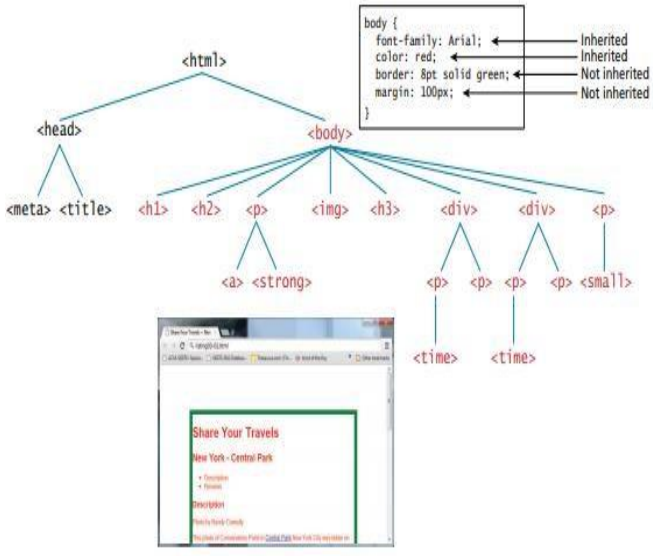



**Figure 2.7:** Inheritance

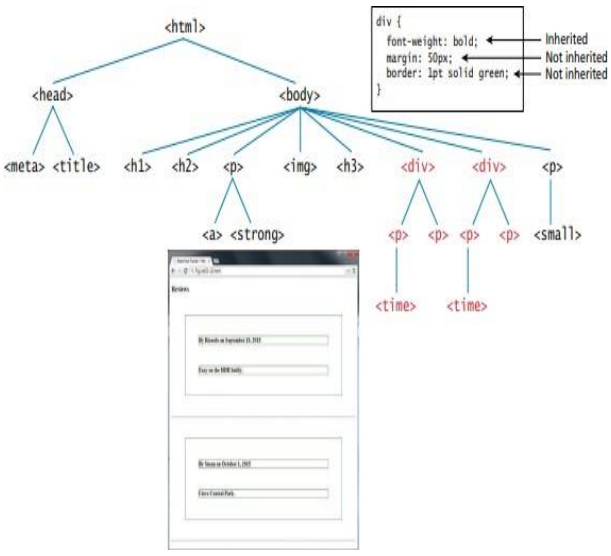



**Figure 2.8:** More Inheritance

- However, it is possible to tell elements to inherit properties that are normally not inheritable.

## Specificity

- **Specificity** is how the browser determines which style rule takes precedence when more than one style rule could be applied to the same element.
- In CSS, the more specific the selector, the more it takes precedence (i.e., overrides the previous definition).
- Another way to define specificity is by telling us 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.
- Figure 2.9, the color and font-weight properties defined in the <body> element are inheritable and thus potentially applicable to all the child elements contained within it.
- The and <p> elements also have the same properties set, they *override* the value defined for the<body> element because their selectors ( and <p>) are more specific.
- Figure 2.9, class selectors take precedence over element selectors, and id selectors take precedence over class selectors.

