What is Full Stack Web Development? Layers of Full Stack Development

Full stack development encompasses the complete process of application software development, including both the front-end and back-end development. The front end consists of the user interface (or UI), and the back end handles the business logic and application workflows that run behind the scenes.

Full stack developers possess the skills and knowledge to work across the entire technology stack, enabling seamless user experiences and developing robust backends.

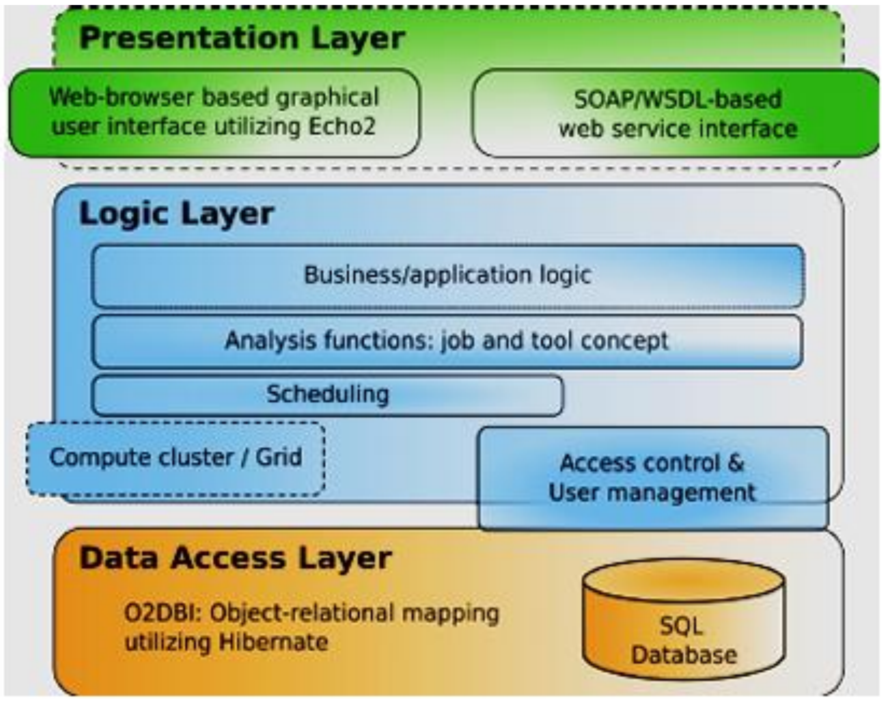
Layers of Full Stack Development

Full Stack Web Development speaks of the development of both the front end and back end of web applications or web applications. The Full Stack web development process consists of three layers, which include:

The presentation layer: This consists of the front end of the application. This means the HTML, CSS, and JavaScript that users interact with. Other application types are mobile applications, desktop applications, and voice interfaces. Nevertheless, sticking with strictly Web applications, is where you'll come across the Web page that the client works with.

The logic layer: This consists of the back end of the application. It should safely retrieve and place data requested or given from the consuming client and pass it along to the data layer. This often involves more complicated tasks like authentication, authorization, API design, or creating the logic to implement business logic.

The database layer: The database layer, which is also recognized as the data tier, stores all the information connected to user profiles and transactions. Principally, it consists of any data that needs to persist in being stored in the data tier


Presentation Layer
Web-browser based graphical user interface utilizing Echo2
SOAP/WSDL-based web service interface
Logic Layer
Business/application logic
Analysis functions: job and tool concept
Scheduling
Compute cluster / Grid
Access control & User management
Data Access Layer
O2DBI: Object-relational mapping utilizing Hibernate
SQL Database

CONCEPTS OF THE WEB EVERY DEVELOPER NEEDS TO KNOW

WEB:

The web is also known as Word Wide Web (WWW) or W3, it's an interconnected system of public webpages usually known as servers. The WWW is a web technology that Links web resources together over the internet. The Web is not the same as the Internet, but the Web is one of many applications built on top of the Internet because the internet housed the Web.

URL:

Uniform Resource Locator (URL) also known as Web address is a reference to a web resource that specifies its location on a computer network and a gateway for retrieving it. A typical example of a URL is https://www.google.com. Https represent the Transfer Control Protocol, while Google.com is the URL for Google.

HTTPS:

Hypertext Transfer Protocol Secured is a secured protocol that governs the transfer of Web resources. While Hypertext Transfer Protocol is a none secured protocol that governs the transfer of Web resources. You can identify between this two when surfing the internet. Secured protocol usually puts a visible padlock before the address bar of the Web browser while none secured does not.

WEBSITE:

A website is said to be a collection of related web pages with common web resources and share the same domain name. A web page may consist of text, images, audio, video animations, and other related resources. These resources could be static or dynamic. There are primarily two (2) types of websites; Static and Dynamic Websites.

DOMAIN NAME:

Domain is simply your website address. It is a unique name that identifies resources stored on a website. It's a unique name that houses your website. Because of the above definition, two websites cannot have the same name, but a website can have a primary domain and a sub-domain. The primary domain is said to be the main domain name of a website. For instance, facebook.com, google.com, etc. While the sub-domain is a secondary domain to the primary domain. For instance app.facebook.com, app.google.com, etc.

HOSTING:

Hosting serves as the land upon which your website is built. Web hosting servers are the space where your website is built and stored which is accessible on the web.

SERVER:

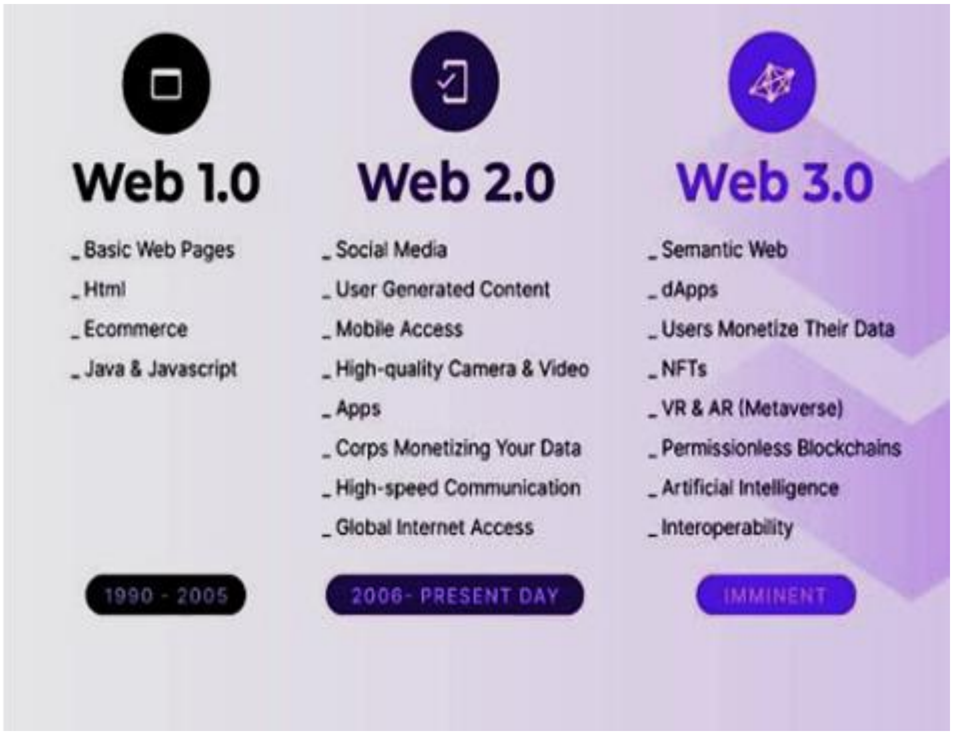
A server is a software or hardware device that accepts and responds to requests made over a network. The device that makes the request, and receives a response from the server, is called a client. On the Internet, the term "server" commonly refers to the computer system that receives requests for a web file and sends those files to the client. Servers manage network resources. For example, a user may set up a server to control access to a network, send/receive an e mail, manage print jobs, or host a website. They are also proficient at performing intense calculations. Some servers are committed to a specific task, often referred to as dedicated. However, many servers today are shared servers that take on the responsibility of e mail, DNS, FTP, and even multiple websites in the case of a web server

UNDERSTANDING THE TYPES OF WEB

1. Web 1.0: Web 1.0, also called the Static Web, was the first available and most reliable internet in the 1990s notwithstanding the fact it gives access to limited information with little to no user interaction. Back in the day, creating user pages or even remarking on articles wasn't possible since it was not built with such a feature. Web 1.0 didn't have algorithms to sift internet pages and bring them together in a dynamic way, which made it extremely hard for users to find relevant information. Simply put, it was like a one-way highway with a narrow footpath where content creation was done by a select few algorithms and information came mostly from directories where they are stored.

2. Web 2.0: Web 2.0, also known as Social Web, or Web 2.0, made the internet a lot more interactive. Here, messages can be exchanged on a real-time basis. The advancements in web technologies like JavaScript, HTML5, CSS3, etc. It enabled startups to build interactive web platforms such as YouTube, Facebook, Wikipedia, and lots more. This paved the way for both social networks and user-generated content production to flourish since businessmen and women can run business adverts on those platforms, thereby promoting their products and services. Data can now be distributed and shared between various platforms and applications.

3. Web 3.0: Web 3.0 is the third generation of the internet where websites and applications will be able to process information in a smart human-like way through technologies like Big Data, machine learning (ML), decentralized ledger technology (DLT), etc. Web 3.0 was originally called the Semantic Web by World Wide Web inventor Tim Berners Lee, and was aimed at being a more autonomous, intelligent, and open internet. The Web 3.0 definition can be expanded as follows: data will be interconnected in a decentralized way, which would be a huge leap forward to our current generation of the internet unlike (Web 2.0), where data is mostly stored in centralized repositories or systems. The two functions of Web 3.0 is the semantic web and artificial intelligence (AI). Web 3.0 is the succeeding stage of the web evolution that would make the internet more intellectual or process information with near-human like brainpower through the power of AI systems that can run smart programs to assist users.



Git, GitHub, and Version Control

What is GitHub? GitHub is a website and cloud-based technology that helps developers store and manage their code, as well as keep track and control modifications to their code. To understand exactly what GitHub is, you need to understand Version control and Git principles. Understanding Git Technology Git is a Distributed Version Control System (DVCS) used to preserve different versions of a file or set of files so that any of the saved versions are retrievable at will. Git also makes it stress-free to record and compare, unlike file versions. This means that the details about what is to be changed, who does the change, or who instigated an issue are reviewable anytime.

What does distributed mean? The term distributed means that each time you instruct Git to distribute or share a project's directory with the individual in view, Git does not only share the up-to-date file version. Instead, it distributes every version it has documented for that project. This distributed system is a sharp difference from other version control systems. They only distribute whatever single version a user has openly checked out from the principal or local database. Consequently, distributing means distributing all files and not just the selected few versions of a project's files that Git has documented.

Understanding files states in Git

Git has three primary states or conditions in which a file can be, and they are explained below.

1. Modified state: A file in the modified state is a revised file but uncommitted is an unrecorded file. In other words, files in the modified state are files you have revised but have not authorized Git to monitor.

2. Staged state: Files in the staged state are modified files that have been carefully selected in their current state or version and are being organized to be saved or committed into the .git storehouse or repository during the next commit caption. Once a file gets staged, it indicates that you have clearly instructed Git to monitor that file's version.

3. Committed state: Files in the committed state are files that are successfully warehoused into the .git repository. Hence, a committed file is a file in which you have recorded its staged version into the Git directory or folder. (NB: The state of a file defines the location where Git will place it).

Follow the steps below to host or share a Git repository on GitHub.

1. Signup for a GitHub account.
2. Create a remote repository in GitHub.
3. Connect the project's Git directory to the remote repository.
4. Confirm the connection.
5. Push a local Git repo to the remote repo

What is Version Control?

A Version Control System (VCS) refers to the technique used to save a file's versions for future use. Version control is a technology that helps developers track and manage modifications to a software project's code. As a project grows, version control becomes indispensable. With branching, a developer can duplicate part of the source code called the repository. After this, the developer can then safely make modifications to that part of the code without changing the rest of the project.

At that time, the developer gets his or her part of the code working appropriately, he or she can merge that code back into the main source code to make it official. All of these changes are then tracked and can be reverted if need be.

Unthinkingly, several people already version control their project's work by retitling different versions of the same file in different ways like storeScript.js, storeScript_v2.js, storeScript_v3.js, storeScript_final .js, storeScript_definite_final.js, etc. But this approach is prone to error and unproductive for team projects. Similarly, tracking what is being changed, who changed it, and why it was changed is a boring endeavor with this old-style approach. This lightens the significance of a dependable and collaborative version control system like Git.

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.
- 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** (or more formally, HTML elements, but more on that later).
- The markup in Figure 2.1 consists of the red text and the various circles and arrows and the little yellow sticky notes. HTML does the same thing but uses textual tags.
- Markup languages are able to encode information how to display the content for the end user.
- The presentation semantics can be as simple as specifying a bold weight font for certain words, and were a part of the earliest HTML specification.

