

1

Web Basics and Overview

1.1 What is the Internet?

Internet is a global network of computers linked together for the exchange of information. Also referred to as the 'Net'.

Imagine a bunch of powerful computers, each located in different cities all over the world. These computers are connected by high-speed network connections, so they can exchange information very quickly. Figure 1 shows how such a network might be configured.

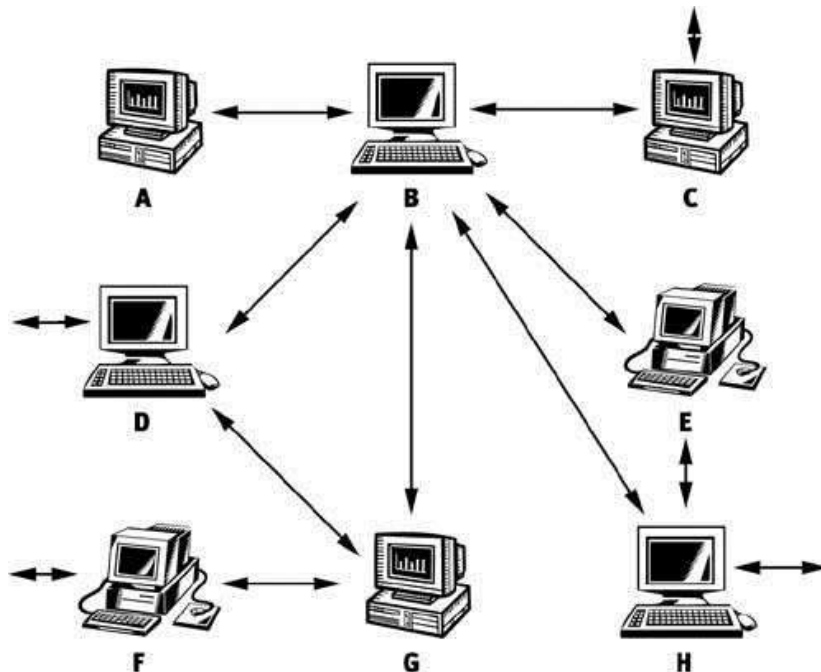


Figure 1: The Internet is a collection of computers connected by high-speed network connections.

These computers are not always directly connected to each other, so sometimes information must pass through one computer to get to another. For example, if Computer A has information that Computer F

wants, the information may need to pass from Computer A through Computers B and G on its way to Computer F.

This is how the Internet works: by passing information to a computer that requests it, from a computer that has it. The information can be anything that can be stored on a computer: like an e-mail message, a Web page, a digital picture, or a computer program.

1.1.2 Internet Features

The Internet offers access to many features. Here are just a few that interest most Internet users:

- **E-mail.** Electronic mail makes it possible to exchange written messages with other people all over the world, quickly and cost effectively.
- **Software.** FTP (file transfer protocol) makes it possible to exchange computer programs or documents with others, via upload or download.
- **Discussion groups.** Newsgroups and mailing lists let participants join in topical discussions with people who share their interests.
- **"Published" information.** Gopher, WAIS, and World Wide Web servers make it possible to publish and retrieve information from a wide variety of sources.

1.2 The World Wide Web

The World Wide Web—or simply the Web—is a part of the Internet that provides information, using documents that include one or more of these components:

- **Formatted text.** Formatted text makes it more attractive and easier to read and understand.
- **Images.** Graphic images make Web information more visually appealing or can provide information that can't be provided with text. Other multimedia elements—including sounds, movies, and even interactive games—can also be found on the Web.
- **Hyperlinks.** Links enable you to navigate from one piece of information to another with just a click.
- **Forms.** Fill-in forms enable you to provide information that can be used to find information that interests you (in the case of a search form) or stored in a database (in the case of a database entry form).

1.2.1 The World Wide Web Environment

- **Web pages:** A Web page is a document, typically written in (X)HTML, that is almost always accessible via HTTP, a protocol that transfers information from the Web server to display in the user's Web browser.
- **Web site:** A website (alternatively, web site or Web site) is a collection of Web pages, images, videos or other digital assets that

is hosted on one or more web servers, usually accessible via the Internet.

- **Web servers:** computers that web pages are stored on.
- **Web clients:** Computers reading the web pages.
- **Web browser:** a program with which web clients view the pages with, i.e. Internet Explorer, Netscape Navigator, Opera, FireFox, Google Chrome and other.



Figure 2: Logos of Some Browsers

2013	<u>Internet Explorer</u>	<u>Firefox</u>	<u>Chrome</u>	<u>Safari</u>	<u>Opera</u>
January	14.3 %	30.2 %	48.4 %	4.2 %	1.9 %
2012	<u>Internet Explorer</u>	<u>Firefox</u>	<u>Chrome</u>	<u>Safari</u>	<u>Opera</u>
December	14.7 %	31.1 %	46.9 %	4.2 %	2.1 %
November	15.1 %	31.2 %	46.3 %	4.4 %	2.0 %
October	16.1 %	31.8 %	44.9 %	4.3 %	2.0 %
September	16.4 %	32.2 %	44.1 %	4.2 %	2.1 %
August	16.2 %	32.8 %	43.7 %	4.0 %	2.2 %
July	16.3 %	33.7 %	42.9 %	3.9 %	2.1 %
June	16.7 %	34.4 %	41.7 %	4.1 %	2.2 %
May	18.1 %	35.2 %	39.3 %	4.3 %	2.2 %
April	18.3 %	35.8 %	38.3 %	4.5 %	2.3 %
March	18.9 %	36.3 %	37.3 %	4.4 %	2.3 %
February	19.5 %	36.6 %	36.3 %	4.5 %	2.3 %
January	20.1 %	37.1 %	35.3 %	4.3 %	2.4 %

Figure 3: Browser Statistics

1.2.2 How does the Browser Fetch the Pages?

- A browser fetches a Web page from a server **by a request**.
- A request is a standard **HTTP** (Hypertext Transfer Protocol) request containing a **page address**.
- A page address looks like this: **http://www.someone.com/page.htm**. This web address is called **URL** (Uniform Resource Locator).

Every URL has several parts. The first part provides the type of request. For Web pages, this is usually **http**. The second part provides the name of the server. This usually begins with **www**, but doesn't have to. And it ends with **.com**, **.org**, **.net**, or other dot-letters. The last part provides the path and name of the Web page. If this last part is omitted, the Home page is assumed.

- All Web pages contain instructions on **how to be displayed**.
- The browser displays the page by **reading these instructions**.
- The most common display instructions are called **HTML tags**. HTML tags look like this:

`<p>This is a Paragraph</p>`

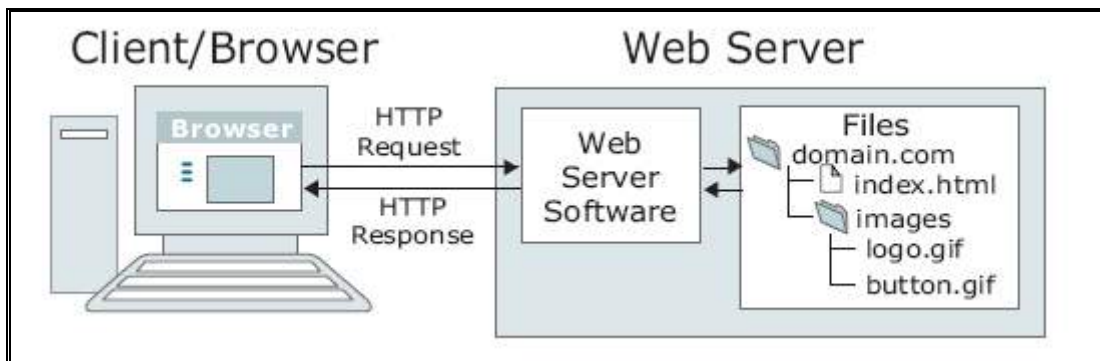


Figure 4: Browser Fetch a Web Page.

1.2.3 What is an HTML File?

- HTML stands for **Hyper Text Mark-up Language**
- An HTML file is a text file containing small **mark-up tags** that tell the Web browser **how to display** the page
- An HTML file must have an **htm** or **html** file extension
- An HTML file can be created using a **simple text editor**

1.2.4 The Page Appears

The page begins to appear in the Web browser window. How quickly it appears depends on several factors:

- How fast is the user's connection to the Internet?
- How fast is the server's connection to the Internet?
- How large are the files that make up the Web page?
- How many requests is the server handling at once?
- How much traffic is on the 'Net, slowing things down?

(Depending on how the server is configured, it can send multiple files at the same time—even to multiple Web browsers—so with fast connections, page elements can pop up quickly.)