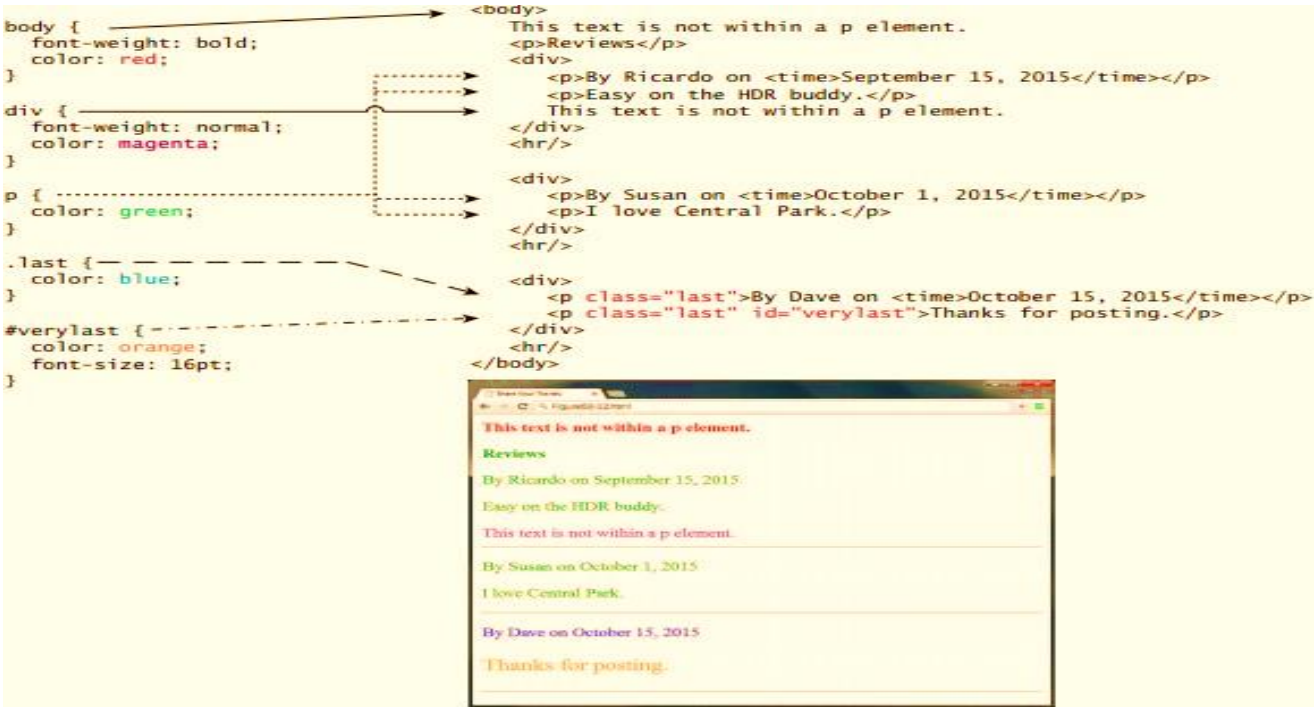



**Figure 2.9:** Specificity

**Location**

- When inheritance and specificity cannot determine style precedence, the principle of **location** will be used.
- The principle of location is that when rules have the same specificity, then the latest are given more weight. For instance, an inline style will override one defined in an external author style sheet or an embedded style sheet.
- Similarly, an embedded style will override an equally specific rule defined in an external author style sheet if it appears after the external sheet's <link> element.
- Similarly, when the same style property is defined multiple times within a single declaration block, the last one will take precedence as shown in Figure 2.10.