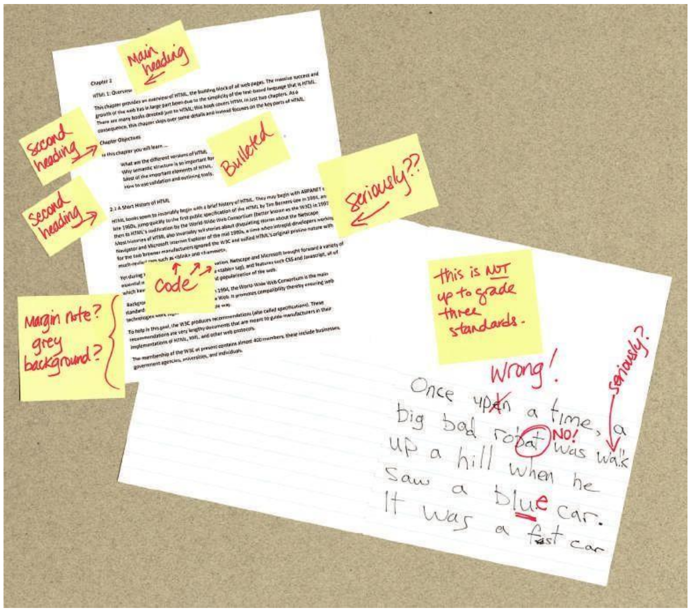

Main heading
Second heading
Bulleted
Second heading
Seriously??
Code
this is NOT up to grade three standards.
Margin note? grey background?
Wrong!
NO!
seriously?
Once upon a time, a big bad robot was walk up a hill when he saw a blue car. It was a fast car

Figure 2.1 : Sample ad-hoc markup languages

XHTML

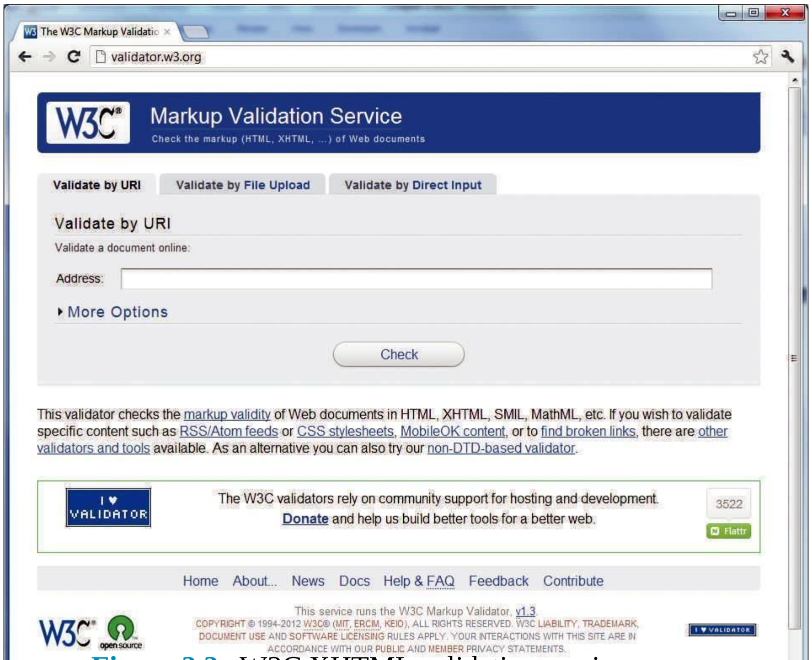
- The goal of XHTML with its strict rules was to make page rendering more predictable by forcing web authors to create web pages without **syntax errors**.
- To help web authors, two versions of XHTML were created: **XHTML 1.0Strict** and **XHTML 1.0 Transitional**. The strict version was meant to be renderedby a browser using the strict syntax rules and tag support described by the W3C XHTML 1.0 Strict specification.
- XML is a more general markup language than HTML. It is (and has been) used to mark up any type of data. XML-based data formats (called **schemas** in XML) are almost everywhere.
- Rss data feeds use XML and Web 2.0 sites often use XML data formats to move data back and forth asynchronously between the browser and the server. 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>
```

- By and large, the XML-based syntax rules (called “well-formed” in XML lingo) 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.



This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available. As an alternative you can also try our non-DTD-based validator.

The W3C validators rely on community support for hosting and development. Donate and help us build better tools for a better web.

This service runs the W3C Markup Validator, v1.3.

COPYRIGHT © 1994-2012 W3C® (MIT, ERCIM, KEIO), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.

Figure 2.2 W3C XHTML validation service

- **HTML validators** means verifying that a web page's markup followed the rules for XHTML Transitional or Strict.
- In the mid-2000s, the W3C presented a draft of the XHTML 2.0 specification. It proposed a revolutionary and substantial change to HTML. The most important was

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

- The XHTML 2.0 specification also dropped familiar tags such as <img>, <a>, <br>, and numbered headings such as <h1>.

HTML5

- At around the same time the XHTML 2.0 specification was being developed, 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:
  - Specify unambiguously how browsers should deal with invalid markup.
  - Provide an open, nonproprietary programming framework (via JavaScript) for creating rich web applications.
  - Be backwards compatible with the existing web.
  - While parts of the HTML5 are still being finalized, all of the major browser manufacturers have at least partially embraced HTML5.
  - 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.

HTML Syntax

- The current W3C Recommendation for HTML is the HTML 4.01 specification, which dates back all the way to 1999.
- The syntax for marking up documents was defined and centered around using elements and attributes.
- Learning the fundamental concepts and terms that have survived multiple standards is essential in a discipline like web development where specifications, standards, and browsers are constantly evolving.

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 a more expansive term that encompasses the element name within angle brackets (i.e., the tag) and the content within the tag (though some elements contain no extra content).
- 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.
- 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.

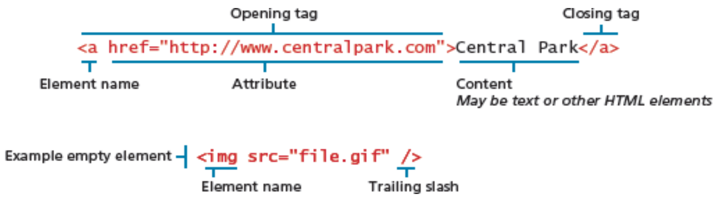


Figure 2.4 The parts of an HTML element

- In XHTML, empty elements had to be terminated by a trailing slash). In HTML5, the trailing slash in empty elements is optional.

Nesting HTML Elements

- HTML element will contain other HTML elements. The such ,container element is said to be a parent of the contained, or child, element.

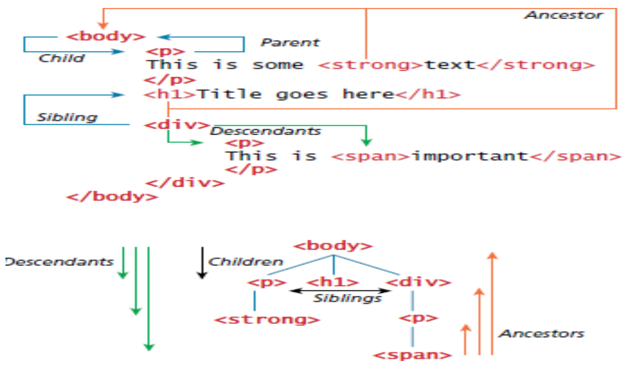


Figure 2.5 HTML document outline

- 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 is not case sensitive for element or attribute names.
- This underlying family tree or hierarchy of elements **Cascading Style Sheets** (CSS) and JavaScript programming and parsing.
- This concept is called the **Document ObjectModel** (DOM) formally, though for now we will only refer to its hierarchical aspects.
- In order to properly construct this 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, as shown in Figure 2.6.

Correct nesting

<h1>Share Your<strong>Travels</strong></h1>

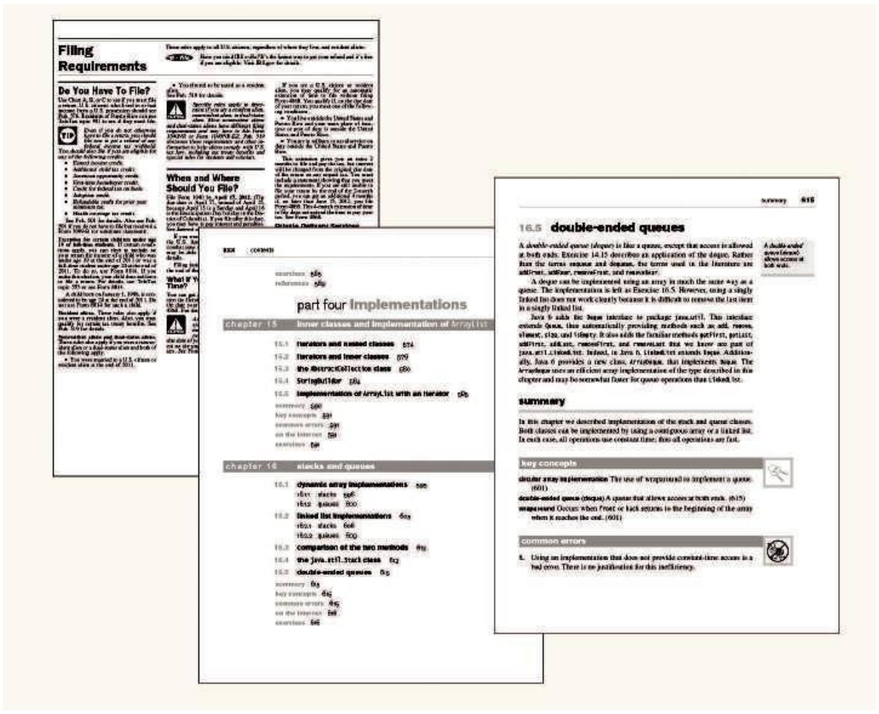
Incorrect nesting

<h1>Share Your<strong>Travels</h1></strong>

Figure 2.6 The proper nesting of HTML elements

2.3 Semantic Markup

- Information about how the content should look when it is displayed in the browser is best left to CSS (Cascading Style Sheets).
- **Semantic HTML** documents is an HTML document should not describe how to visually present content, but only describe its content's structural semantics or meaning. This advice might seem mysterious, but it is actually quite straightforward.
- Figure 2.7, One is a page from the United States IRS explaining the 1040 tax form; another is a page from a textbook.



Filing Requirements

Do You Have To File?

When and Where Should You File?

part four implementations

chapter 15 inner classes and implementation of ArrayList

chapter 16 stacks and queues

16.5 double-ended queues

summary

key concepts

common errors

Figure 2.7 Visualizing structure

- In each of them, you will notice that the authors of the two documents use similar means to demonstrate to the reader the structure of the document.
- That structure 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.

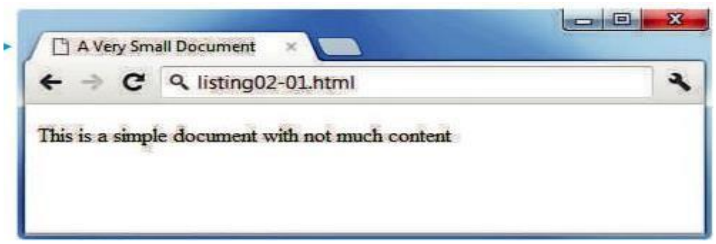
- Eliminating presentation-oriented markup and writing semantic HTML markup has a variety of important advantages:

1. **Maintainability**. Semantic markup is easier to update and change than web pages that contain a great deal of presentation markup. This is even truer with web projects. From our experience, web projects have a great deal of "requirements drift" due to end user and client feedback than traditional software development projects.
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. As well, many governments insist that sites for organizations that receive federal government funding must adhere to certain accessibility guidelines. For instance, the United States government has its own Section 508 Accessibility Guidelines (**http://www.section508.gov**).

2.4 Structure of HTML Documents

- Figure 2.8 illustrates one of the simplest *valid* HTML5 documents you can create.
- As can be seen in the corresponding capture of the document in a browser, such a simple document is hardly an especially exciting visual spectacle.
- The <title> element 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 2.8.
- The title has some additional uses that are also important to know. The title is used by the browser for its bookmarks and its browser history list.
- For readers with some familiarity with XHTML or HTML 4.01, this listing will appear to be missing some important elements. Indeed, in previous versions, a valid HTML document required additional structure.
- Figure 2.9 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>
```

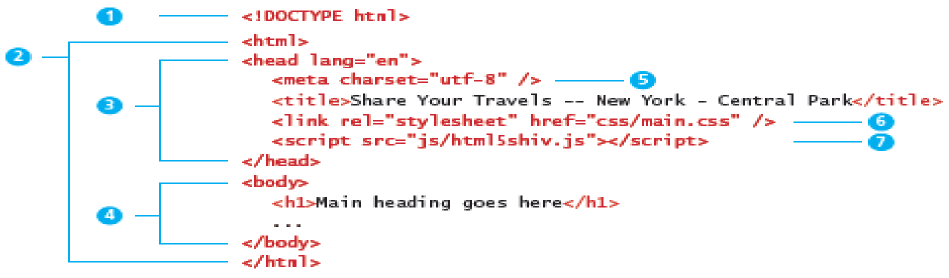


This is a simple document with not much content

Figure 2.8: One of the simplest possible HTML5 documents

2.4.1 DOCTYPE

- Figure 2.9 points to the DOCTYPE (short for **Document Type Definition**) element, which tells the browser (or any other client software that is reading this HTML document) what type of document it is about to process.



```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8" />
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="css/main.css" />
   <script src="js/html5shiv.js"></script>
</head>
<body>
   <h1>Main heading goes here</h1>
   ...
</body>
</html>
```

FIGURE 2.9 Structure elements of an HTML5 document

- The 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">

- The XHTML doctype instructed the browser to follow XHTML rules. In the early years of the 2000s, not every browser followed the W3C specifications for HTML and CSS; as support for standards developed in newer browsers,

- The doctype was used to tell the browser to render an HTML document using the so- called **standards mode** algorithm or render it with the particular browser's older non standards algorithm, called **quirks mode**.
- 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.
- The following code from the official XHTML DTD defines the syntax of the <p> element:

```
<!ELEMENT p %Inline;>
<!ATTLIST p
  %attrs;
  %TextAlign;
>
```

Within XML, DTDs have largely been replaced by XML schema.

2.4.2 Head and Body

- HTML5 does not equire the use of the <html>, <head>, and <body> elements. 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.
- Notice that it also has a long attribute. This optional attribute tells the browser the natural language that is being used for textual content in the HTMLdocument, 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.
- Every character in a standard text document is represented by a standardized bit pattern. The original ASCII standard of the 1950s defined in upper and lowercase letters as well as a variety of common punctuation symbols using 8 bits for each character.
- Figure 2.9 specifies an external CSS style sheet file that is used with this document. Virtually all commercial web pages created in the last decade 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. Notice that in this example, the file being referenced (**main.css**) resides within a subfolder called **css**.
- Figure 2.9 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.

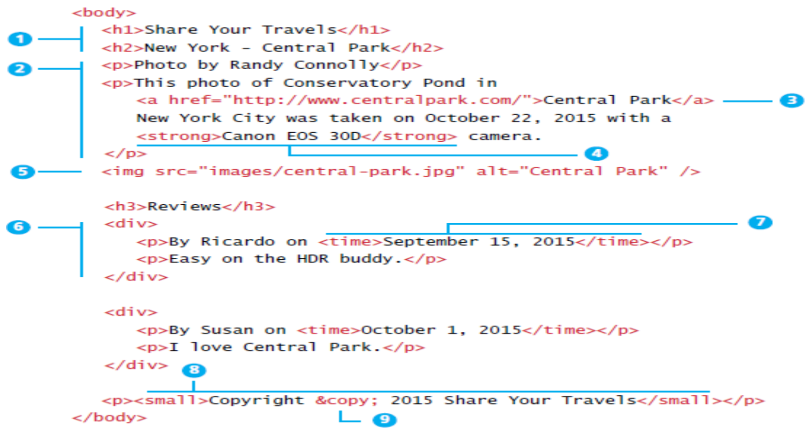
2.5 Quick Tour of HTML Elements

- Figure 2.10 contains the HTML we will examine in more detail (note that some of the structural tags like <html> and <body> from the previous section are omitted in this example for brevity's sake).
- Figure 2.11 illustrates how the markup in Figure 2.10 appears in the browser.