1.2.5 Website styles

Mainly there are two types of website styles, Static and Dynamic website:

1.2.5.1 Static Website

A Static Website is one that has web pages stored on the server in the same form as the user will view them. It is primarily coded in HTML (Hyper-text Markup Language).

1.2.5.2 Dynamic website

A Dynamic Website is one that does not have web pages stored on the server in the same form as the user will view them. Instead, the web page content changes automatically and/or frequently based on certain criteria. It generally collates information on the hop each time a page is requested.

There are two meanings for a dynamic website. The first is that the web page code is constructed dynamically, piece by piece. The second is that the web page content displayed varies based on certain criteria. The criteria may be pre-defined rules or may be based on variable user input.

1.3 Web Design

Before you get started you must answer the following questions...

- What sites are you working on?
- What's the purpose of your site?
- What audience(s) are you trying to reach?

1.3.1 Site Planning

1. **Determine** site goals
2. **Analyze** your audience
3. **Analyze** the “competition”
4. **Know** your own abilities and resources
5. **Map** the current site
6. **Sketch** a storyboard

7. **Design** your new site
8. **Select** a Web service provider (web hosting)
9. **Select** a domain name
10. **Publish** the website
11. **Try** it out
12. **Maintain** the site
13. **Promote** your Web site

1.3.1.1 Site Goals and Guidelines

- Why are you creating this site?
- What does the site owner hope to achieve with this site?
- What action does the site owner want the audience to take as a result of visiting?
- What restrictions or guidelines must be followed when designing the site?

1.3.1.2 Audience Analysis

- **Who** are you trying to reach?
 - Age
 - Language and Culture
 - Level of education
 - Access to the Web (High-speed? Dial-in?)
 - Familiarity with the Web
- **What** are they looking for at your site?
 - Information
 - Services
 - Community

1.3.1.3 Analyze the “Competition”

- Look for sites with similar contents, purpose
- What are the trends and precedents?
- Where do they shine or fall short?
- Does your site need to “match” a parent site?

1.3.1.4 Know Resources & Abilities

- What technical knowledge do you have?
- What tools, resources, and time do you have access to (now AND later)?

- Software
- Web authoring tools
- Image editing and tools
- Animation tools
- Hardware
 - Camera (video and/or still)
 - Scanner (flatbed or slide)
- Other people

1.3.1.5 Site Map

- All the pages, all the links of the current site
- Boxes for pages, lines for links
- Shows how “deep” your site is

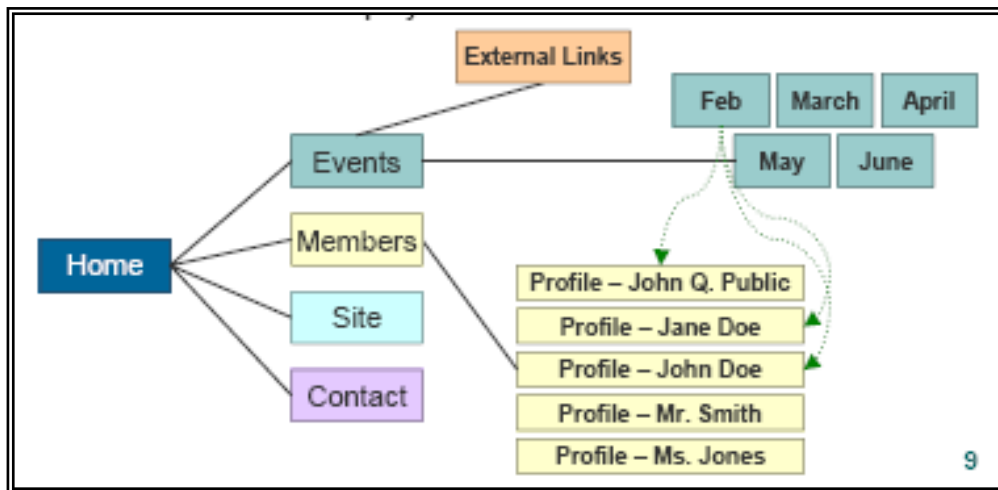


Figure 5: Site Map

Web sites are built around basic structural themes. These fundamental architectures govern the navigational interface of the Web site and mold the user's mental models of how the information is organized. Three essential structures can be used to build a Web site: sequences, hierarchies, and webs.

Sequences

The simplest way to organize information is to place it in a sequence. Sequential ordering may be chronological, a logical series of topics progressing from the general to the specific, or alphabetical, as in indexes, encyclopedias, and glossaries. Straight sequences are the most appropriate organization for training sites, for example, in which the reader is expected to go through a fixed set of material and the only links are those that support the linear navigation path:

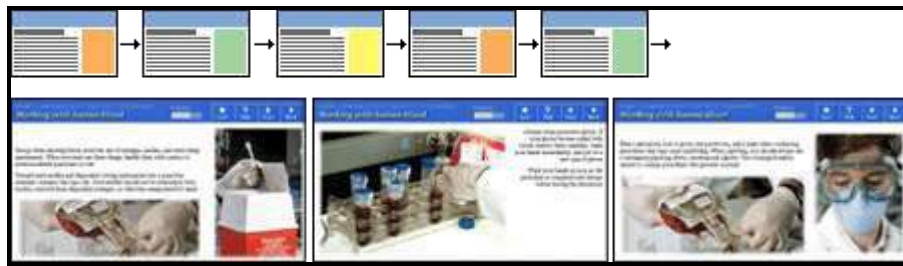


Figure 6: Sequential Ordering

More complex Web sites may still be organized as a logical sequence, but each page in the main sequence may have links to one or more pages of digressions, parenthetical information, or information on other Web sites:

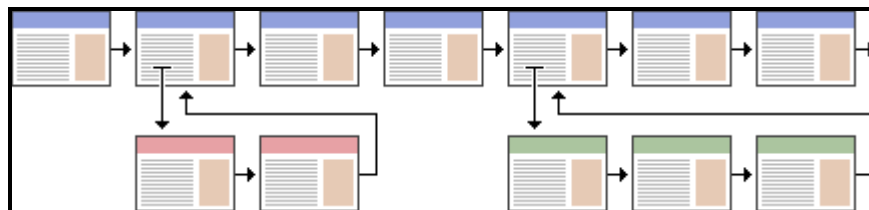


Figure 7: More Complex Sequential Ordering

Hierarchies

Information hierarchies are the best way to organize most complex bodies of information. Because Web sites are usually organized around a single home page, hierarchical schemes are particularly suited to Web site organization. Hierarchical diagrams are very familiar in corporate and institutional life, so most users find this structure easy to understand. A hierarchical organization also imposes a useful discipline on your own analytical approach to your content, because hierarchies are practical only with well-organized material.

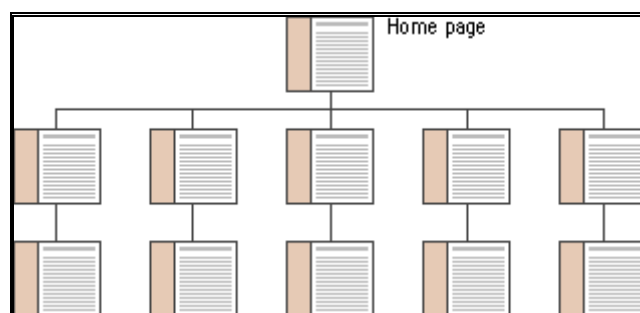


Figure 8: Hierarchies Structure.

Webs

Weblike organizational structures pose few restrictions on the pattern of information use. In this structure the goal is often to mimic associative thought and the free flow of ideas, allowing users to follow their interests in a unique, heuristic, idiosyncratic pattern. This organizational pattern develops with dense links both to information elsewhere in the site and to information at other sites. Although the goal of this organization is to exploit the Web's power of linkage and association to the fullest, weblike structures can just as easily propagate confusion.

Ironically, associative organizational schemes are often the most impractical structure for Web sites because they are so hard for the user to understand and predict. Webs work best for small sites dominated by lists of links and for sites aimed at highly educated or experienced users looking for further education or enrichment and not for a basic understanding of a topic.