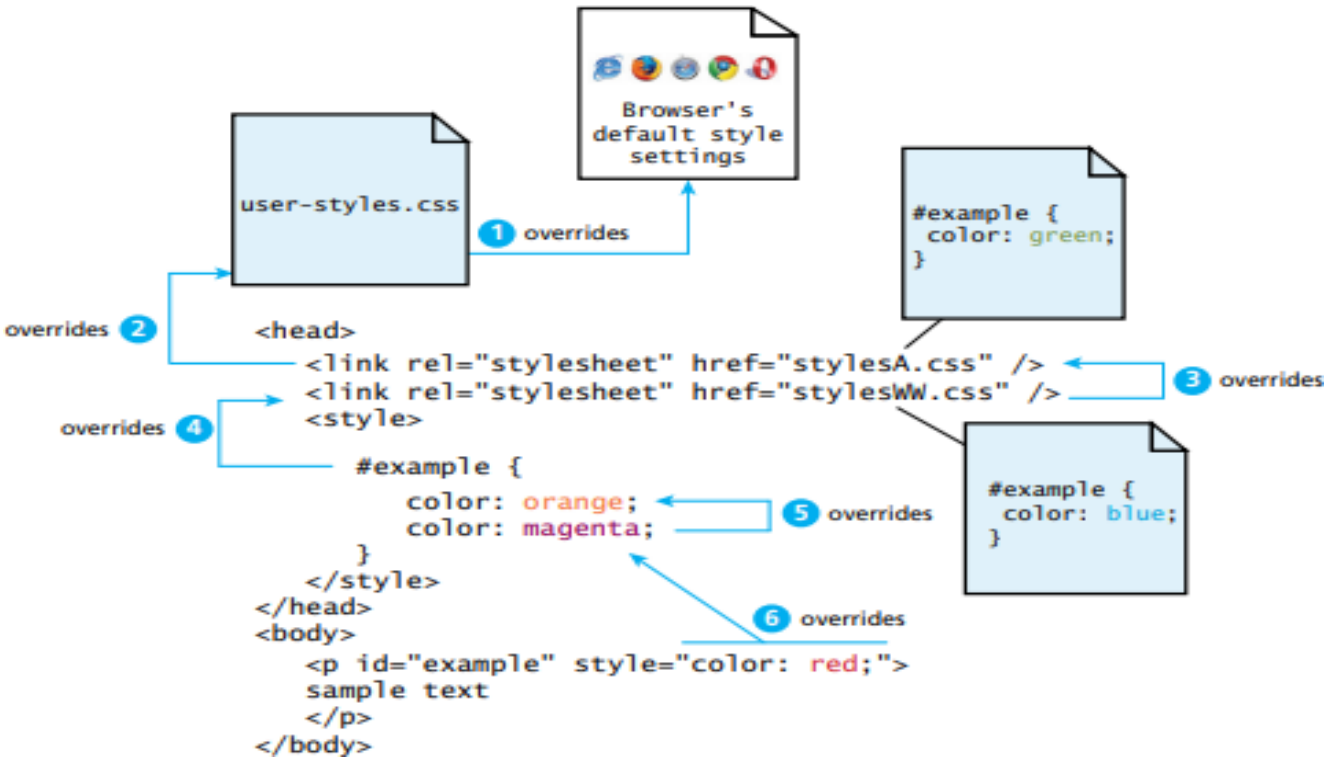



**Figure 2.10:** Location

## 2.6 The Box Model

In CSS, all HTML elements exist within an **element box** shown in Figure 2.11.