2.5.1 Headings

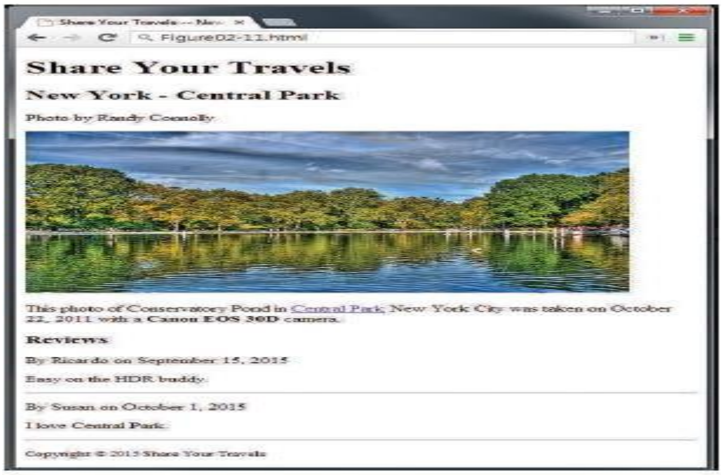
- Figure 2.10 defines two different headings. HTML provides six levels of heading (h1 through h6), with the higher heading number indicating a heading of less importance.

- In Figure 2.7, the 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.
- The internal data representation of the control on the page. This document outline is used by the browser to render the page. It is also potentially used by web authors when they write JavaScript to manipulate elements in the document or when they use CSS to style different HTML elements.



```
<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 2.10 Sample HTML5 document



Share Your Travels

New York - Central Park

Photo by Randy Connolly

This photo of Conservatory Pond in Central Park New York City was taken on October 22, 2011 with a Canon EOS 30D camera.

Reviews

By Ricardo on September 15, 2015

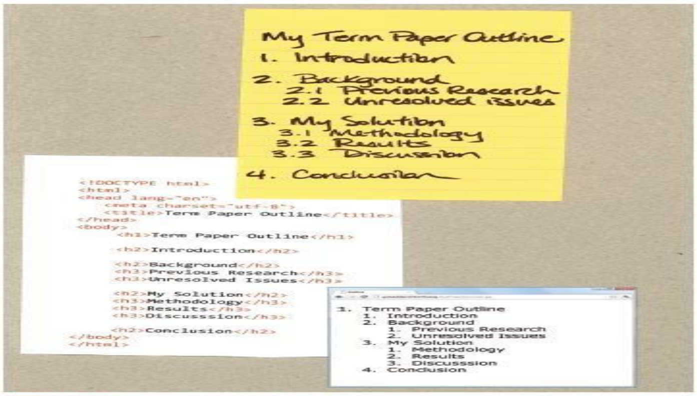
Easy on the HDR buddy.

By Susan on October 1, 2015

I love Central Park.

Copyright © 2015 Share Your Travels

FIGURE 2.11 Figure 2.10 in the browser

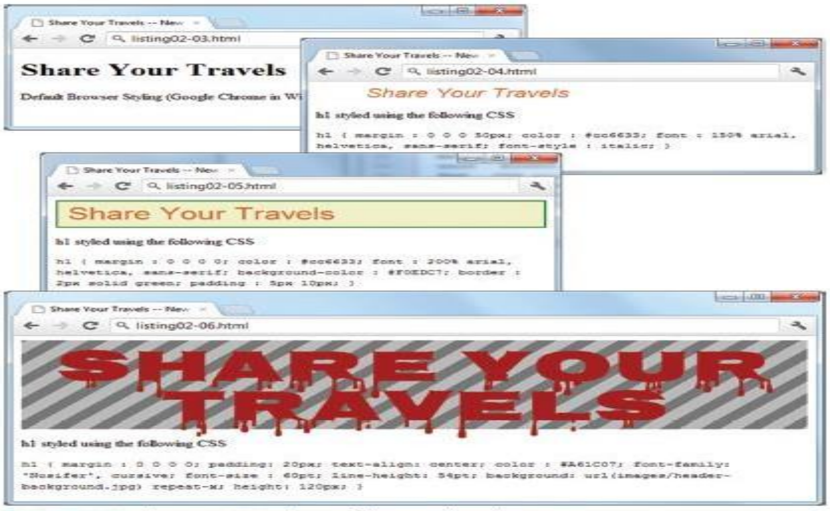

My Term Paper Outline
1. Introduction
2. Background
2.1 Previous Research
2.2 Unresolved issues
3. My Solution
3.1 Methodology
3.2 Results
3.3 Discussion
4. Conclusion

```
<!DOCTYPE html>
<html>
<head lang="en">
   <meta charset="utf-8">
   <title>Term Paper Outline</title>
</head>
<body>
     <h1>Term Paper Outline</h1>

    <h2>Introduction</h2>

    <h2>Background</h2>
    <h3>Previous Research</h3>
    <h3>Unresolved Issues</h3>

    <h2>My Solution</h2>
    <h3>Methodology</h3>
    <h3>Results</h3>
    <h3>Discusssion</h3>

    <h2>Conclusion</h2>
</body>
</html>
```

```
1. Term Paper Outline
   1. Introduction
   2. Background
      1. Previous Research
      2. Unresolved Issues
   3. My Solution
      1. Methodology
      2. Results
      3. Discusssion
   4. Conclusion
```

FIGURE 2.12 Example document outlines

- ➢ This document outline is constructed from headings and other structural tags in your content and is analogous to the outlines you may have created for your own term papers in school.
- ➢ Figure 2.12 illustrates one of these tools; this one is available from **http://gsnedders.html5.org/outliner/**. The browser has its own default styling for each heading level. However, these are easily modified and customized via CSS.
- ➢ Figure 2.13 illustrates just some of the possible ways to style a heading. 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



h1 styled using the following CSS

```
h1 { margin : 0 0 0 50px; color : #cc6633; font : 150% arial,
helvetica, sans-serif; font-style : italic; }
```

h1 styled using the following CSS

```
h1 { margin : 0 0 0 0; color : #cc6633; font : 200% arial,
helvetica, sans-serif; background-color : #F0EDC7; border :
2px solid green; padding : 5px 10px; }
```

h1 styled using the following CSS

```
h1 { margin : 0 0 0 0; padding: 20px; text-align: center; color : #A61C07; font-family:
"Nosifer", cursive; font-size : 60pt; line-height: 54pt; background: url(images/header-
background.jpg) repeat-x; height: 120px; }
```

FIGURE 2.13 Alternate CSS stylings of the same heading

2.5.2 Paragraphs and Divisions

- Figure 2.10 defines two paragraphs, 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** (the <strong> and <a> elements in Figure 2.10).
- This term refers to HTML elements that do not cause a paragraph break but are part of the regular "flow" of the text.
- The indenting on the second paragraph element is optional. Some developers like to use indenting to differentiate a container from its content.
- The former is a container for text and other inline elements. The line break element 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.
- Figure 2.10 illustrates the definition of a <div> element. This element is also a container element and is used to create a logical grouping of content.
- The <div> element has no intrinsic presentation; it is frequently used in contemporary CSS-based layouts to mark out sections.

2.5.3 Links

Figure 2.10 defines a hyperlink. Links are an essential feature of all web pages. Links are created using the <a> element (the “a” stands for anchor).

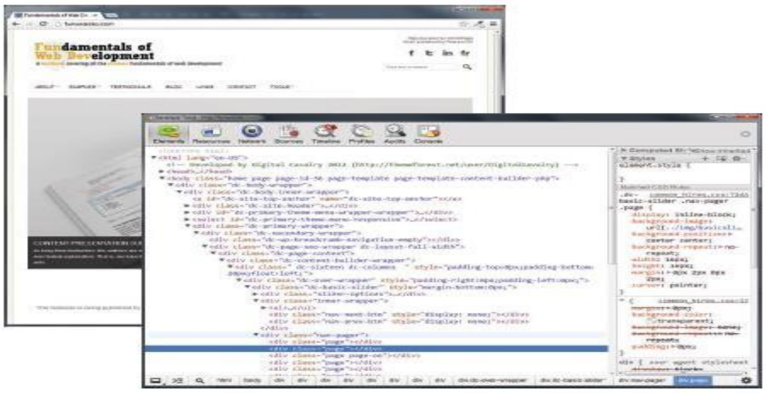
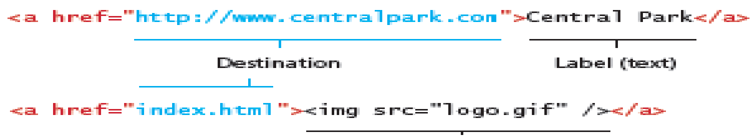


FIGURE 2.14 Using <div> elements to create a complex layout

A link label of a link can be text or another HTML element such as an image. We can use the anchor element to create a wide range of links. These include:

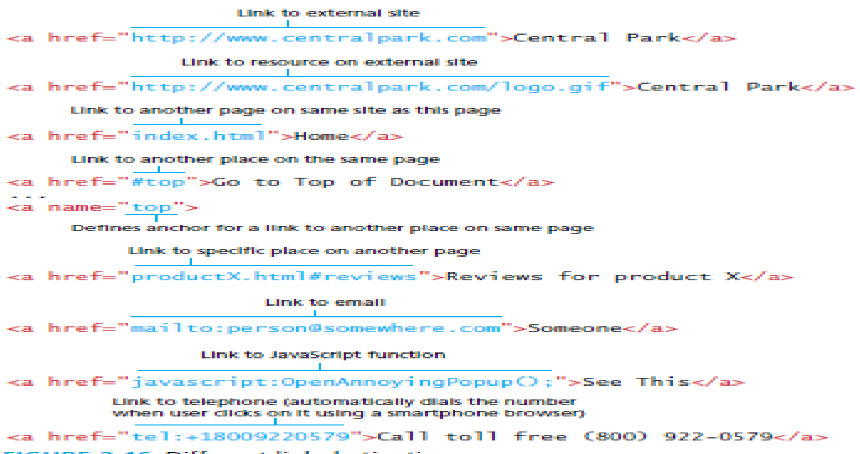
- ➢ Links to external sites (or to individual resources such as images or movies on an external site).
- ➢ Links to other pages or resources within the current site.
- ➢ Links to other places within the current page.
- ➢ Links to particular locations on another page (whether on the same site or on an 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 2.16 illustrates the different ways to construct link destinations



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

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

URL Relative Referencing

- Whether we are constructing links with the <a> element, referencing images with the <img> element, or including external JavaScript or CSS files, we need to be able to successfully reference files within our site. This requires learning the syntax for so called **relative referencing**.