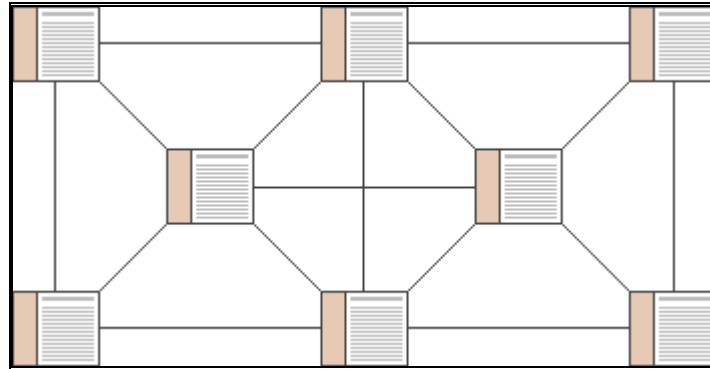


Figure 9: Weblike Structure.

Most complex Web sites share aspects of all three types of information structures. Except in sites that rigorously enforce a sequence of pages, users are likely to use your site in a free-form weblike manner, just as they would a reference book. But the nonlinear usage patterns typical of Web surfers do not absolve you of the need to organize your thinking and present it within a clear, consistent structure that complements your design goals. The chart below summarizes the three basic organization patterns against the "linearity" of the narrative and the complexity of the content:

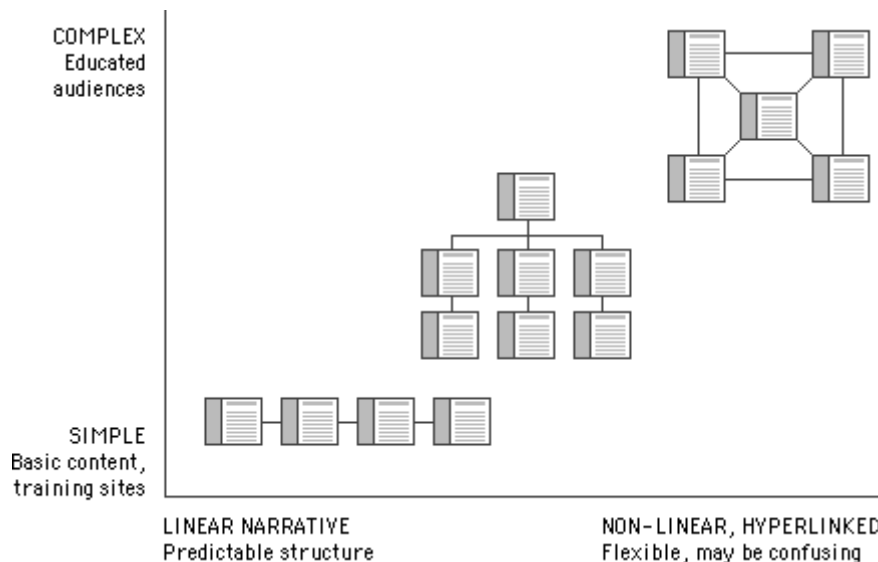


Figure 10: Linearity of the Narrative and the Complexity of the Content

1.3.1.6 Storyboards

Sketch an interaction sequence, minimal page level detail. The purpose of a grayscale storyboard is to show the layout and underlying grid of the page: where the logo, navigation, page title, content, and main images

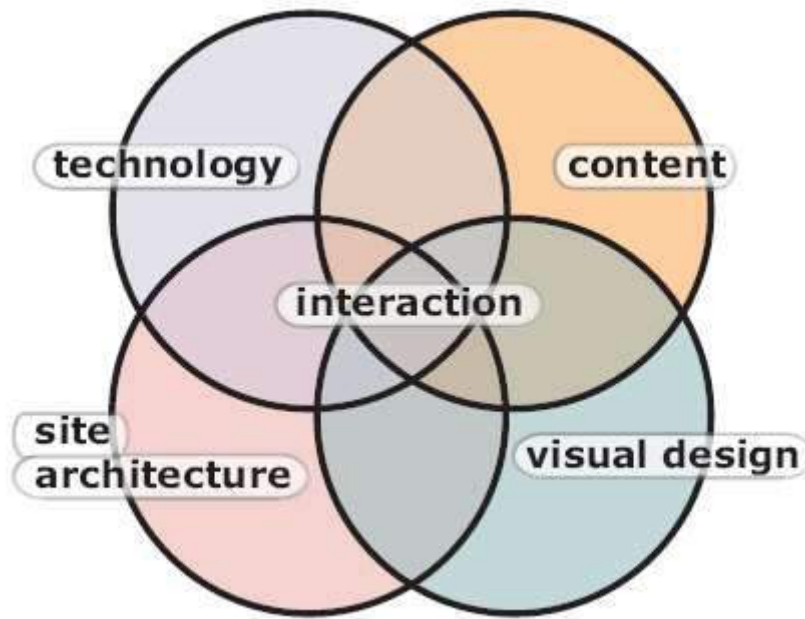


Figure 13: Web design component.

1.3.1.8 Web hosting

A web hosting service is a type of Internet hosting service that allows individuals and organizations to provide their own website accessible via the World Wide Web. Web hosts are companies that provide space on a server they own for use by their clients as well as providing Internet connectivity, typically in a data center

A customer needs to evaluate the requirements of the application to choose what kind of hosting to use. Such considerations include database server software, scripting software, and operating system. Most hosting providers provide Linux-based web hosting which offers a wide range of different software. A typical configuration for a Linux server is the LAMP platform: Linux, Apache, MySQL, and PHP/Perl/Python. The webhosting client may want to have other services, such as email for their business domain, databases or multi-media services for streaming media. A customer may also choose Windows as the hosting platform. The customer still can choose from PHP, Perl, and Python but may also use ASP.Net or Classic ASP.

1.3.1.9 Domain name

The address for your Web site is called its *domain name*. It's what visitors need to know to find your Web site. For example, the domain name of our college is www.scbaghdad.edu.iq.

When you search for a domain name, you need to determine not only the first part of the name but also the ending. Table 1 provides a list of common domain name endings, their purposes, and any restrictions.

Table 1: Domain Name Endings

Domain ending	Purpose	Restriction
.com	Commercial organizations	No restrictions; by far the most popular domain ending
.net	Internet services	No restrictions; used increasingly by people who didn't get the .com name they wanted
.org	Nonprofit organizations	No restrictions
.biz	Businesses	No restrictions; one of the newer domains, and it is increasingly used by businesses when they can't get the .com domain name
.name	Individuals	No restrictions
.info	Informational sites	No restrictions
.pro	Professionals, such as doctors and attorneys	No restrictions; in the process of being established
.aero	Air-transport industry	Restricted
.coop	Cooperative associations	Restricted
.museum	Museums	Restricted
.gov	United States government	Restricted
.edu	Accredited colleges and universities	Restricted
.mil	United States military use	Restricted

The .com domain has emerged as the most valuable because it is the best recognized and the one people are most likely to remember. However, all these domains work the same way in terms of directing users to a Web site address. For example, www.scbaghdad.com, www.scbaghdad.net, and www.scbaghdad.org work the same way on the Internet.

Nearly every country in the world now has its own domain. Country domains include domains such as .am for Armenia, .br for Brazil, .uk for the United Kingdom, .iq for Iraq, .sa for Saudi Arabia, and .zw for Zimbabwe.

1.3.1.10 publishing

After you build your Web site on your own computer, you transfer the site's files to your Web server by using a technology protocol called FTP. The File Transfer Protocol (FTP) describes the system used to transfer any files from one computer to another on the Internet. The term is often used as a verb, as in, "Can you FTP that graphic to the Web server?" A number of freeware and shareware programs have been designed for file transfers, such as Fetch for the Macintosh and WS_FTP, FileZilla, CoffeeCup for Windows.

To upload — or FTP — your site, you just copy files from your computer to your server. Always, web authoring tools have built-in FTP capabilities, making this process almost automatic. You can also use an FTP program to transfer files from your Web server to your computer. Simply select the files in the window that represents your server and click the arrow that points to the window that represents your computer.

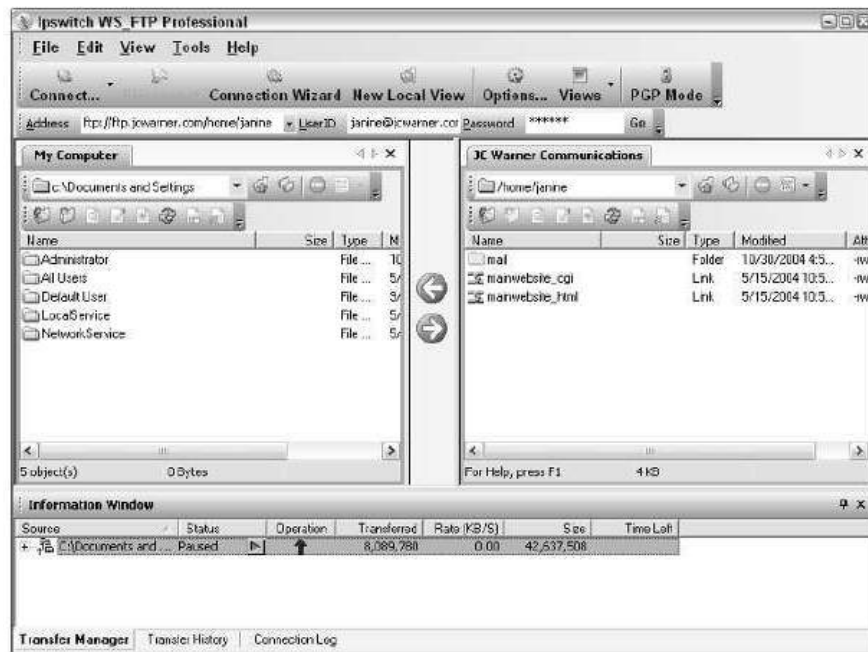


Figure 14: You can use a program like WS_FTP to transfer files from your computer's hard drive to your Web server.