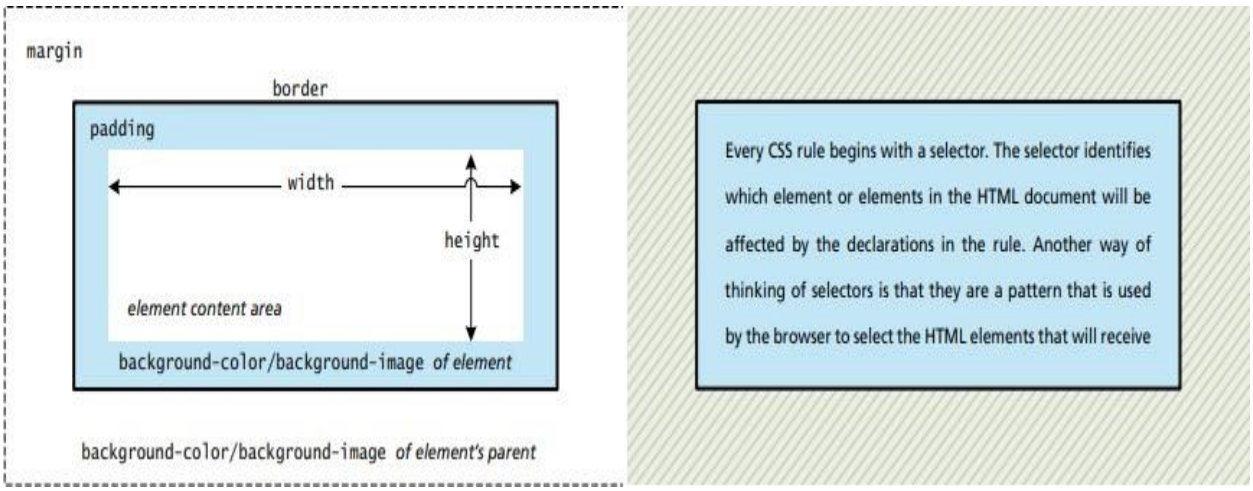



**Figure 2.11:** CSS box model

### Background

- The background color or image of an element fills an element out to its border.
- In contemporary web design, it has become extremely common to use CSS to display purely presentational images (such as background gradients and patterns, decorative images, etc.) rather than using the <img> element.
- Table 2.7 lists the most common background properties

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

**Table 2.7:** Common Background Properties

## Borders

- Borders provide a way to visually separate elements. We can put borders around all four sides of an element, or just one, two, or three of the sides. Table 2.8 lists the various border properties.

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

**Table 2.8:** Border Properties

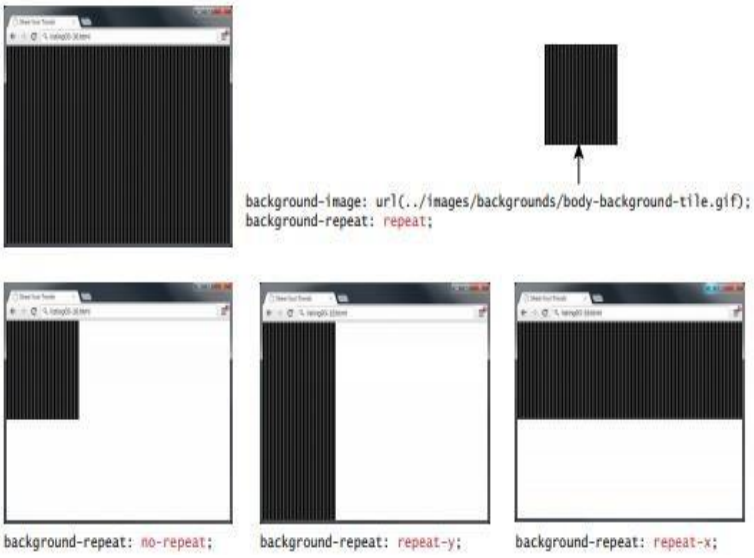



**Figure 2.12**: Background repeat

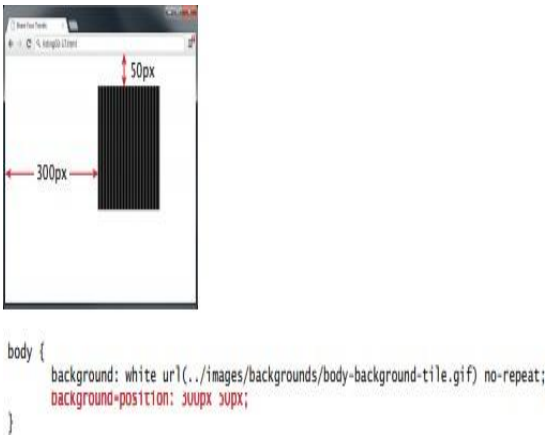



**Figure 2.13:** Background position

## Margins and Padding

- Margins and padding are essential properties for adding white space to a web page, which can help differentiate one element from another.
- Figure 2.14 illustrates how these two properties can be used to provide spacing and element differentiation.
    - **Margins** add spacing around an element's content,

- while **padding** adds spacing within elements.

- Borders divide the margin area from the padding area.