```
                 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>
...
<a name="top">
        Defines anchor for a link to another place on same page
              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>
```

FIGURE 2.16 Different link destinations

- As you can see from Figure 2.16, when referencing a page or resource on an external site, a full **absolute reference** is required.

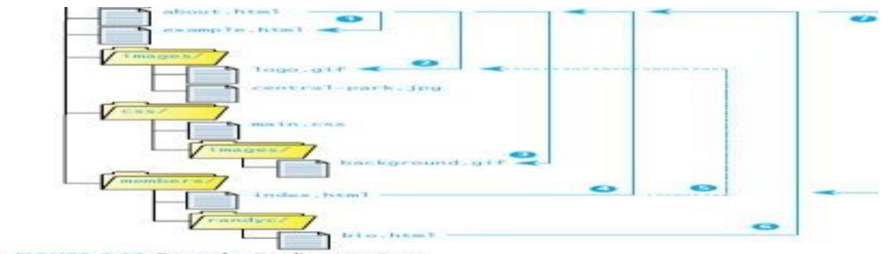
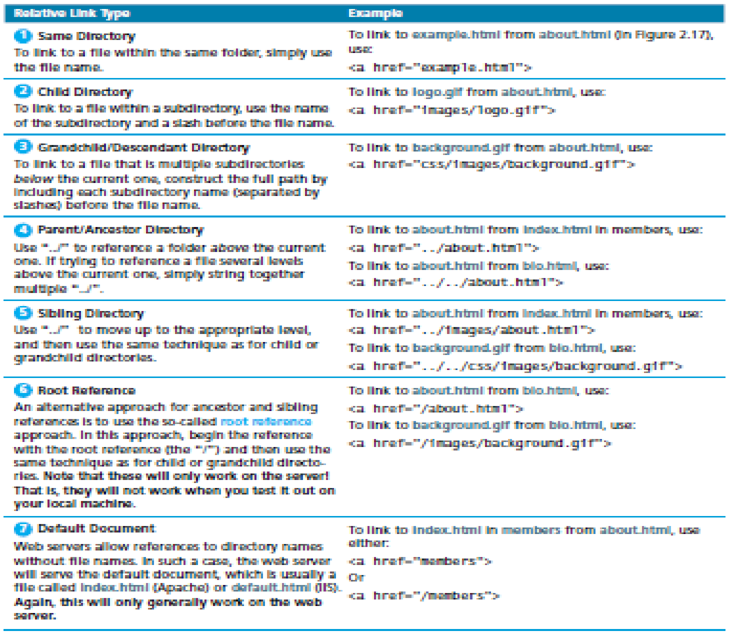
- 

FIGURE 2.17 Example site directory tree

- When referencing a resource that is on the same server as your HTML document, 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. 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 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 2.17 illustrates the file structure of an example site. Table 2.1 provides additional explanations and examples of the different types of URL 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 2.17), 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">`
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 about.html 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">`

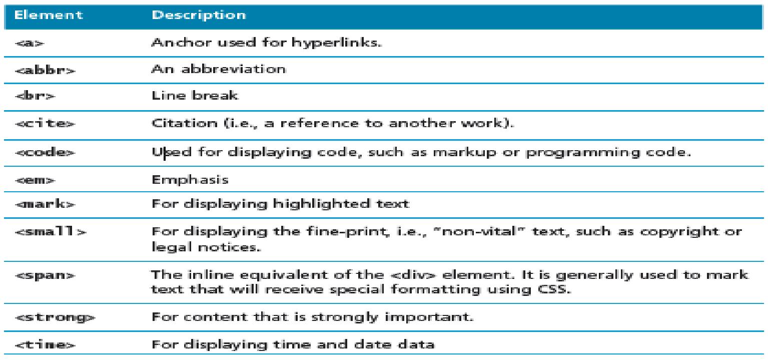
TABLE 2.1 Sample Relative Referencing

2.5.5 Inline Text Elements

- They are called inline elements because they do not disrupt the flow of text (i.e., cause a line break). HTML defines over 30 of these elements. Table 2.2 lists some of the most commonly used of these elements

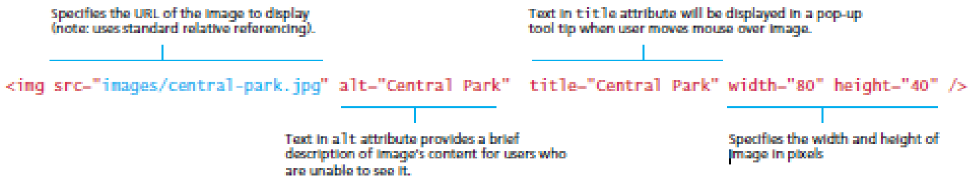
2.5.6 Images

- Figure 2.10 defines an image. While the <img> tag is the oldest method to for displaying an image, it is not the only way. In fact, it is very common for images be added to HTML elements via the background-image property in CSS, a technique.
- For purely decorative images, such as background gradientsand 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.



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.
<em>	Emphasis
<mark>	For displaying highlighted text
<small>	For displaying the fine-print, i.e., "non-vital" text, such as copyright or legal notices.
<span>	The inline equivalent of the <div> element. It is generally used to mark text that will receive special formatting using CSS.
<strong>	For content that is strongly important.
<time>	For displaying time and date data

TABLE 2.2 Common Text-Level Semantic Elements



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

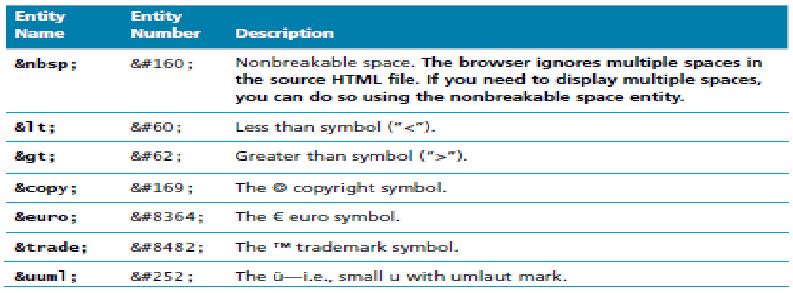
Specifies the URL of the image to display (note: uses standard relative referencing).
Text in title attribute will be displayed in a pop-up tool tip when user moves mouse over image.
Text in alt attribute provides a brief description of image's content for users who are unable to see it.
Specifies the width and height of image in pixels

FIGURE 2.18 The <img> element

- 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.

2.5.7 Character Entities

- Figure 2.10 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 keyboard or which have a reserved meaning in HTML.
- There are many HTML character entities. They can be used in an HTML document by using theentity name or the entity number. Some of the most common are listed in Table 2.3

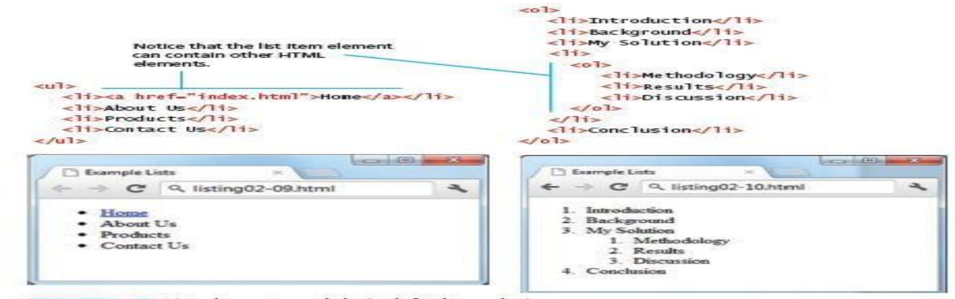


Entity Name	Entity Number	Description
`&nbsp;`	`&#160;`	Nonbreakable space. The browser ignores multiple spaces in the source HTML file. If you need to display multiple spaces, you can do so using the nonbreakable space entity.
`&lt;`	`&#60;`	Less than symbol ("<").
`&gt;`	`&#62;`	Greater than symbol (">").
`&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 2.3 Common Character Entities

2.5.8 Lists

- Figure 2.10 is missing one of the most common block-level elements in HTML,namely, 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. However, it is common in CSS to style unordered lists without the bullets. Unordered lists have become the conventional way to markup navigational menus.
- **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. As can be seen in Figure 2.19, 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 2.19.


Notice that the list item element can contain other HTML elements.
Example Lists
listing02-09.html
Example Lists
listing02-10.html

```
<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>
```

```
• Home
• About Us
• Products
• Contact Us
```

```
1. Introduction
2. Background
3. My Solution
      1. Methodology
      2. Results
      3. Discussion
4. Conclusion
```

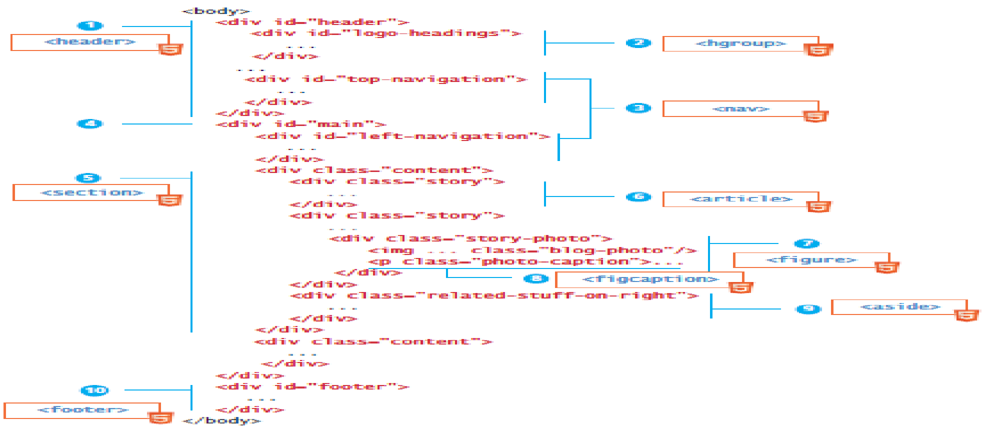
FIGURE 2.19 List elements and their default rendering

2.6 HTML5 Semantic Structure Elements

- The other key semantic block element, namely the division(i.e., <div> element).Figure 2.14 did, however, illustrate one substantial problem with modern, preHTML5 semantic markup.
- Most complex websites are absolutely packed solid with<div> elements. Most of these are marked with different id or class attributes.
- Developers typically try to bring some sense and order to the <div> chaos by using id or class names that provide some clue as to their meaning, as shown in Figure 2.20.
- Their findings helped standardize the names of the new semantic block structuring elements in HTML5, most of which are shown in Figure 2.20.The idea behind using these elements is that your markup will be easier to understand because you will be able to replace some of your <div> sprawl with cleaner and more self-explanatory HTML5 elements.
- Figure 2.21 illustrates the simpler version of Figure 2.20, one that uses the new semantic elements in HTML5.Each of these elements is briefly discussed below.

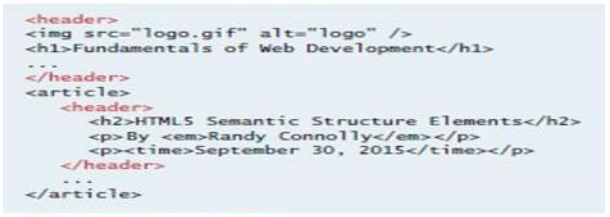
2.6.1 Header and Footer

- 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 twitter feeds or links to other social sites.
- Both the HTML5 <header> and <footer> element can be used not only for*page* headers and footers (as shown in items 1and 9 in figure 2.21), but also forheader and footer elements within other HTML5 containers, such as <article> or<section>, as indicated by the W3C draft of the HTML5 standard:



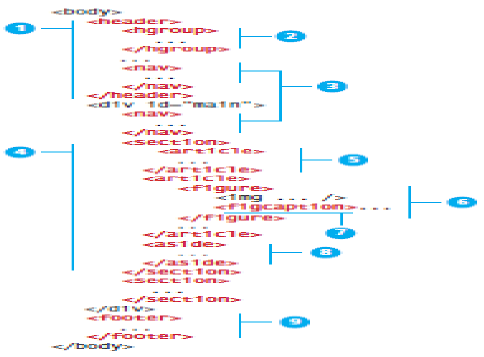
```
<body>
   <div id="header">
      <div id="logo-headings">
         ...
      </div>
      ...
      <div id="top-navigation">
         ...
      </div>
   </div>
   <div id="main">
      <div id="left-navigation">
         ...
      </div>
      <div class="content">
         <div class="story">
            ...
         </div>
         <div class="story">
            ...
            <div class="story-photo">
               <img ... class="blog-photo"/>
               <p class="photo-caption">...
            </div>
         </div>
         <div class="related-stuff-on-right">
            ...
         </div>
      </div>
      <div class="content">
         ...
      </div>
   </div>
   <div id="footer">
      ...
   </div>
</body>
```

FIGURE 2.20 Sample <div>-based XHTML layout (with HTML5 equivalents)



```
<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>
```

.ISTING 2.1 Heading example



```
<body>
   <header>
      <hgroup>
         ...
      </hgroup>
      ...
      <nav>
         ...
      </nav>
   </header>
   <div id="main">
      <nav>
         ...
      </nav>
      <section>
         <article>
            ...
         </article>
         <article>
            <figure>
               <img ... />
               <figcaption>...
            </figure>
            ...
         </article>
         <aside>
            ...
         </aside>
      </section>
      <section>
         ...
      </section>
   </div>
   <footer>
      ...
   </footer>
</body>
```

FIGURE 2.21 Sample layout using new HTML5 semantic structure elements

➢ The browser really doesn’t care how one uses these HTML5 semantic structureelements. Just like with the <div> element, there is no predefined presentation forthese tags.

2.6.2 Heading Groups

- The <hgroup> element (seen as item 2 in Figure2.21) can be used in such a circumstance to group them 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.
- The <nav> element represents a section of a page that contains links to other pages or to other parts within the same page.
- In April 2013, the W3C decided to drop the <hgroup> element from theW3C specification. While browsers will likely continue to support the <hgroup>element, if you need to group headings, you are encouraged to instead nest them within a <div> element. element was intended to be used for major navigation blocks, presumably the global and secondary navigation systems as well as perhaps search facilities.
- Its sole purpose is to make your markup easier to understand, and by limiting the use of the <nav> element tomajor elements, your markup will more likely achieve that aim.

```
<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 2.3 nav example

Articles and Sections

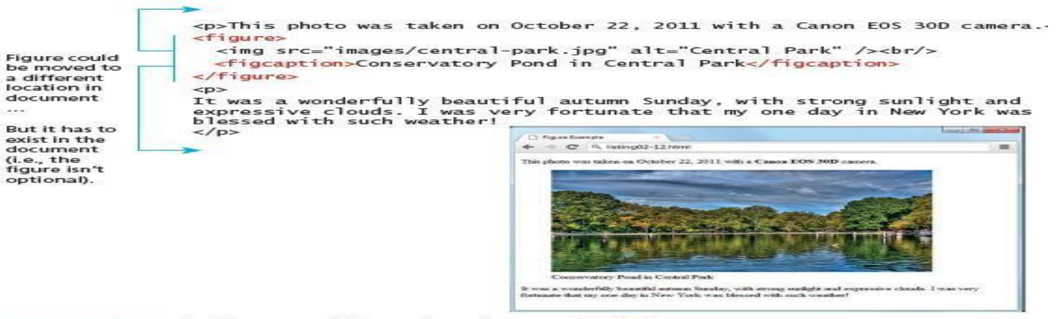
- The new HTML5 semantic elements <section> and <article>) play a similar role within web pages.
- According to the W3C, <section> is a much broader element, while the<article> element is to be used for blocks of content that could potentially be read or consumed independently of the other content on the page. We can gain a further understanding of

how to use these two elements by looking at the more expansive WHATWG specification.

- The reference to syndication in the WHATWG explanation of the <article>element is useful.
- In the context of the web, **syndication** refers to websites making their content available to other websites for display. If some block of content could theoretically exist on another website (as if it were syndicated) and still make sense in that new context, then wrap that content within an <article> element.

Figure and Figure Captions

- Throughout this chapter you have seen screen captures or diagrams or photographsthat are separate from the text (but related to it), which are described by a caption,and which are given the generic name of *Figure*. Prior to HTML5.
- In HTML5 we can instead use the more obvious <figure> and <figcaption>elements (items 6 and 7 in Figure 2.21).
- The W3C Recommendation indicates that the <figure> element can be usednot just for images but for any type of *essential* content that could be moved to adifferent location in the page or document and the rest of the document would still make sense.



```
<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 my one day in New York was
blessed with such weather!
</p>
```

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).

This photo was taken on October 22, 2011 with a Canon EOS 30D camera.

Conservatory Pond in Central Park

It was a wonderfully beautiful autumn Sunday, with strong sunlight and expressive clouds. I was very fortunate that my one day in New York was blessed with such weather!

FIGURE 2.22 The figure and figcaption elements in the browser

Aside

- The <aside> element is similar to the <figure>.

- The <aside> element "represents a section of a page that consists of content that is tangentially related to the content around the aside element" (from WHATWG specification).
- The <aside> element could thus be used for sidebars, pull quotes, groups of advertising images, or any other grouping of non-essential elements.

3.1 What Is CSS?

- CSS is a W3C standard for describing the appearance of HTML elements. Another common way 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.

3.1.1 Benefits of CSS

The benefits ofCSS include:

- **Improved control over formatting.** The degree of formatting control in CSS is significantly better than that provided in HTML. CSS gives web authors 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, or a small handful of them. This allows you 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. This approach to CSS page design is often referred to as **responsive design**. Figure 3.1 illustrates a site that responds to different browser and window sizes.