1.4 Search Engine

Search engine is an online utility that enables Web surfers to find Web sites based on search criteria they provide.

1.4.1 Robots and Spiders and Crawlers

Many search engines—such as HotBot (<http://www.hotbot.com/>) and WebCrawler (<http://www.webcrawler.com/>), Google (<http://www.google.com/>)—utilize software programs called robots, spiders, or crawlers. These programs venture out onto the Web when activity is low (whenever that is) and search for new pages. When they find a site they've never visited before, they follow all the internal links within the pages to learn about the pages on the site. They also occasionally (but not often enough) revisit sites to update the information they already have

1.4.2 Directories

While some search engines depend solely on robots, spiders, and crawlers, others—such as Excite (<http://www.excite.com/>) and Yahoo! (<http://www.yahoo.com/>)—start with a directory that entries can be manually added to. Some of these directories are static, containing only the information manually entered. Others use directory entries to send out robots, crawlers, and spiders to gather information directly from the Web sites entered. See Figure 16.

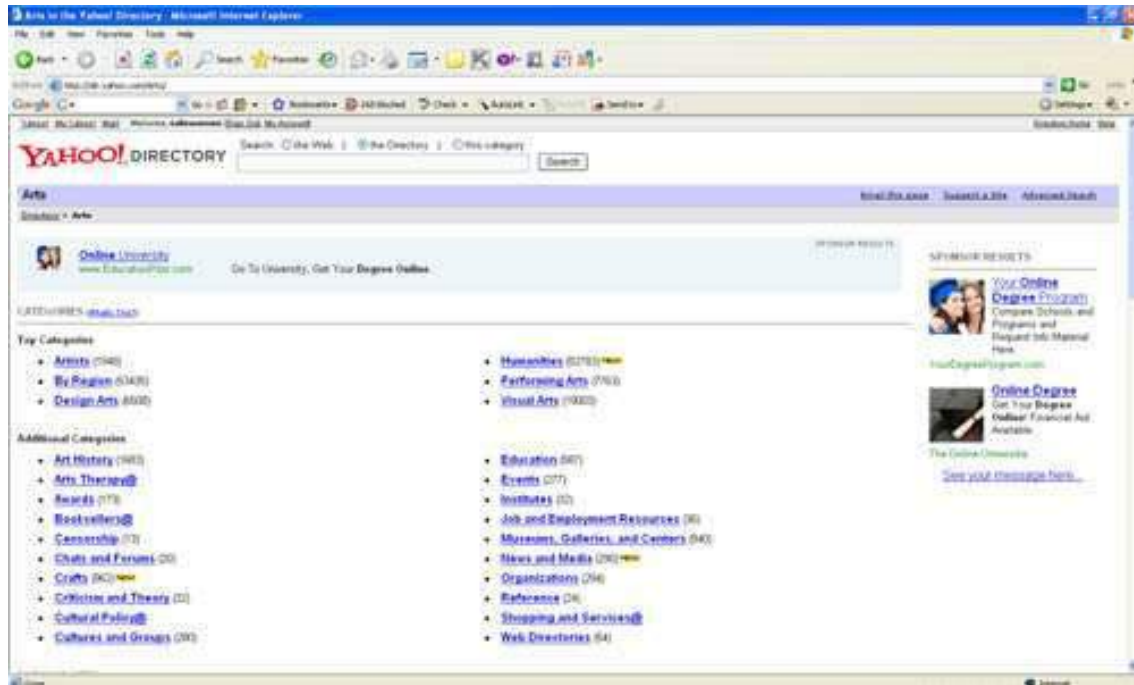


Figure 16: Yahoo! Directory

1.4.3 Meta Tags, Defined

Meta tags are HTML tags that can be inserted in the HEAD part of an HTML document to embed document information. Now, in English: meta

tags enable you to include information about a Web page where it won't be seen by site visitors but can be seen by robots, spiders, and crawlers.

There are many uses for meta tags, but the ones you should be most interested in are for storing description and keyword information. This is the information most often picked up by robots, etc.

1.4.3.1 Description

The description information should be a short, concise description of the site or page. Don't repeat the page title; it's a waste of words. Since you're usually limited to 256 characters, make the most of them.

The format for the description meta tag is as follows:

```
<META name="description" content="concise description of Web site or  
page">
```

When you omit the description meta tag from an HTML document, the first handful of words on the page appear as a description

1.4.3.2 Keywords

The keyword information should be a list of key words or phrases, separated by commas. Again, there's usually a limit of 256 characters for keywords, but not all robots, etc., accept that many. If a robot, etc., is limited to, say, 150 characters, it will take the first 150 characters it finds. For that reason, you should put the most important keywords at the beginning of the meta tag. Also, don't repeat the same words over and over again because most robots, etc., are programmed to ignore repetition—and a few will actually penalize you for it!

The format for the keyword meta tag is as follows:

```
<META name="keywords" content="list of key words and phrases separated by  
commas">
```

1.5 Ten Tips for Testing, Updating, and Promoting Your Site

1. **Send an E-card to Announce Your New Site:** Many people send a simple e-mail message to promote their Web sites. That can work fine, but it's even more fun to send an e-card with a colorful character, animation, and music to dramatize your announcement.
2. **Make Your Site Easy to Find:** Search engines can make it easier for friends and family to find you, but you have to be listed on them first. The good news is that registering your site with most search engines is simple and free. Google and Yahoo!, the two most important search engines, include a link on their home page for adding your URL to

their database. After you add your Web address to a search engine, the address becomes available in the matches when someone searches for your Web site.

3. **Update Regularly:** If you want your visitors to know when to look for updates, consider making changes to your Web site on a regular basis.
4. **Wait Until New Technologies Are Widely Supported:** A big mistake novice Web designers make is to try to use every new tool and technique as it becomes available. Although it may be fun to show off the latest bells and whistles of the Web, most Internet users lag behind new innovations and may not be able to see your handiwork.
5. **Make Your Pages Load Quickly:** No matter how beautiful your site or how many great images and stories it has, your friends and family may never see them if the pages take too long to download.

You can use a number of techniques to help ensure that your pages load quickly for your visitors, but the most important involve reducing the file size of your images and multimedia files. Text loads quickly, but pictures can take a while. Video, audio, and animation files are usually the biggest and take the longest to download.

6. **Test Your Links:** It's relatively easy to create links on a Web site. Unfortunately, it's also relatively easy to break links by moving files around, deleting files or images, or just typing a URL or Web address incorrectly.

As you build your Web site, it's important to check your links, but don't stop there. After you publish your pages to your Web server, you should check them again.

7. **Ask Friends and Family to Test Your Site:** Even though your Web site looks good to you, that doesn't mean it will look good to everyone else.
8. **Make It Easy for Visitors to Contact You:** Don't assume that all your visitors have your contact information.
9. **Test for Accessibility:** Making your Web site accessible means making sure anyone can access the information on your pages. Internet users who are blind or have limited vision use synthetic speech synthesizers to "read" Web sites and refreshable Braille displays that translate Web content. But if you don't design your site to be accessible to these special programs, users may not be able to understand the information on your pages.

Here are a few tips for creating accessible Web sites:

- Use alternative text for images and multimedia. Also known as *alt text*, alternative text is an option in HTML (the language used to create Web pages) that provides a text description if the image or multimedia file can't be displayed. Alternative text

is especially important when you use images as links because screen readers can't decipher text in an image.