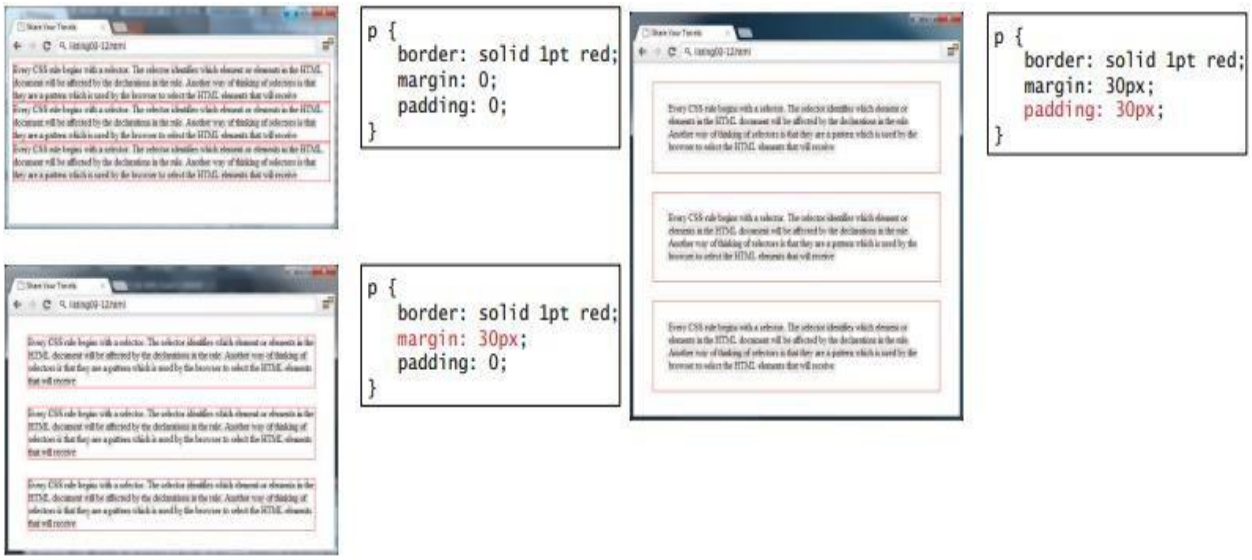



**Figure 2.14**: Borders, margins, and padding provide element spacing and differentiation

- Figure 2.15 illustrates how adjoining vertical margins collapse in the browser.
- If overlapping margins did not collapse, then margin space for 2 would be 180 px (90 px for the bottom margin of the first  + 90 px for the top margin of the second ), while the margins for 4 and 5 would be 100 px.
- The W3C specification defines this behavior as **collapsing margins**:
  - In CSS, the adjoining margins of two or more boxes (which might or might not be siblings) can combine to form a single margin. Margins that combine this way are said to collapse, and the resulting combined margin is called a collapsed margin.

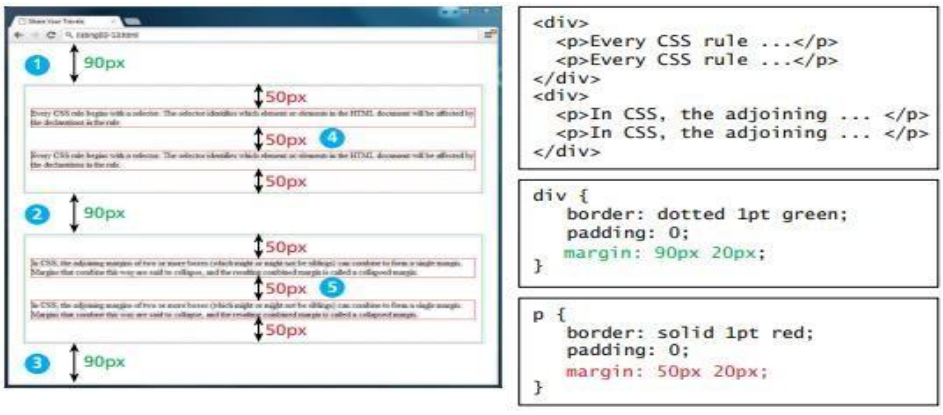



**Figure 2.15:** Collapsing vertical margins

- The **vertical** margins of two elements touch, only the largest margin value of the elements will be displayed, while 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.