3.1.2 CSS Versions

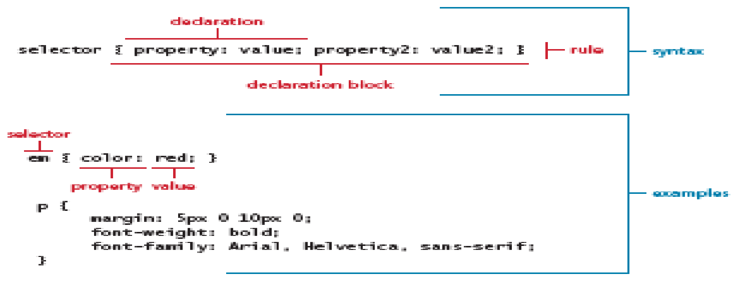
- Netscape's proposal was one that required the use of JavaScript programming to perform style changes.
- The W3C decided to adopt CSS, and by the end of 1996 the CSS Level 1 Recommendation was published. Ayear later, the CSS Level 2 Recommendation (also more succinctly labeled simply asCSS2) was published.
- To makeCSS3 more manageable for both browser manufacturers and web designers, theW3C has subdivided it into a variety of different **CSS3 modules**.

3.1.3 Browser Adoption

- While 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.
- For this reason, CSS has a reputation for being a somewhat frustrating language. Based on over a decade of experience teaching university students CSS, this reputation is well deserved. Since CSS was designed to be a styling language, text styling is quite easy.
- When one adds in the uneven CSS 2.1 support (prior to IE8 and Firefox 2) in browsers for CSS2.1, it becomes quite clear why many software developers developed a certain fear and loathing of CSS.

3.2 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 3.2.



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

em { color: red; }

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

FIGURE 3.2 CSS syntax

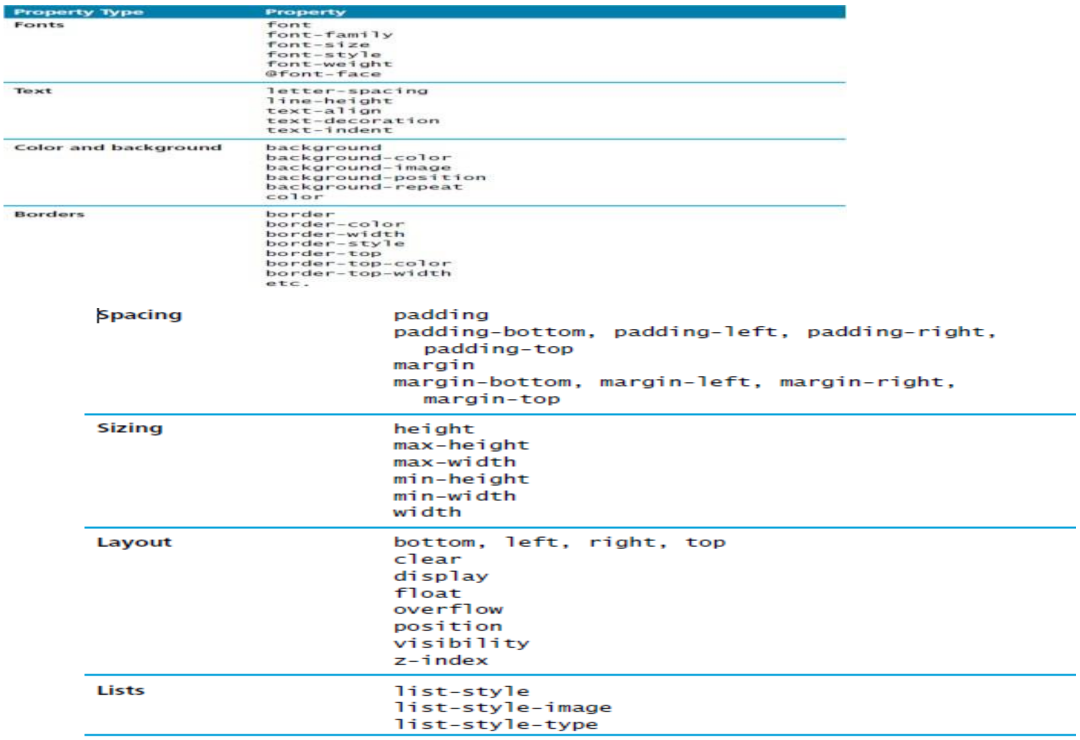
- The series of declarations is also called the **declaration block**. As one can see in the illustration, a declaration block can be together on a single line, or spread across multiple lines.
- The browser ignores white space (i.e., spaces, tabs, and returns)between your CSS rules so you can format the CSS however you want. Notice that 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; that way, if you add rules to the end later, you will reduce the chance of introducing a rather subtle and hard-to-discover bug.

3.2.1 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.
- The CSS2.1 recommendation defines over a hundred different property names, so some type of reference guide, whether in a book, online, or within your web development software, can be helpful.
- Table 3.1 lists many of the most commonly used CSS properties. the browser to select the HTML elements that will receive the style.



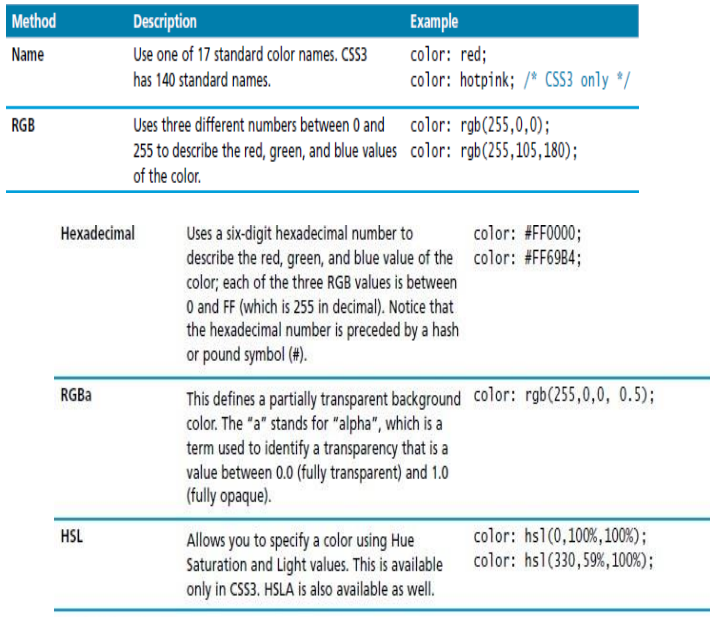
Property Type	Property
Fonts	font font-family font-size font-style font-weight @font-face
Text	letter-spacing line-height text-align text-decoration text-indent
Color and background	background background-color background-image background-position background-repeat color
Borders	border border-color border-width border-style border-top border-top-color border-top-width etc.
Spacing	padding padding-bottom, padding-left, padding-right, padding-top margin margin-bottom, margin-left, margin-right, margin-top
Sizing	height max-height max-width min-height min-width width
Layout	bottom, left, right, top clear display float overflow position visibility z-index
Lists	list-style list-style-image list-style-type

TABLE 3.1 Common CSS Properties

3.2.3 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.

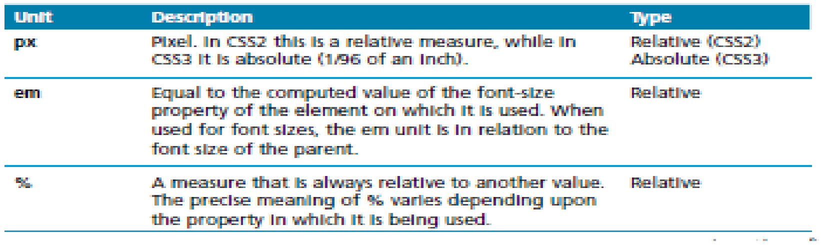
- Others are values such as length measurements, percentages, numbers without units, color values, and URLs.Colors would seem at first glance to be the most clear of these units.



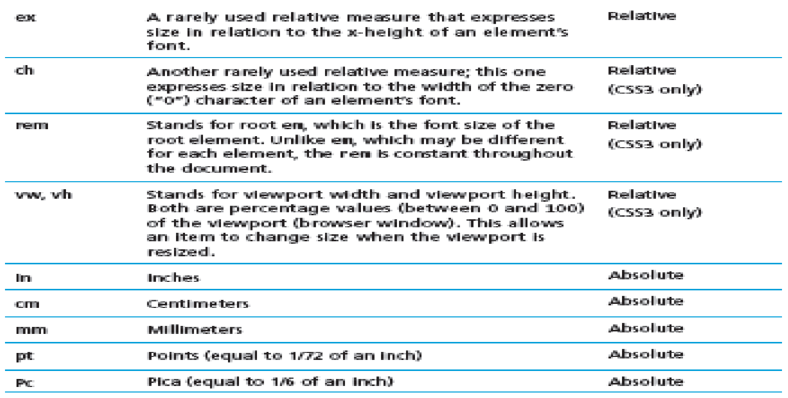
Method	Description	Example
Name	Use one of 17 standard color names. CSS3 has 140 standard names.	`color: red;` `color: hotpink; /* CSS3 only */`
RGB	Uses three different numbers between 0 and 255 to describe the red, green, and blue values of the color.	`color: rgb(255,0,0);` `color: rgb(255,105,180);`
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	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%);` `color: hsl(330,59%,100%);`

TABLE 3.2 Color Values

- 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



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
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

TABLE 3.3 Units of Measure Values

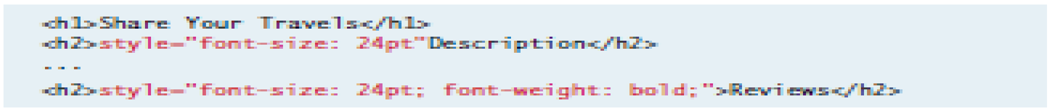
- ➢ Pixels are perhaps the one popular exception (though, as we shall see later, there are also good reasons for avoiding the pixel unit). In general, most of the CSS that you will see uses either px, em, or % as a measure unit

3.3 Location of Styles

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

3.3.1 Inline Styles

- ➢ **Inline styles** are style rules placed within an HTML element via the style attribute, as shown in Listing 3.1.
- ➢ An inline style only affects the element it is defined within and overrides any other style definitions for properties used in the inline style.
- ➢ Selector is not necessary with inline styles and that semicolons are only required for separating multiple rules.

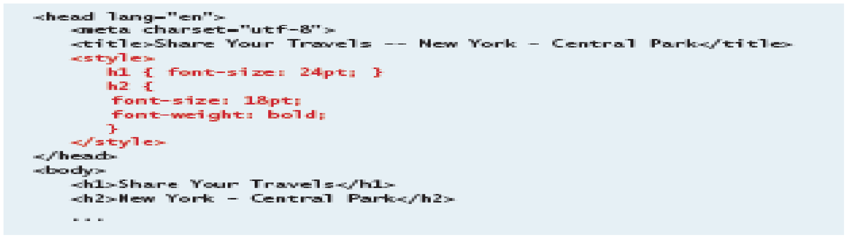


```
<h1>Share Your Travels</h1>
<h2>style="font-size: 24pt"Description</h2>
...
<h2>style="font-size: 24pt; font-weight: bold;">Reviews</h2>
```

LISTING 3.1 Internal styles example

3.3.2 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 3.2.



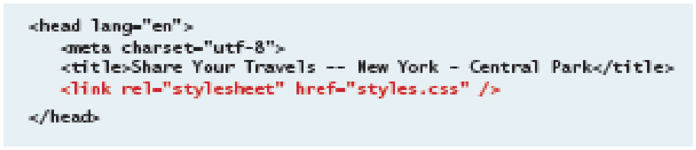
```
<head lang="en">
    <meta charset="utf-8">
    <title>Share Your Travels -- New York - Central Park</title>
    <style>
        h1 { font-size: 24pt; }
        h2 {
         font-size: 18pt;
         font-weight: bold;
        }
    </style>
</head>
<body>
    <h1>Share Your Travels</h1>
    <h2>New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

LISTING 3.2 Embedded styles example

- ➢ 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.

3.3.3 External Style Sheet

- ➢ **External style sheets** are style rules placed within a external text file with the **.css** extension.
- ➢ This is by far the most common place to locate style rules because it provides the best maintainability.
- ➢ When we 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.
- ➢ To reference an external style sheet, you must use a <link> element (within the<head> element), as shown in Listing 3.3.



```
<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

3.4 Selectors

- ➢ When defining CSS rules, you will need to first use a selector to tell the browser which elements will be affected by the property values.

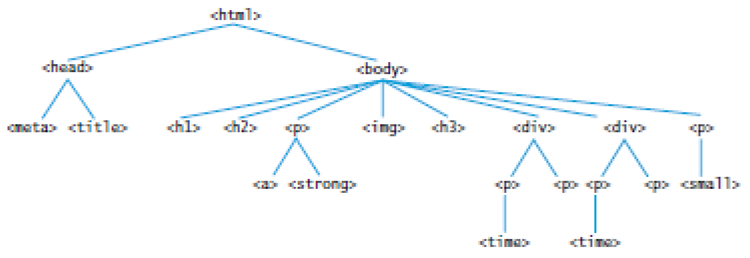
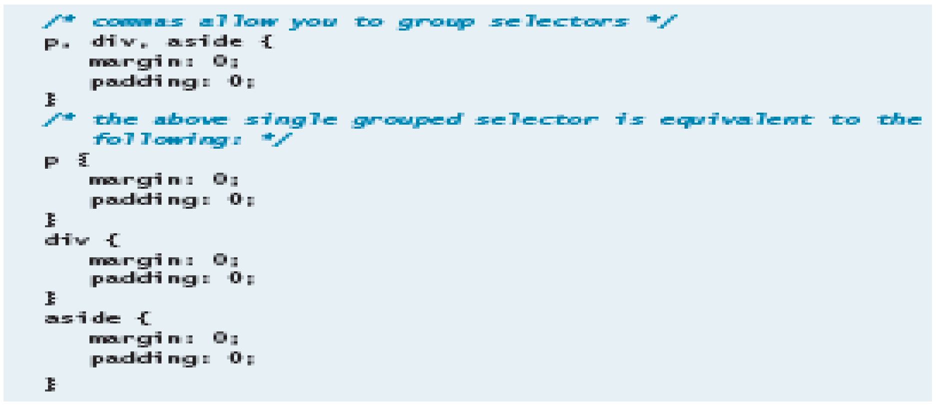


FIGURE 3.3 Document outline/tree

- CSS selectors allow you to select individual or multiple HTML elements.
- There are now a variety of new selectors that are supported by most modern browsers. Before we get to those, let us look at the three basic selector types that have been around since the earliest CSS2specification.