- Create links that have clear descriptions about where they go, group related information together, and be consistent about where navigation links are located so they are easy to find on all your pages.
- Choose colors carefully. If you're distinguishing sections of your site with color coding, remember that color differences won't mean anything to someone who can't see them. Also be careful to provide enough contrast between text and background colors so that those who are visually impaired can still decipher the words. For example, light-colored text on a light background is much harder to read than dark text on a light background.
- If you use multimedia, provide transcripts of audio and text descriptions of video and animation files.
- Test your site at the Center for Applied Special Technology site. To use their free service, visit www.cast.org/bobby and enter the address of any Web site into their online testing tool, called Bobby. You get a free report listing errors or omissions that may limit accessibility.
- Try this simple test to see how well the content and links on your site are organized. Imagine that you're reading your Web site to someone over the phone. Start at the top of the page and read all the content from left to right. Does it make sense? How long would it take the person on the other end of the phone to find the information he or she wanted if this was the only way the person could access the information on your site?

10. Visit Other Web Sites for Ideas: One of the best ways to acquire good habits in Web design is to visit other people's Web sites and study what works and what doesn't on their pages.

2

Topics in Web Design: The Basic, Links, and Images

2.1 The World Wide Web Consortium

The World Wide Web Consortium (called the W3C for short, www.w3.org) is the organization that oversees the development of web technologies. The group was founded in 1994 by Tim Berners-Lee, the inventor of the Web, at the Massachusetts Institute of Technology (MIT).



In the beginning, the W3C concerned itself mainly with the HTTP protocol and the development of the HTML. Now, the W3C is laying a foundation for the future of the Web by developing dozens of technologies and protocols that must work together in a solid infrastructure.

2.2 Technologies Associated with Web Development

2.2.1 HTML/XHTML

HTML (HyperText Markup Language) is the language used to create web page documents. The updated version, XHTML (eXtensible HTML) is essentially the same language with stricter syntax rules. It is common to see HTML and XHTML referred to collectively as (X)HTML.

(X)HTML is not a programming language; it is a markup language, which means it is a system for identifying and describing the various components of a document such as headings, paragraphs, and lists.

2.2.2 CSS (Cascading Style Sheets)

While (X)HTML is used to describe the content in a web page, it is Cascading Style Sheets (CSS) that describe how you want that content to look (its presentation). CSS is now the official and standard mechanism for formatting text and page layouts.

CSS also provides methods for controlling how documents will be presented in media other than the traditional browser on a screen, such

as in print and on handheld devices. It also has rules for specifying the non-visual presentation of documents, such as how they will sound when read by a screen reader.

Style sheets are also a great tool for automating production, because you can make changes to all the pages in your site by editing a single style sheet document.

2.2.3 JavaScript/DOM scripting

Despite its name, JavaScript is not at all related to Java. JavaScript is a scripting language that is used to add interactivity and behaviors to web pages, including these (just to name a few):

- Checking form entries for valid entries
- Swapping out styles for an element or an entire site
- Making the browser remember information about the user for the next time they visit

You may also hear the term DOM scripting used in relation to JavaScript. DOM stands for Document Object Model, and it refers to the standardized list of web page elements that can be accessed and manipulated using JavaScript (or another scripting language). DOM scripting is an updated term for what used to be referred to as DHTML (Dynamic HTML), now considered an obsolete approach.

2.2.4 Server-side programming

Some web sites are collections of static (X)HTML documents and image files, but most commercial sites have more advanced functionality such as forms handling, dynamically generated pages, shopping carts, content management systems, databases, and so on. These functions are handled by special web applications running on the server. There are a number of scripting and programming languages that are used to create web applications, including:

- CGI Scripts (written in C+ , Perl, Python, or others)
- Java Server Pages (JSPs)
- PHP
- VB.NET
- ASP.NET
- Ruby on Rails

2.2.5 XML

XML stands for eXtensible Markup Language. XML is not a specific language in itself, but rather a robust set of rules for creating other markup languages.

To use a simplified example, if you were publishing recipes, you might use XML to create a custom markup language that includes the elements <ingredient>, <instructions>, and <servings> that accurately describe the types of information in your recipe documents. Once labeled correctly, that information can be treated as data. In fact, XML has proven to be a powerful tool for sharing data between applications. Despite the fact that XML was developed with the Web in mind, it has actually had a larger impact outside the web environment because of its data-handling capabilities. There are XML files working behind the scenes in an increasing number of software applications, such as Microsoft Office, Adobe Flash, and Apple iTunes.

2.2.6 Java

Although Java can be used for creating small applications for the Web (known as “applets”), it is a complete and complex programming language that is typically used for developing large, enterprise-scale applications.

2.2.7 Ajax

Ajax stands for Asynchronous JavaScript and XML. Ajax is a technique for creating interactive web applications. The significant advantage to using Ajax for web applications is that it allows the content on the screen to change instantly, without refreshing the whole page. This makes using the application more like a desktop program than a web page because controls react instantly, without all that pesky waiting for server calls and page redraws.

2.3 Site Types

2.3.1 Static site

A static site is one where content is relatively fixed, and users are unable to affect the look or scope of the data they view. In short, the visitor has minimal ability to interact with the site’s content other than choosing the order in which to view content.

Accessing a static site is like reading a paper magazine. A user can choose to flip back and forth between pages and read articles in a different order, but the presentation is relatively rigid.

2.3.2 Dynamic site

A dynamically generated page is created at request or view time for the user.

There are numerous benefits to dynamically generated pages. First the content can be customized to suit what the user may be looking for. Search result pages are a common form of dynamic page. Dynamic pages can also be created to take into account browsing conditions or technology restrictions. For example, a static site has only one form of

presentation that all users must deal with, while a dynamic site may have multiple forms optimized for different browsers or bandwidth levels. Dynamically generated sites often use a database to store site content. In these sites, pages are constructed from content merged into page templates at request time to create the final page for delivery.

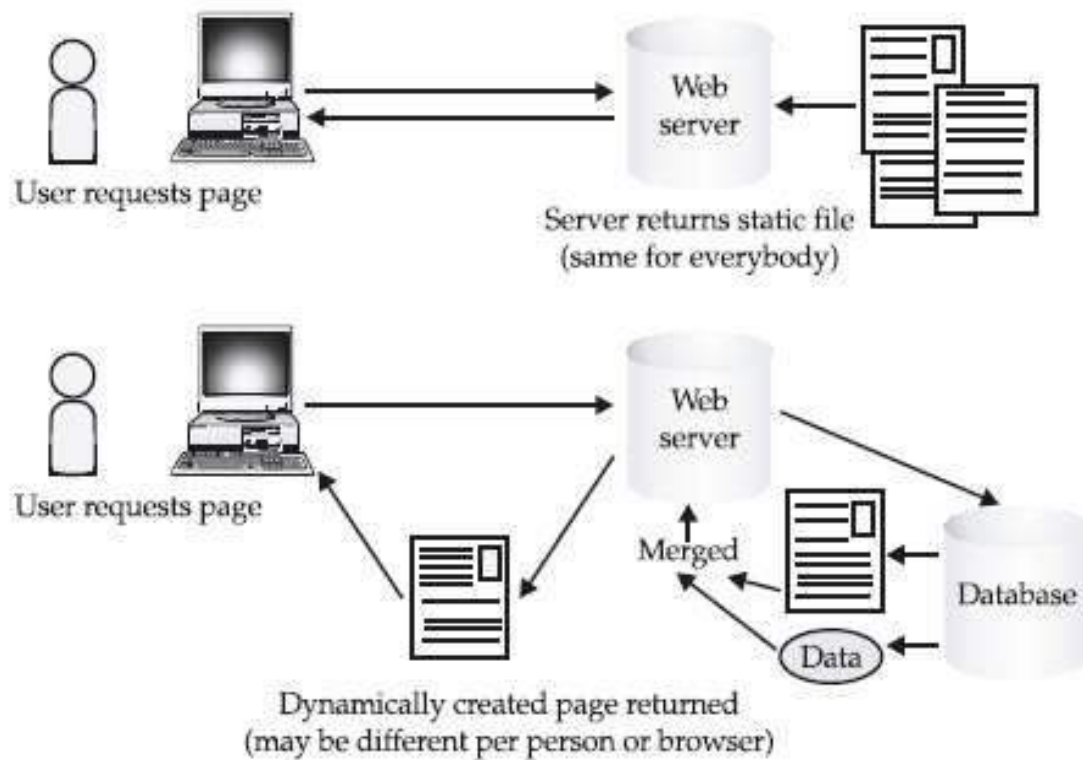


Figure 2.1: Statically and dynamically generated sites

2.4 Web Page Addresses (URLs)

With all those web pages on all those servers, how would you ever find the one you're looking for? Fortunately, each document has its own special address called a URL (Uniform Resource Locator).

A complete URL is generally made up of three components: the protocol, the site name, and the absolute path to the document or resource, as shown in Figure 2.2.