## Box Dimensions

- By default (in CSS this is the auto value), 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.
- Since the width and the height only refer to the size of the content area, the total size of an element is equal to the size of its content area plus the sum of its padding, borders, and margins.

**height + padding + border = actual height of an element**

**width + padding + border = actual width of an element**

- Figure 2.16 illustrates the default content-box element sizing behavior.

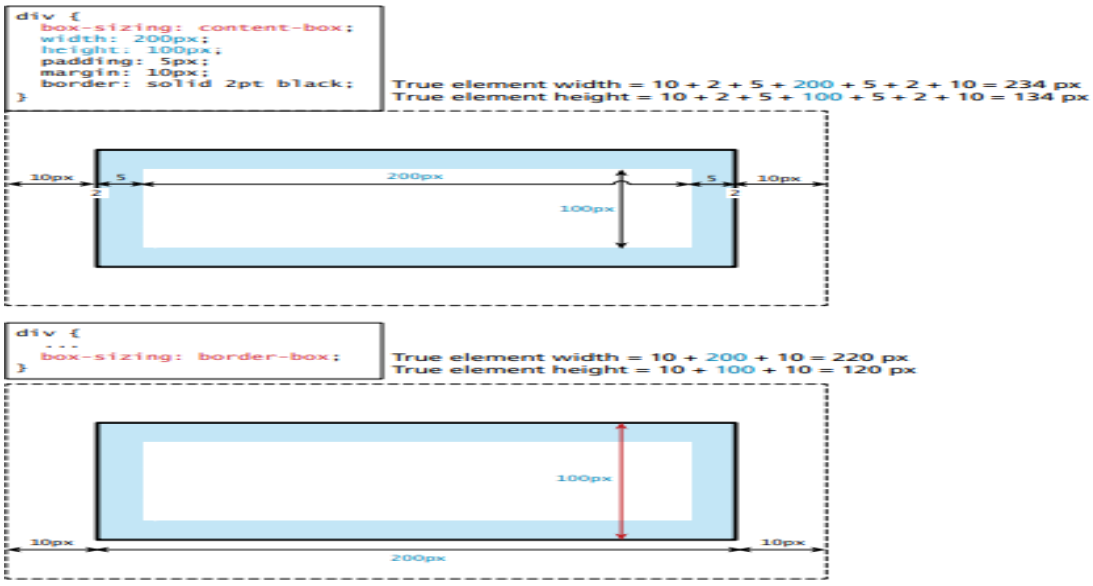



**Figure 2.16:** Calculating an element's true size

- For block-level elements such as <p> and  elements, there are limits to what the width and height properties can actually do.
- We can shrink the width, but the content still needs to be displayed, so the browser may very well ignore the height that we set.

- Figure 2.17, the default width is the browser viewport. But in the second screen capture in the image, with the changed width and height, there is not enough space for the browser to display all the content with in the element. The height of the actual textual content is much larger (depending on the font size).

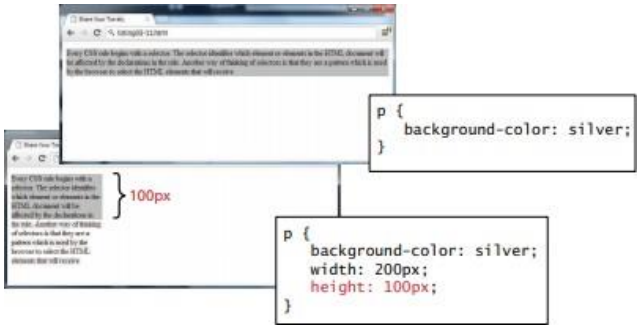



**Figure 2.17:** Limitations of height property

- It is possible to control the content if the box's width and height are not large enough to display the content using the overflow property, as shown in Figure 2.18.