3.4.1 Element Selectors

- **Element selectors** select all instances of a given HTML element.
- 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.
- This is a sensible way to reduce the size and complexity of our CSS files, by combining multiple identical rules into a single rule. An example-grouped **selector** is shown in Listing 3.4, along with its equivalent as three separate rules.

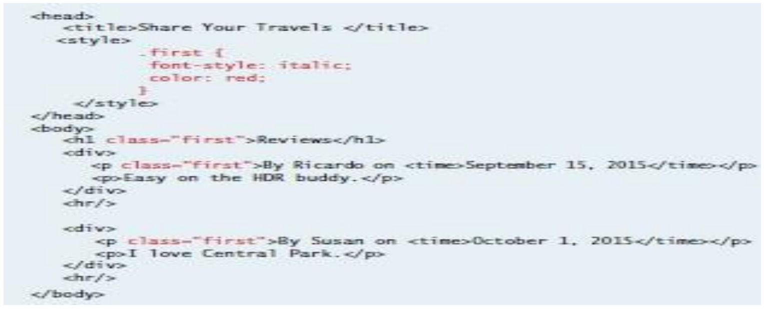


```
/* commas allow you to group selectors */
p, div, aside {
    margin: 0;
    padding: 0;
}
/* the above single grouped selector is equivalent to the
   following: */
p {
    margin: 0;
    padding: 0;
}
div {
    margin: 0;
    padding: 0;
}
aside {
    margin: 0;
    padding: 0;
}
```

LISTING 3.4 Sample grouped selector

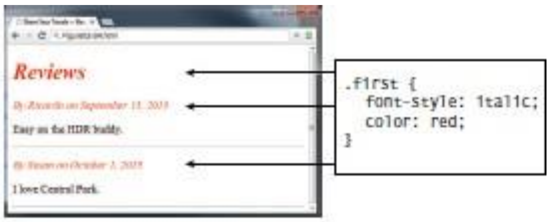
3.4.2 Class Selectors

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



```
<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 3.5 Class selector example



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

FIGURE 3.4 Class selector example in browser

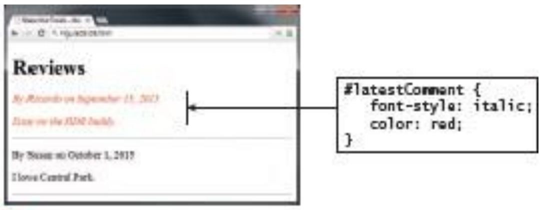
3.4.3 Id Selectors

- An **id selector** allows us to target a specific element by its id attribute regardlessof 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.

```
<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 3.6 Id selector example

➢ Listing 3.6 illustrates an example of styling using an id selector. The result in the browser is shown in Figure 3.5.



Reviews

By Ricardo on September 15, 2015

Easy on the HDR buddy.

By Susan on October 1, 2015

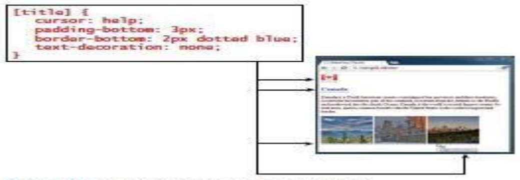
I love Central Park.

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

FIGURE 3.5 Id selector example in browser

```
<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 3.7 Attribute selector example

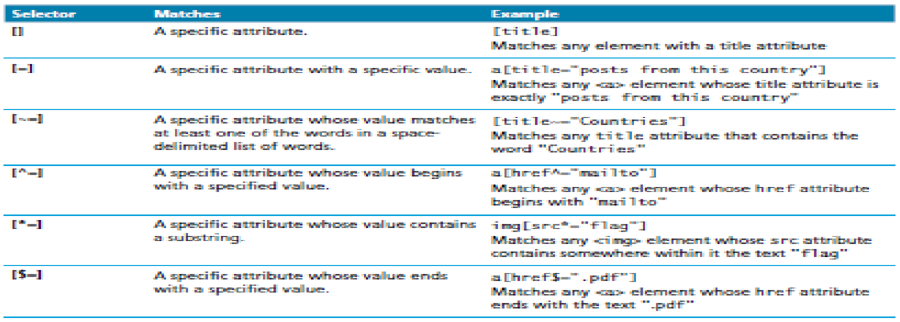


```
[title] {
   cursor: help;
   padding-bottom: 3px;
   border-bottom: 2px dotted blue;
   text-decoration: none;
}
```

FIGURE 3.6 Attribute selector example in browser

3.4.4 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.



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

TABLE 3.4 Attribute Selectors

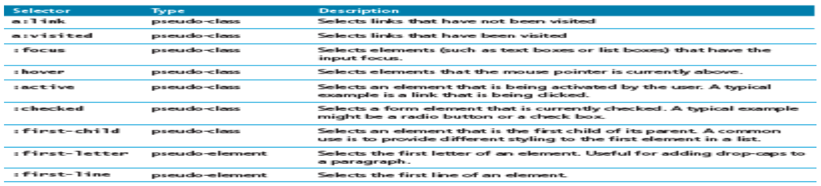
- Attribute selectors can be a very helpful technique in the styling of hyperlinks and images. For instance, we want to make it more obvious to the user when a pop-up tooltip is available for a link or image.
- We can do this by using thefollowing attribute selector:

  [title] { … }

- This will match any element in the document that has a title attribute.

3.4.5 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, you can select the first line or first letter of any HTML element using a pseudo-element selector.

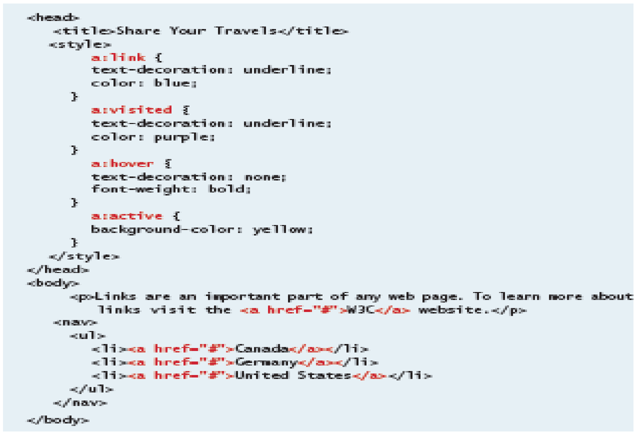


Selector	Type	Description
a:link	pseudo-class	Selects links that have not been visited
a:visited	pseudo-class	Selects links that have been visited
:focus	pseudo-class	Selects elements (such as text boxes or list boxes) that have the input focus.
:hover	pseudo-class	Selects elements that the mouse pointer 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.

TABLE 3.5 Common Pseudo-Class and Pseudo-Element Selectors

- A **pseudo-class selector** does apply to an HTML element, but targets either

- Listing 3.8 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.
- 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. Believe it or not, the order of these pseudo-class elements is important.
- The :link and :visited pseudo-classes should appear before the others. Some developers use a mnemonic to help them remember the order.



```
<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

3.4.6 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.

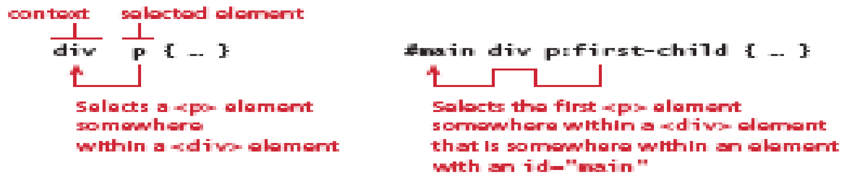

context
selected element
div p { … }
Selects a <p> element somewhere within a <div> element
#main div p:first-child { … }
Selects the first <p> element somewhere within a <div> element that is somewhere within an element with an id="main"

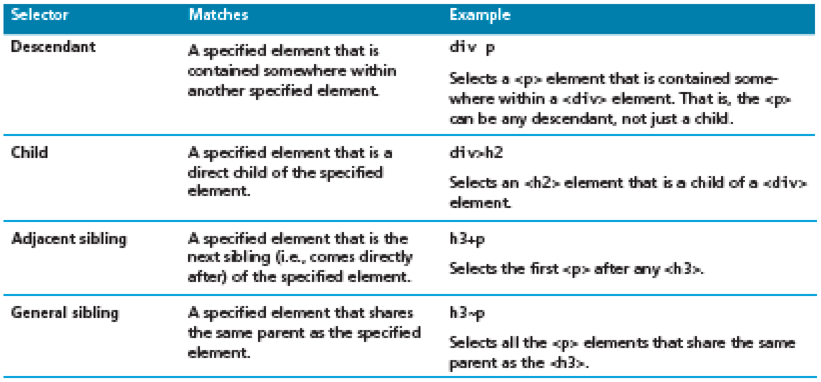
FIGURE 3.7 Syntax of a descendant selection

- 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 3.7 illustrates the syntax and usage of the syntax of the descendant selector.

3.5 The Cascade: How Styles Interact

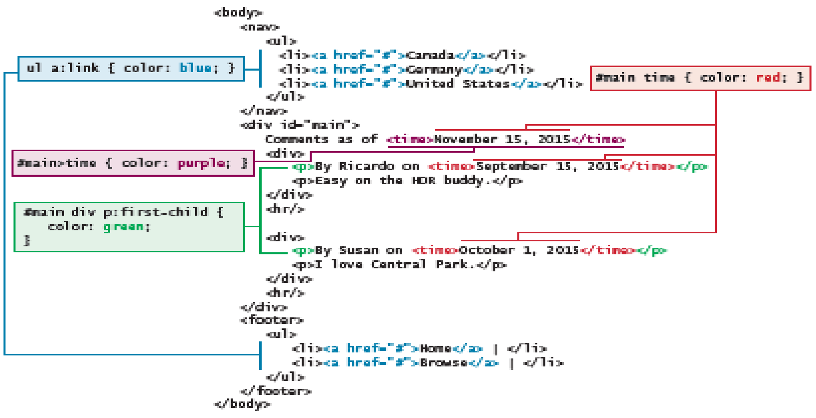
- There are three different types of style sheets: author-created, user-defined, and the default browser style sheet.
- It is possible within an author-created style sheet to define multiple rules for the same HTML element.
- CSS has a system to help the browser determine how to display elements when different style rules conflict. The "Cascade" in CSS refers to how conflicting rules are handled.



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

TABLE 3.6 Contextual Selectors

- The term **cascade** is that of a mountain stream progressing downstream over rocks. The downward movement of water down a cascade is meant to be analogous to how a given style rule will continue to take precedence with child elements.
- CSS uses the following cascade principles to help it deal with conflicts: inheritance, specificity, and location.

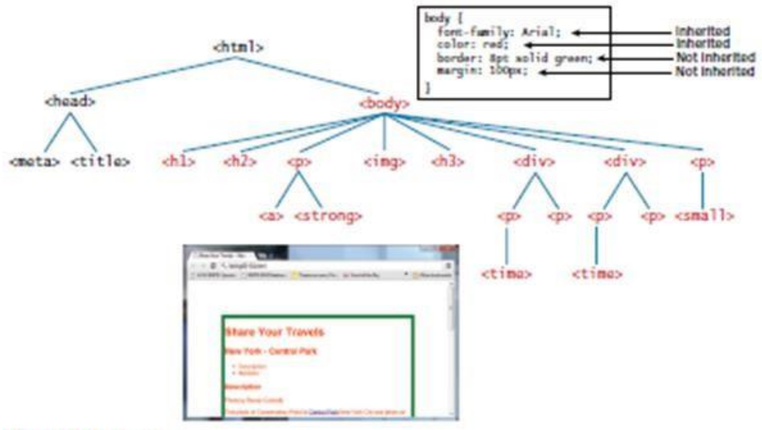


```
<body>
   <nav>
      <ul>
        <li><a href="#">Canada</a></li>
        <li><a href="#">Germany</a></li>
        <li><a href="#">United States</a></li>
      </ul>
   </nav>
   <div id="main">
      Comments as of <time>November 15, 2015</time>
      <div>
         <p>By Ricardo on <time>September 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 3.8 Contextual selectors in action

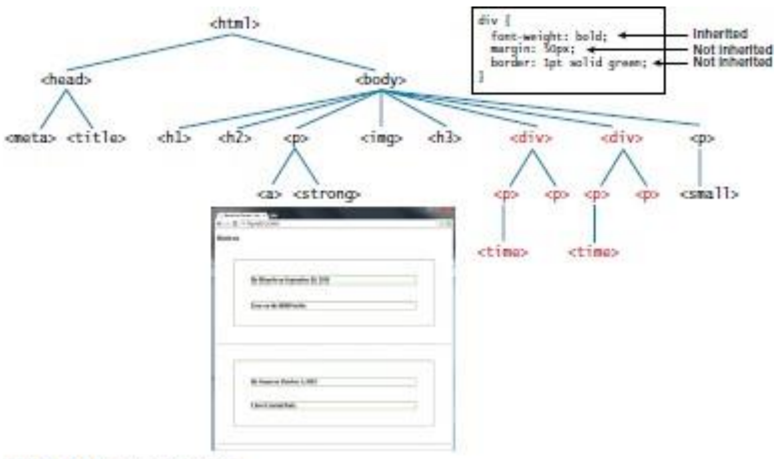
3.5.1 Inheritance