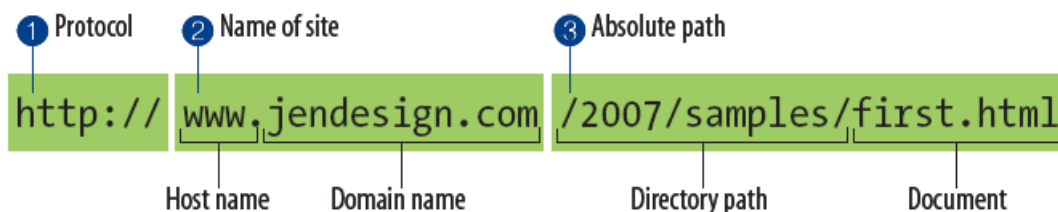


Figure 2.2: The parts of a URL.

❶ `http://`

The first thing the URL does is define the protocol that will be used for that particular transaction. The letters HTTP let the server know to use Hypertext Transfer Protocol.

Note that, sometimes you'll see a URL that begins with `https://`. This is an indication that it is a secure server transaction. Secure servers have special encryption devices that hide delicate content, such as credit card numbers, while they are transferred to and from the browser.

❷ `www.jendesign.com`

The next portion of the URL identifies the web site by its domain name. In this example, the domain name is `jendesign.com`. The “`www.`” part at the beginning is the particular host name at that domain. The host name “`www`” has become a convention, but is not a rule. In fact, sometimes the host name may be omitted. There can be more than one web site at a domain (sometimes called subdomains). For example, there might also be `development.jendesign.com`, `clients.jendesign.com`, and so on.

❸ `/2007/samples/first.html`

This is the absolute path to the requested HTML document, `first.html`. The words separated by slashes indicate the pathway through directory levels, starting with the root directory of the host, to get to `first.html`.

To sum it up, the example URL says it would like to use the HTTP protocol to connect to a web server on the Internet called `www.jendesign.com` and request the document `first.html` (located in the `samples` directory, which is in the `2007` directory).

2.5 Default files

Obviously, not every URL you see is so lengthy. Many addresses do not include a file name, but simply point to a directory, like these:

- `http://www.oreilly.com`
- `http://www.jendesign.com/resume/`

When a server receives a request for a directory name rather than a specific file, it looks in that directory for a default document, typically named `index.html`, and sends it back for display. So when someone types in the above URLs into their browser, what they'll actually see is this:

- `http://www.oreilly.com/index.html`
- `http://www.jendesign.com/resume/index.html`

The name of the default file (also referred to as the index file) may vary, and depends on how the server is configured. In these examples, it is named `index.html`, but some servers use the file name `default.htm`. If

your site uses server side programming to generate pages, the index file might be named `index.php` or `index.asp`.

Another thing to notice is that in the first example, the original URL did not have a trailing slash to indicate it was a directory. When the slash is omitted, the server simply adds one if it finds a directory with that name. The index file is also useful for security. Some servers (depending on their configuration) return the contents of the directory for display in the browser if the default file is not found. Figure 2.3 shows how the documents of the `housepics` directory are exposed as the result of a missing default file. One way to prevent people snooping around in your files is to be sure there is an index file in every directory. Your system administrator may also add other protections to prevent your directories from displaying in the browser.

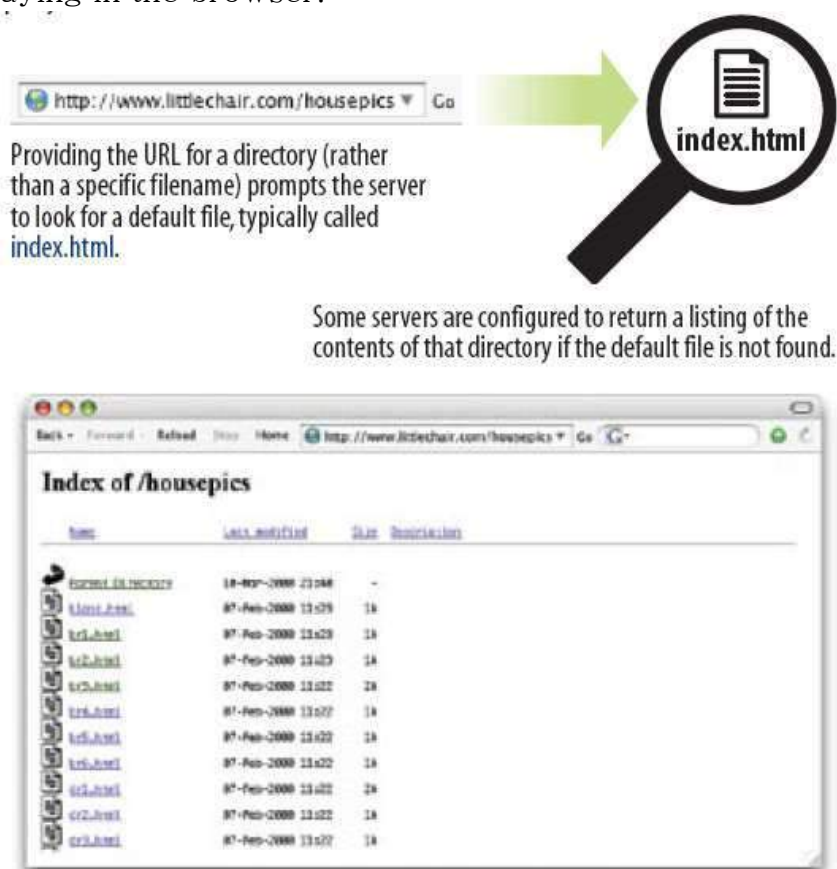


Figure 2.3: Some servers display the contents of the directory if an index file is not found.

2.6 How the Web Works

Let's trace the stream of events that occur with every web page that appears on your screen, Figure 2.4.

❶ You request a web page by either typing its URL (for example, `http://jenskitchensite.com`) directly in the browser, or by clicking on a link on the page. The URL contains all the information needed to target a specific document on a specific web server on the Internet.

- ❷ Your browser sends an HTTP Request to the server named in the URL and asks for the specific file. If the URL specifies a directory (not a file), it is the same as requesting the default file in that directory.
- ❸ The server looks for the requested file and issues an HTTP response.
 - If the page cannot be found, the server returns an error message. The message typically says “404 Not Found,” although more hospitable error messages may be provided.
 - If the document is found, the server retrieves the requested file and returns it to the browser.
- ❹ The browser parses the HTML document. If the page contains images, (indicated by the HTML *img* element), the browser contacts the server again to request each image file specified in the markup.
- ❺ The browser inserts each image in the document flow where indicated by the *img* element. Then the assembled web page is displayed for your viewing pleasure.