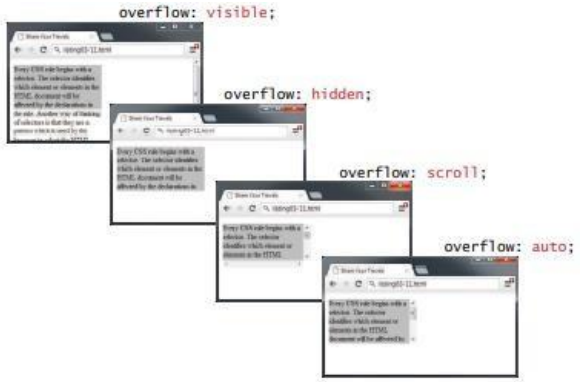



**Figure 2.18:** overflow property

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

## 2.7 CSS Text Styling

- CSS provides two types of properties that affect text.
  - The first we call font properties because they affect the font and its appearance.

- The second type of CSS text properties are referred to here as paragraph properties since they affect the text in a similar way no matter which font is being used.

## Font Family

- A word processor on a desktop machine can make use of any font that is installed on the computer; browsers are no different.
- 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 is not on the user's computer.
- Figure 2.19, the alternatives are separated by commas; as well, if the font name has multiple words, then the entire name must be enclosed in quotes.

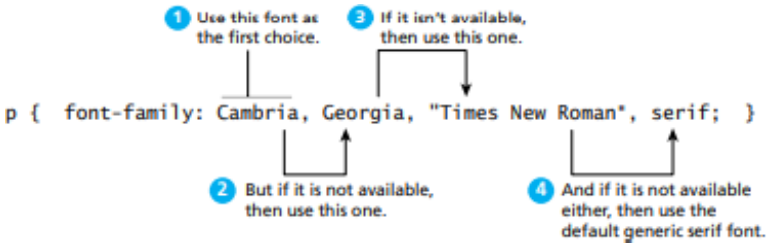



**Figure 2.19**: Specifying the font family

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

**Table 2.9:** font properties

- The font-family property supports five different generic families; the browser supports a type face from each family. The different generic font families are shown in Figure 2.20.