```
body {
   font-family: Arial;
   color: red;
   border: 8pt solid green;
   margin: 100px;
}
```

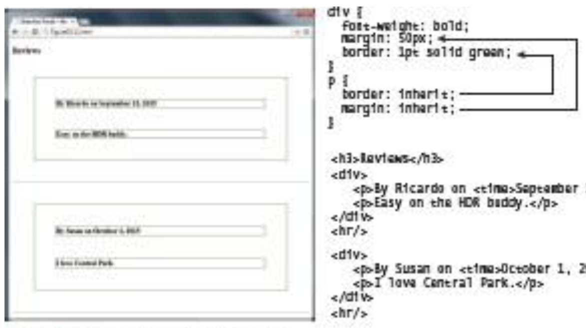
FIGURE 3.9 Inheritance

- **Inheritance** is the first of these cascading principles. Many (but not all) CSS properties affect not only themselves but their descendants as well. Font, color, list, and text properties (from Table 3.1) are inheritable; layout, sizing, border, background, and spacing properties are not.
- It is possible to tell elements to inherit properties that are normallynot inheritable, as shown in Figure 3.11. In comparison to Figure 3.10, notice howthe <p> elements nested within the <div> elements now inherit the border andmargins of their parent.



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

FIGURE 3.10 More Inheritance

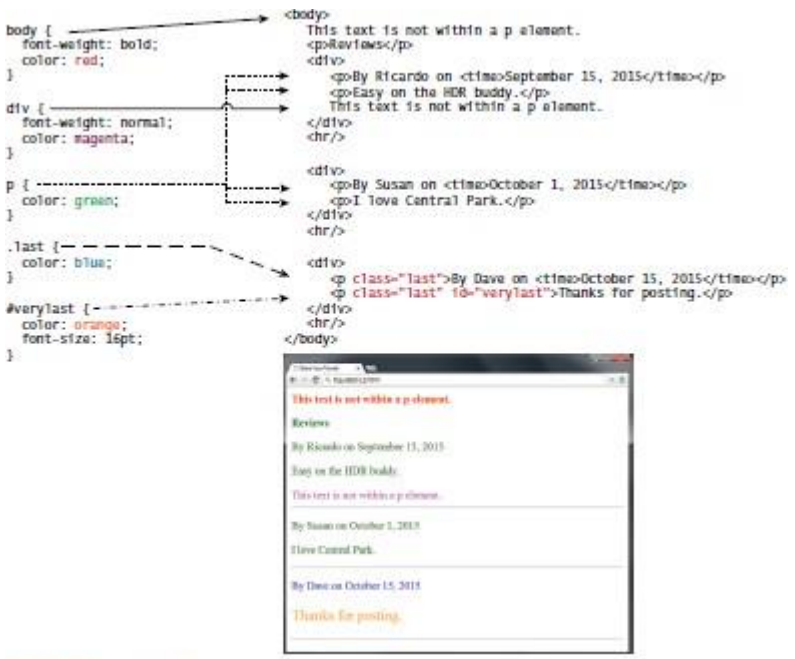


```
div {
  font-weight: bold;
  margin: 50px;
  border: 1pt solid green;
}
p {
  border: inherit;
  margin: inherit;
}

<h3>Reviews</h3>
<div>
   <p>By Ricardo on <time>September 15, 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/>
```

FIGURE 3.11 Using the inherit value

3.5.2 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 .Another way to define specificity is by telling you 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.
- The<div> and <p> elements also have the same properties set, they *override* the value defined for the<body> element because their selectors (<div> and <p>) are more specific.



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

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

p {
  color: green;
}

.last {
  color: blue;
}

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

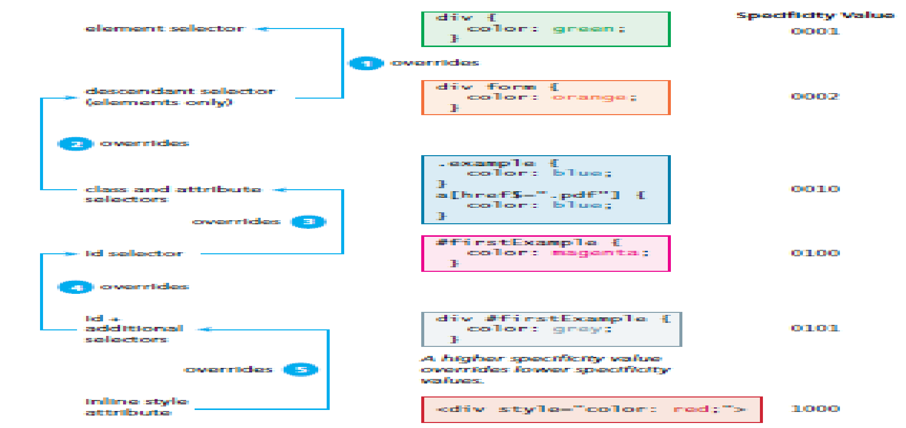
```html
<body>
   This text is not within a p element.
   <p>Reviews</p>
   <div>
      <p>By Ricardo on <time>September 15, 2015</time></p>
      <p>Easy on the HDR buddy.</p>
      This text is not within a p element.
   </div>
   <hr/>

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

   <div>
      <p class="last">By Dave on <time>October 15, 2015</time></p>
      <p class="last" id="verylast">Thanks for posting.</p>
   </div>
   <hr/>
</body>
```

```
This text is not within a p element.

Reviews

By Ricardo on September 15, 2015

Easy on the HDR buddy.

This text is not within a p element.

By Susan on October 1, 2015

I love Central Park.

By Dave on October 15, 2015

Thanks for posting.
```

FIGURE 3.12 Specificity

3.5.3 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.


element selector
1 overrides
descendant selector (elements only)
2 overrides
class and attribute selectors
overrides 3
id selector
4 overrides
id + additional selectors
overrides 5
inline style attribute
Specificity Value
0001
0002
0010
0100
0101
1000
A higher specificity value overrides lower specificity values.

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

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

```
.example {
   color: blue;
}
a[href$=".pdf"] {
   color: blue;
}
```

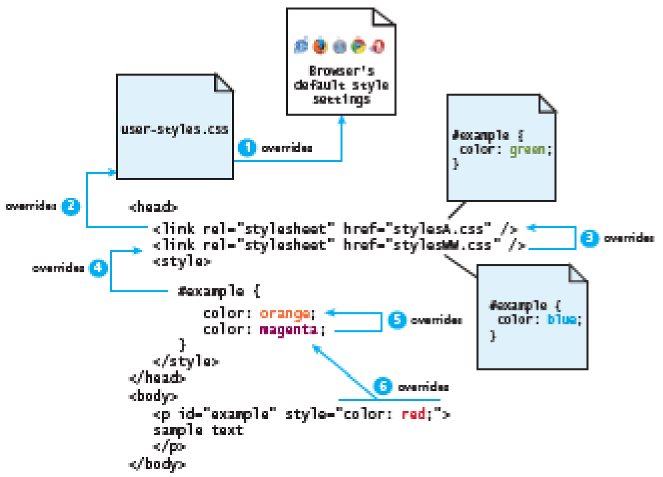
```
#firstExample {
   color: magenta;
 }
```

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

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

FIGURE 3.13 Specificity algorithm

- 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.
- Styles defined in external author style sheet X will override styles in external author style sheet Y if X's <link> element is after Y's in the HTML document.



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

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

```
<head>
   <link rel="stylesheet" href="stylesA.css" />
   <link rel="stylesheet" href="stylesWW.css" />
   <style>
      #example {
         color: orange;
         color: magenta;
      }
   </style>
</head>
<body>
   <p id="example" style="color: red;">
   sample text
   </p>
</body>
```

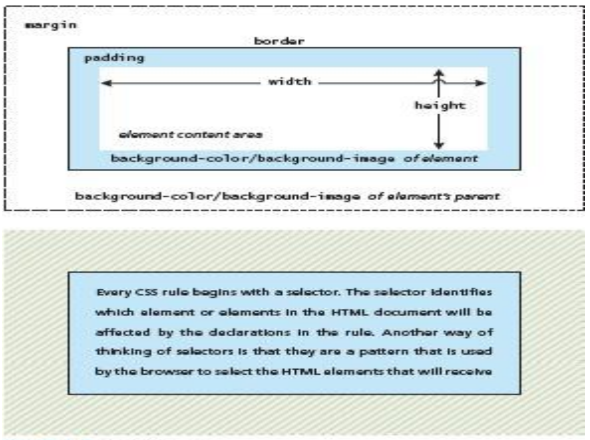
FIGURE 3.14 Location

3.6 The Box Model

In CSS, all HTML elements exist within an **element box** shown in Figure 3.15. Inorder to become proficient with CSS, you must become familiar with the element box.

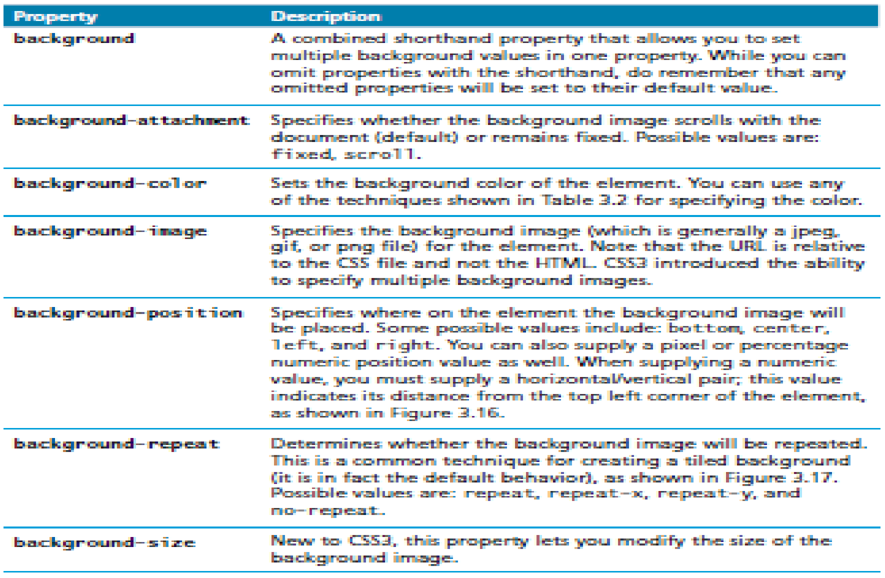
3.6.1 Background

- In Figure 3.15, 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 rather than using the <img> element.


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

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

FIGURE 3.15 CSS box model

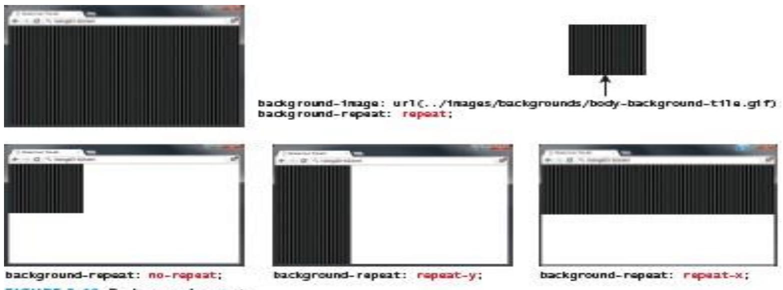


Property	Description
background	A combined shorthand property that allows you to set multiple background values in one property. While you can omit properties with the shorthand, do remember that any omitted properties will be set to their default value.
background-attachment	Specifies whether the background 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 3.7 Common Background Properties

3.6.2 Borders

- Borders provide a way to visually separate elements. You can put borders around all four sides of an element, or just one, two, or three of the sides.
- Border widths are perhaps the one exception to the general advice against using the pixel measure. Using em units or percentages for border widths can result in unpredictable widths as the different browsers use different algorithms (some roundup, some round down) as the zoom level increases or decreases. For this reason, border widths are almost always set to pixel units.



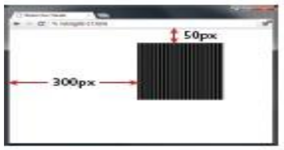
```
background-image: url(../images/backgrounds/body-background-tile.gif)
background-repeat: repeat;
```

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

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

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

FIGURE 3.16 Background repeat

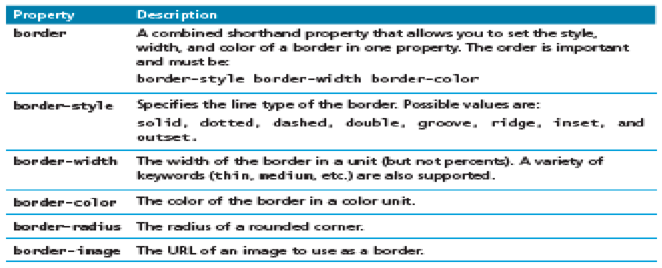


```
body {
        background: white url(../images/backgrounds/body-background-tile.gif) no-repeat;
        background-position: 300px 50px;
}
```

FIGURE 3.17 Background position

3.6.3 Margins and Padding

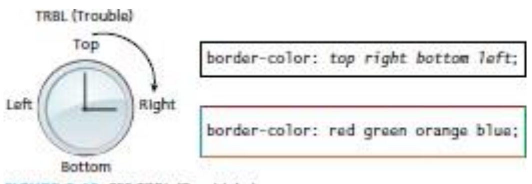
- Margins and padding are essential properties for adding white space to a web page, which can help differentiate one element from another.



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: `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.

TABLE 3.8 Border Properties

- Margins add spacing around an element's content, while padding adds spacing within elements.

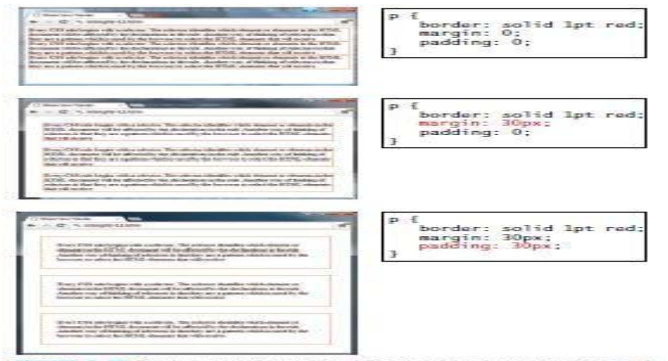