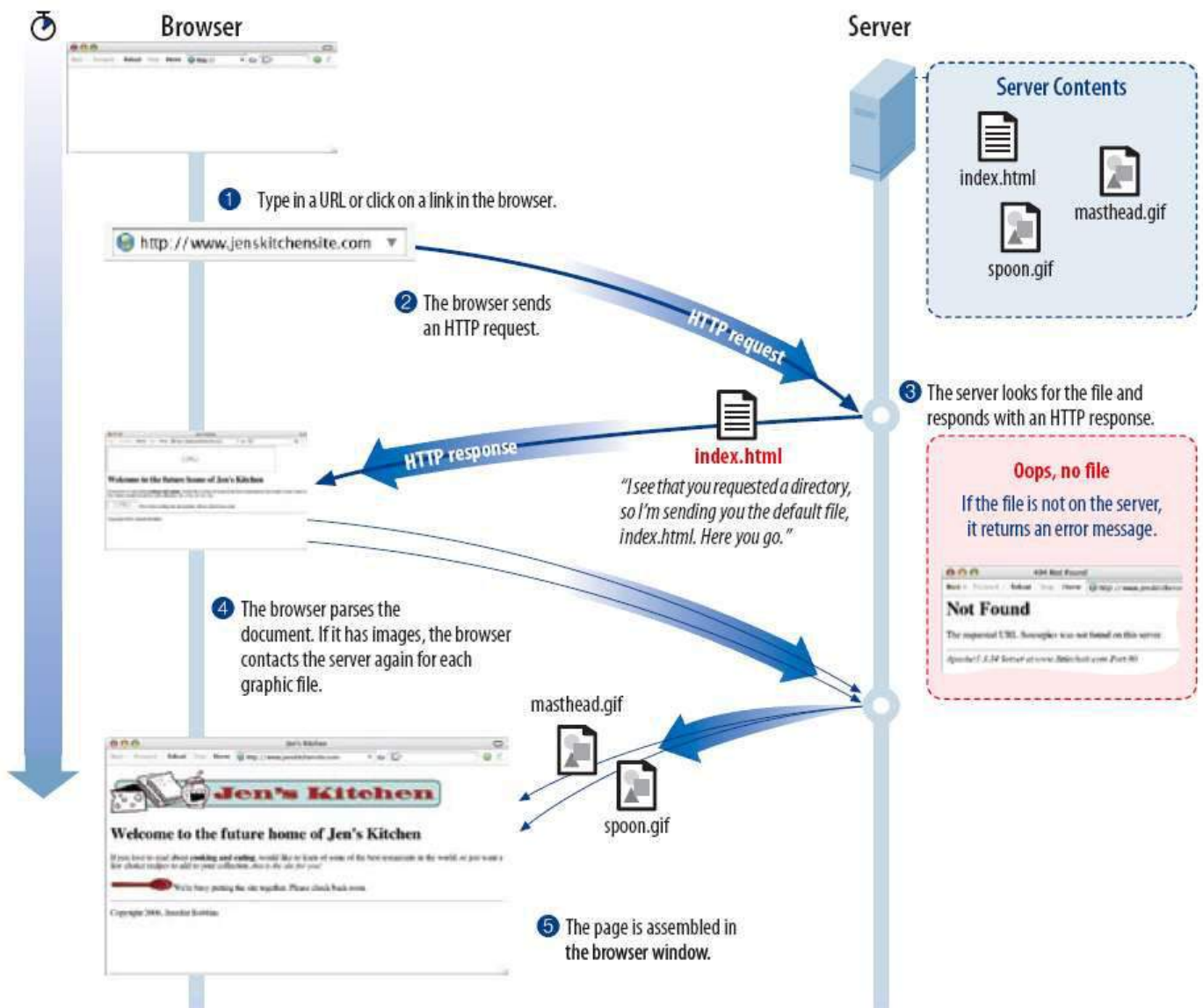


Figure 2.4: How browsers display web pages.

2.7 Coping with various browser versions

Stick with the standards. Following web standards—(X)HTML for document structure and CSS for presentation—as documented by the W3C is your primary tool for ensuring your site is as consistent as possible on all standards-compliant browsers (that’s approximately 99% of browsers in current use).

Start with good markup. When an (X)HTML document is written in logical order and its elements are marked up in a meaningful way, it will be usable on the widest range of browsing environments, including the oldest browsers, future browsers, and mobile and assistive devices. It may not look exactly the same, but the important thing is that your content is available.

Don’t use browser-specific (X)HTML elements. There are markup elements and attributes out there that work only with one browser or another, a remnant from the browser wars of old. Don’t use them! (You won’t learn them here.)

Become familiar with the aspects of CSS that are likely to cause problems. Using style sheets effectively takes some practice, but experienced developers know which properties are “safe,” and which require some extra tweaks to get consistent results on all current browsers.

2.8 Link's URL

There are two ways to specify the URL:

Absolute URLs provide the full URL for the document, including the protocol (`http://`), the domain name, and the pathname as necessary. You need to use an absolute URL when pointing to a document out on the Web.

Example: `href="http://www.oreilly.com/"`

Relative URLs describe the pathname to the linked file relative to the current document. It doesn’t require the protocol or domain name—just the pathname. Relative URLs can be used when you are linking to another document on your own site (i.e., on the same server).

Example: `href="recipes/index.html"`

2.9 Linking Within Your Own Site

A large portion of the linking you’ll do will be between pages of your own site: from the home page to section pages, from section pages to content pages, and so on. In these cases, you can use a relative URL—one that calls for a page on your own server. Consider the following site structure:

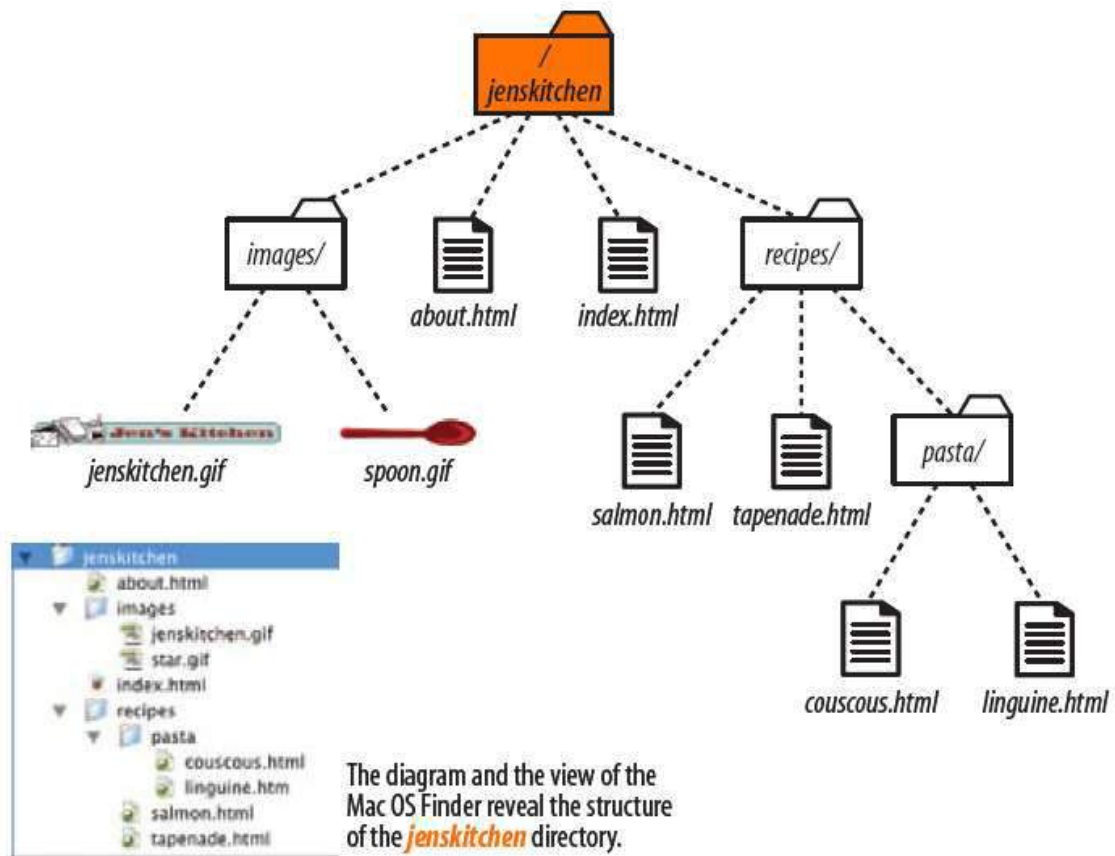


Figure 2.5: A diagram of the jenskitchen site structure

➤ Linking within a directory

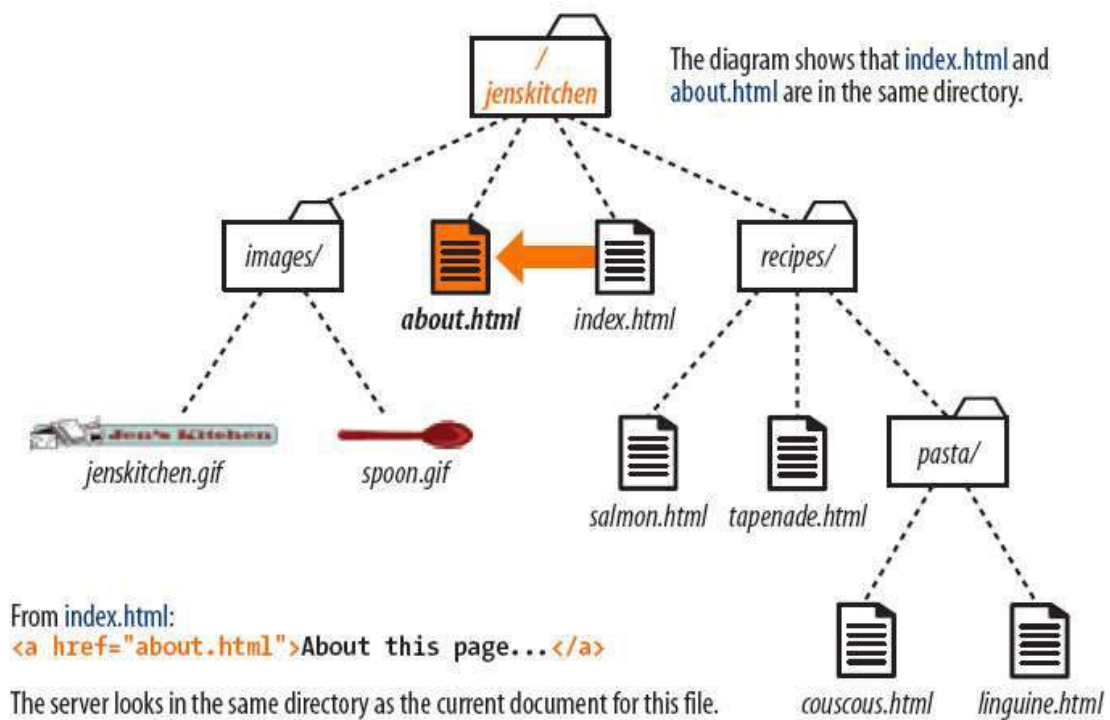


Figure 2.6: Writing a relative URL to another document in the same directory.