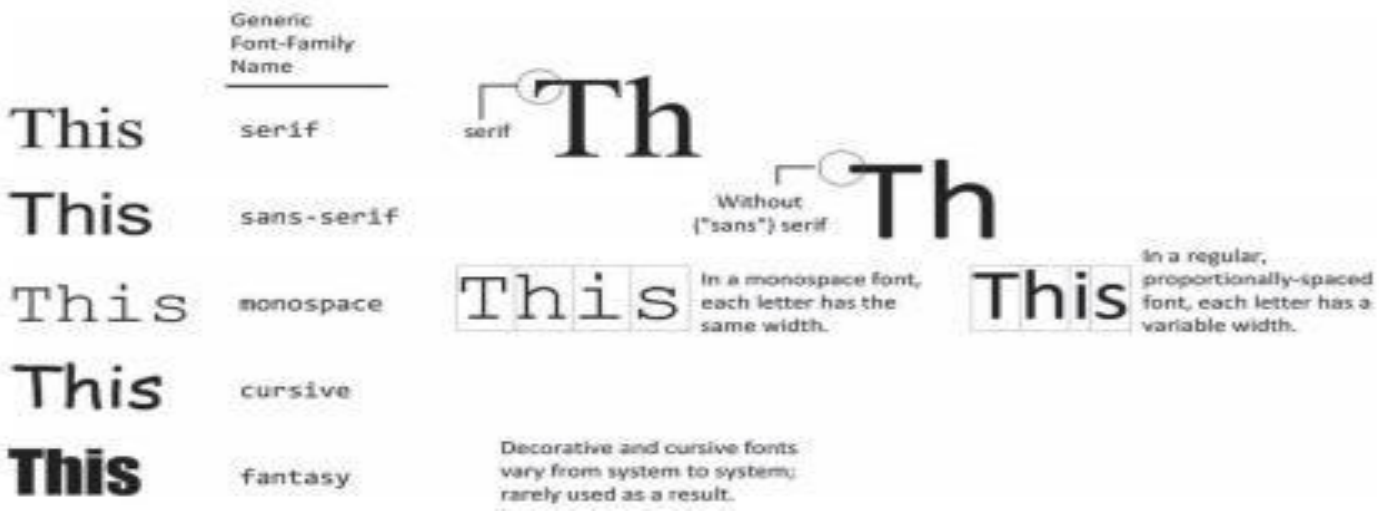



**Figure 2.20:** The different font families

**Font Sizes**

- 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," and not break the page layout.
- When used to specify a font size, both em units and percentages are relative to the parent's font size. This takes some getting used to.
- Figure 2.21 illustrates a common set of percentages and their em equivalents to scale elements relative to the default 16-px font size

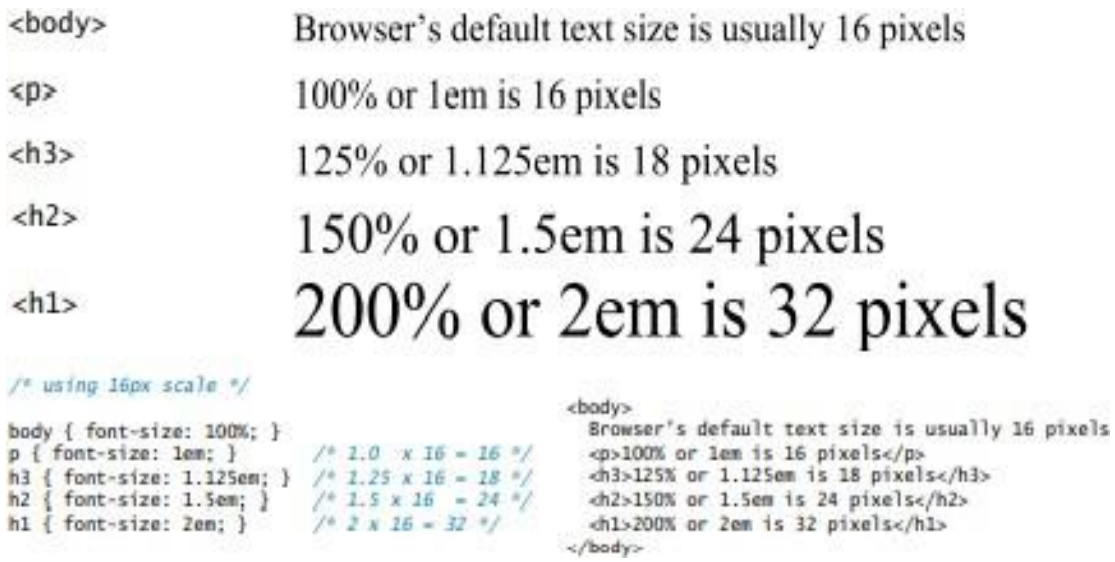



**Figure 2.21:** Using percents and em units for font sizes

- percents and em units are relative to their parents. Figure 2.22 illustrates how in reality it can quickly become difficult to calculate actual sizes when there are nested elements.

```
<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 */
```

**Figure 2.22:** Complications in calculating percents and em units

- To avoid complications, CSS3 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 element).


```
/* 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 */
```

**Figure 2.23:** Using rem units

## Paragraph Properties

- Just as there are properties that affect the font in CSS, there are also a range of CSS properties that affect text independently of the font.
- Many of the most common text properties are shown in Table 2.10, and like the earlier font properties, many of these will be familiar to anyone who has used a word processor

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

**Table 2.10:** Text Properties

**********End of Module_1**********