```
border-color: top right bottom left;
```

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

FIGURE 3.18 CSS TRBL (Trouble) shortcut

- Borders divide the margin area from the padding area.
- What this means is that when the **vertical** margins of two elements touch, onlythe 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.



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

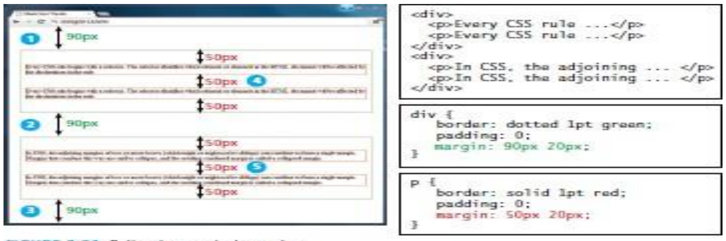
```
p {
    border: solid 1pt red;
    margin: 30px;
    padding: 0;
}
```

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

FIGURE 3.19 Borders, margins, and padding provide element spacing and differentiation

3.6.4 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.



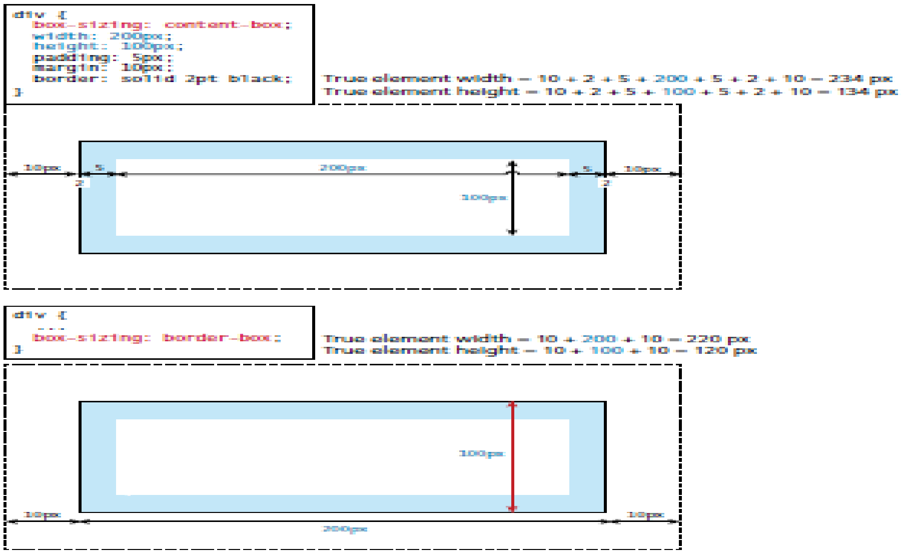
```
<div>
  <p>Every CSS rule ...</p>
  <p>Every CSS rule ...</p>
</div>
<div>
  <p>In CSS, the adjoining ... </p>
  <p>In CSS, the adjoining ... </p>
</div>
```

```
div {
   border: dotted 1pt green;
   padding: 0;
   margin: 90px 20px;
}
```

```
p {
   border: solid 1pt red;
   padding: 0;
   margin: 50px 20px;
}
```

FIGURE 3.20 Collapsing vertical margins

- ➢ 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.



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

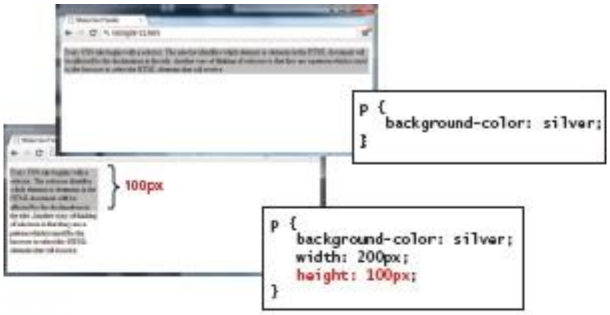
True element width = 10 + 2 + 5 + 200 + 5 + 2 + 10 = 234 px
True element height = 10 + 2 + 5 + 100 + 5 + 2 + 10 = 134 px

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

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

FIGURE 3.21 Calculating an element's true size

- ➢ It also shows the newer alternative border-box approach, which is more intuitive, but which requires vendor prefixes for it to work on all recent browsers.
- ➢ For block-level elements such as <p> and <div> elements, there are limits to what the width and height properties can actually do.
- ➢ It is possible to control what happens with the content if the box's width and height are not large enough to display the content using the overflow property,.
- ➢ While the example CSS in Figure 3.22 uses pixels for its measurement, many contemporary designers prefer to use percentages or em units for widths and heights.

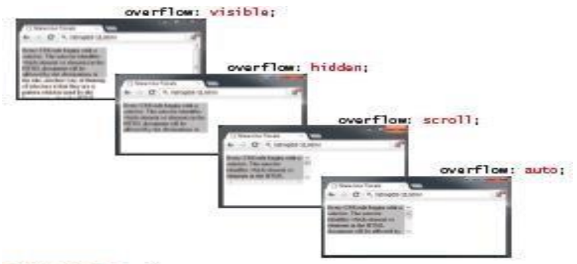


```
p {
    background-color: silver;
}
```

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

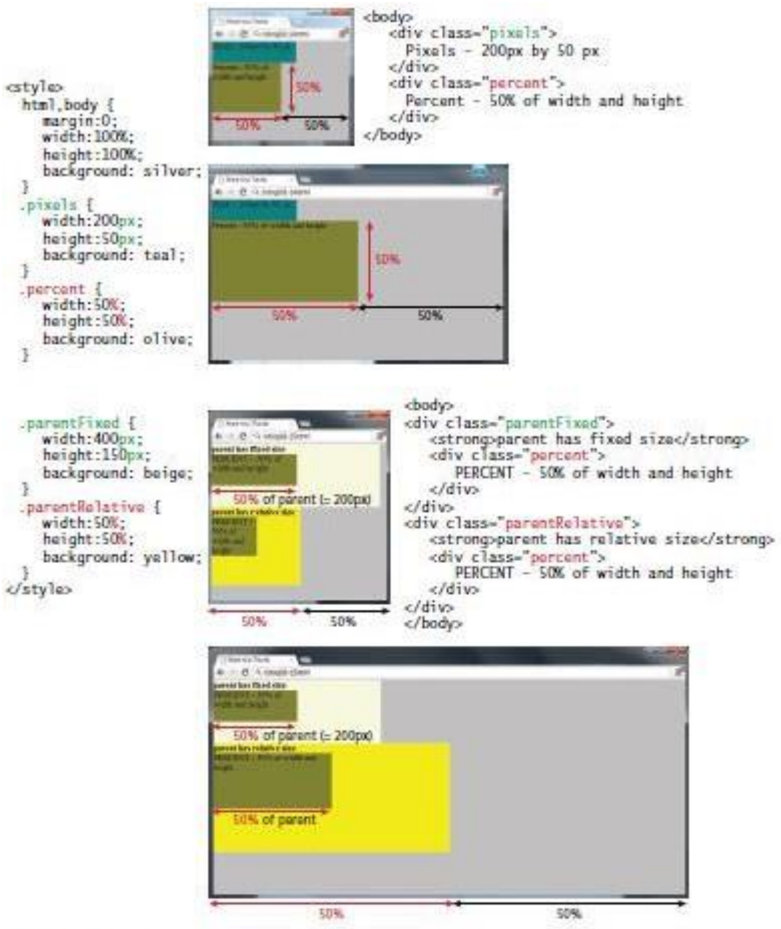
FIGURE 3.22 Limitations of height property

- 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, elements might become too small or too large.



```
overflow: visible;
overflow: hidden;
overflow: scroll;
overflow: auto;
```

FIGURE 3.23 Overflow property



```
<style>
  html,body {
    margin:0;
    width:100%;
    height:100%;
    background: silver;
  }
  .pixels {
    width:200px;
    height:50px;
    background: teal;
  }
  .percent {
    width:50%;
    height:50%;
    background: olive;
  }
  .parentFixed {
    width:400px;
    height:150px;
    background: beige;
  }
  .parentRelative {
    width:50%;
    height:50%;
    background: yellow;
  }
</style>
```

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

```
<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>
```

FIGURE 3.24 Box sizing via percents

FIGURE 3.25 Inspecting CSS using developer tools within modern browsers

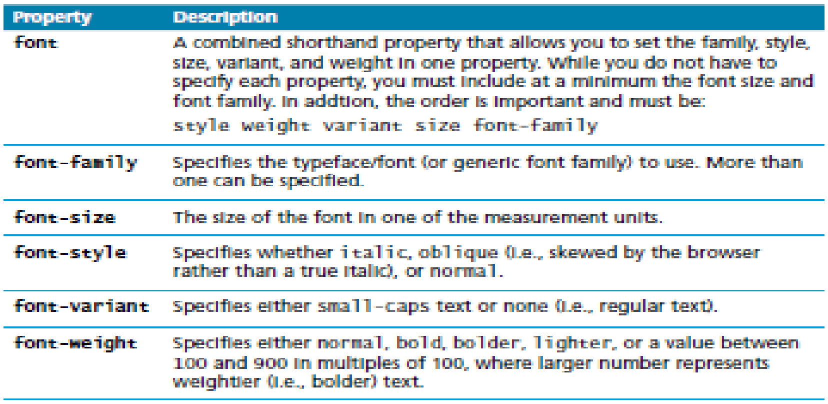
3.7 CSS Text Styling

- ➢ 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.

3.7.1 Font Family

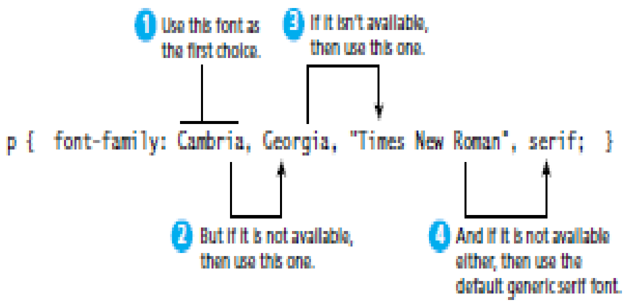
- The first of these problems involves specifying the font family. A word processor on a desktop machine can make use of any font that is installed on the computer; browsers are no different.



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 3.9 Font Properties

- 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 choiceis not on the user's computer.



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

1 Use this font as the first choice.
3 If it isn't available, then use this one.
2 But if it is not available, then use this one.
4 And if it is not available either, then use the default generic serif font.

FIGURE 3.26 Specifying the font family

- The font-family property supports five different generic families; the browser supports a type facefrom each family.

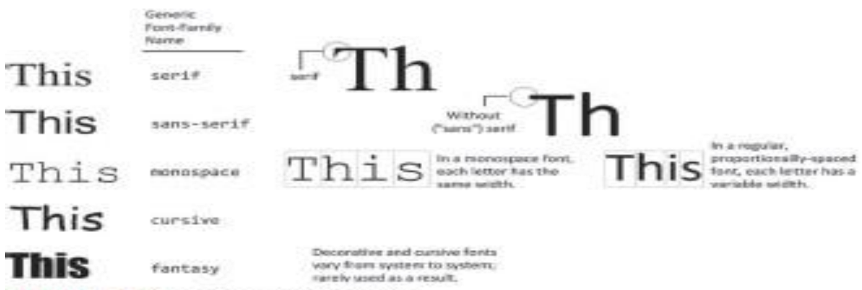

Generic Font-Family Name
This
serif
serif
Th
This
sans-serif
Without ("sans") serif
Th
This
monospace
This
In a monospace font, each letter has the same width.
This
In a regular, proportionally-spaced font, each letter has a variable width.
This
cursive
This
fantasy
Decorative and cursive fonts vary from system to system; rarely used as a result.

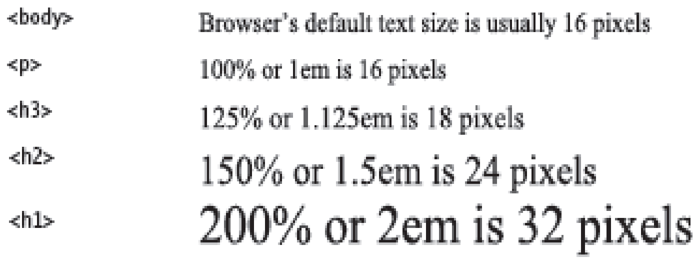
FIGURE 3.27 The different font families

- While there is no real limit to the number of fonts that one can specify with the font- family property, in practice, most developers will typically choose three orfour stylistically similar fonts One common approach is to make your font stack contain, in this order, thefollowing: *ideal*, *alternative*, *common*, and then *generic*. Take for instance, the followingfont stack:font-family { "Hoefler Text", Cambria, "Times New Roman", serif; }
- Times New Roman is installed on almost all PCs and Macsso it is a safe *common* choice.

3.7.2 Font Sizes

- Another potential problem with web fonts is font sizes. In a print-based program such as a word processor, specifying a font size is unproblematic.

- Absolute units such as points and inches do not translate very well to pixel-based devices. Somewhat surprisingly, pixels are also a problematic unit.



<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%; }
p { font-size: 1em; }           /* 1.0  x 16 = 16 */
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>
  Browser's default text size is usually 16 pixels
  <p>100% or 1em is 16 pixels</p>
  <h3>125% or 1.125em is 18 pixels</h3>
  <h2>150% or 1.5em is 24 pixels</h2>
  <h1>200% or 2em is 32 pixels</h1>
</body>
```

FIGURE 3.28 Using percents and em units for 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.

```
<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 3.29 Complications in calculating percents and em units

```
/* 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 */
```

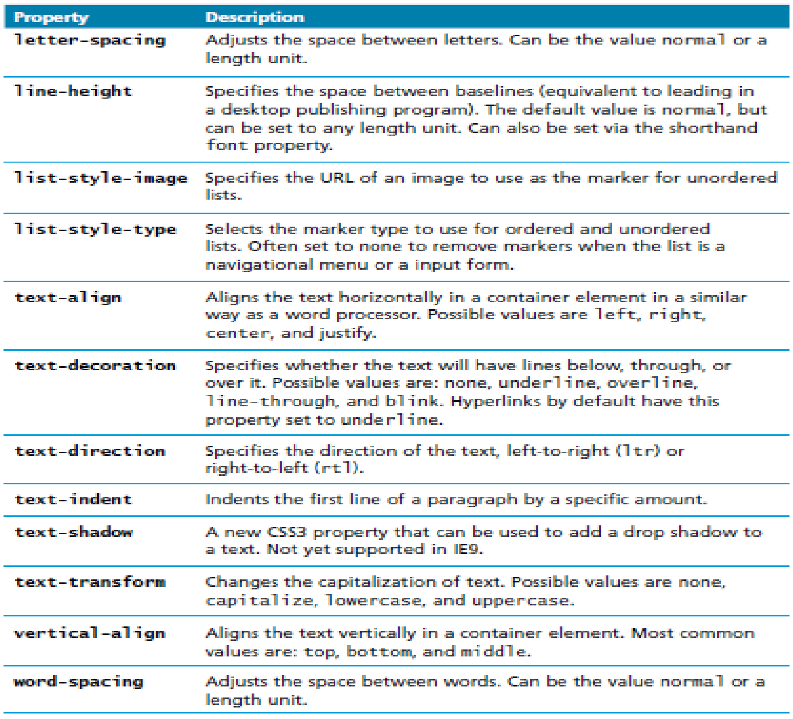
FIGURE 3.30 Using rem units

- For this reason, CSS3 now supports a new relative measure, the **rem** (for rootem unit). This unit is always relative to the size of the root element (i.e., the <html>element).

However, since early versions of Internet Explorer (prior to IE9) do notsupport the rem units

3.7.3 Paragraph Properties

- Just as there are properties that affect the font in CSS, there are also a range of CSSproperties that affect text independently of the font.
- Many of the most common text properties are shown in Table 3.10, 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 3.10 Text Properties