➤ Linking to a lower directory

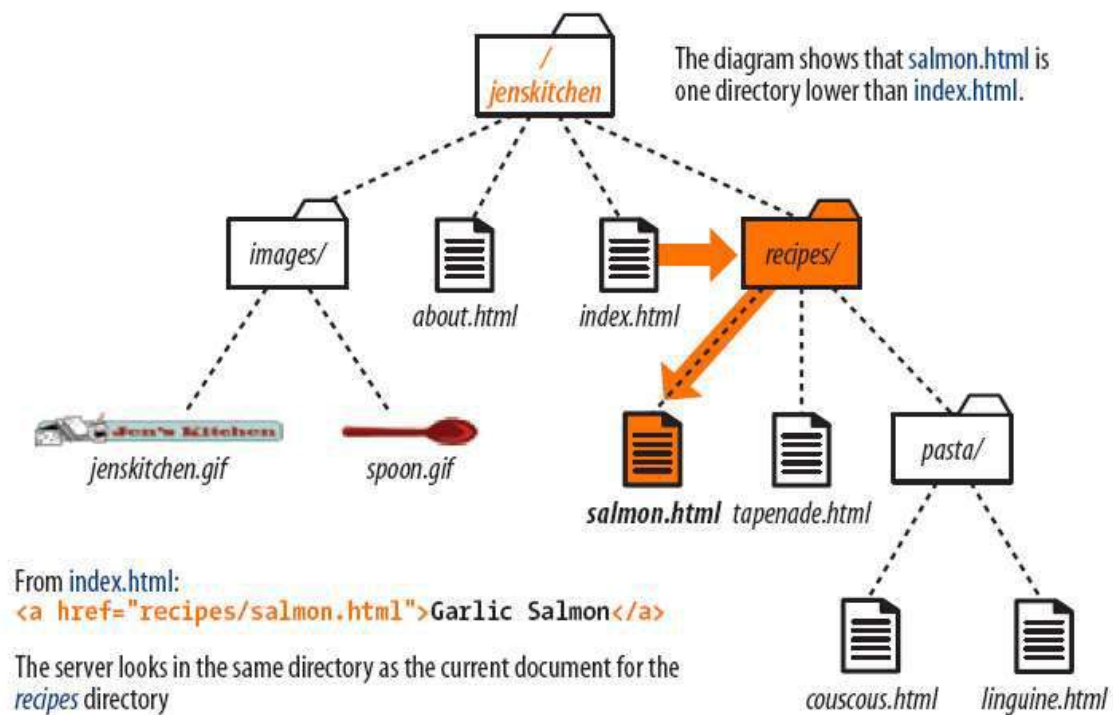


Figure 2.7: Writing a relative URL to a document that is one directory level lower than the current document

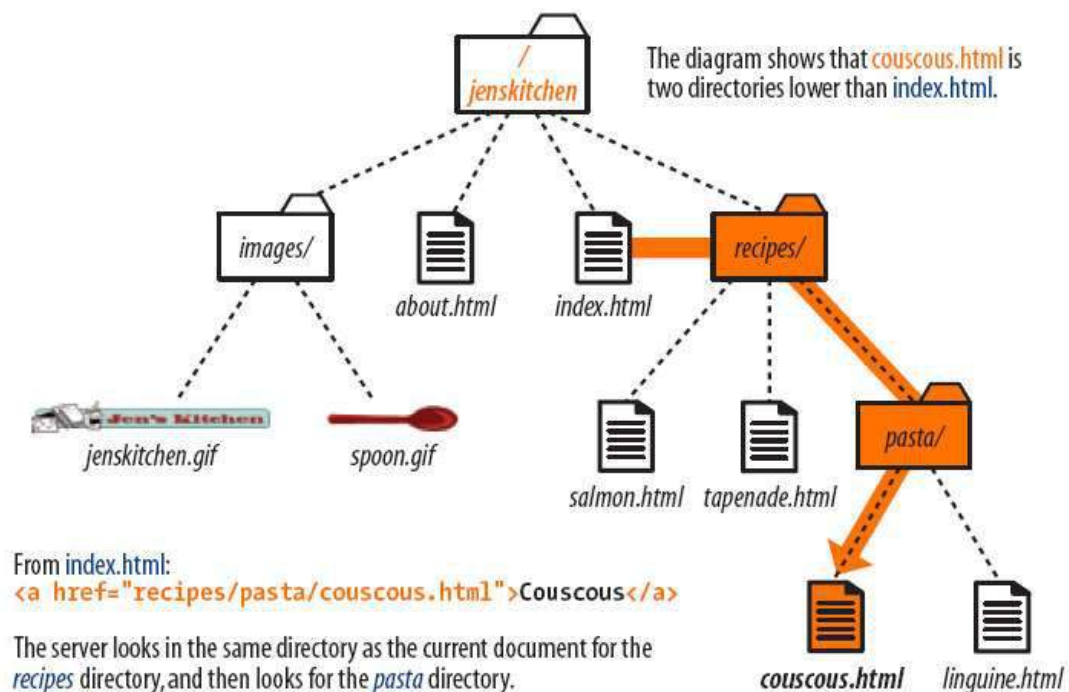


Figure 2.8: Writing a relative URL to a document that is two directory levels lower than the current document.

➤ Linking to a higher directory

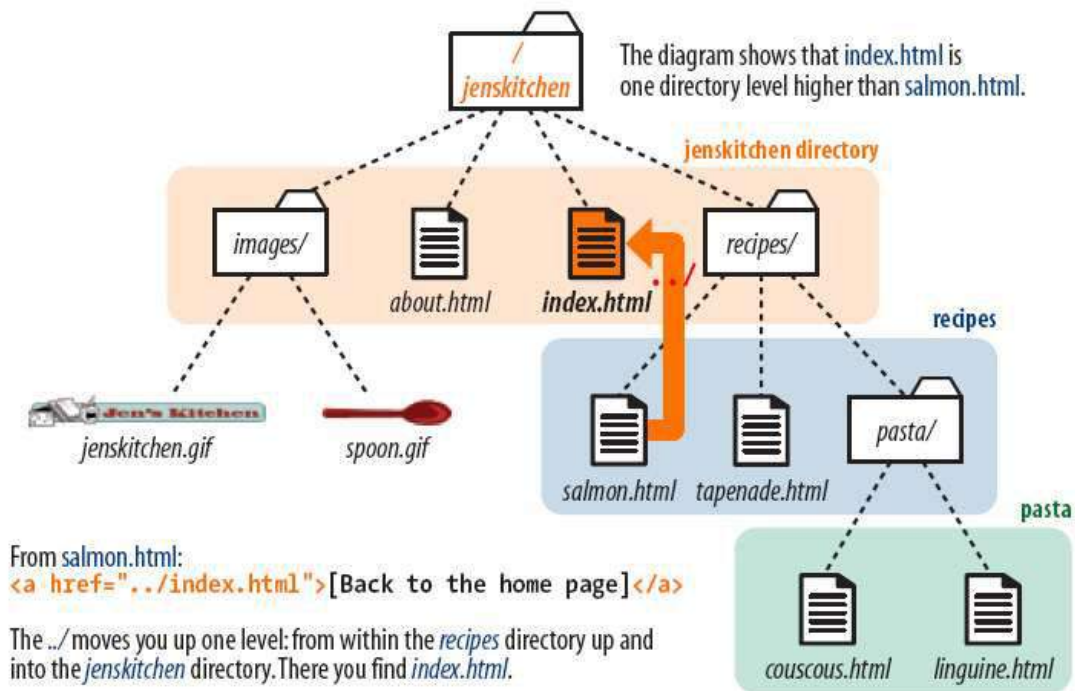


Figure 2.9: Writing a relative URL to a document that is one directory level higher than the current document.

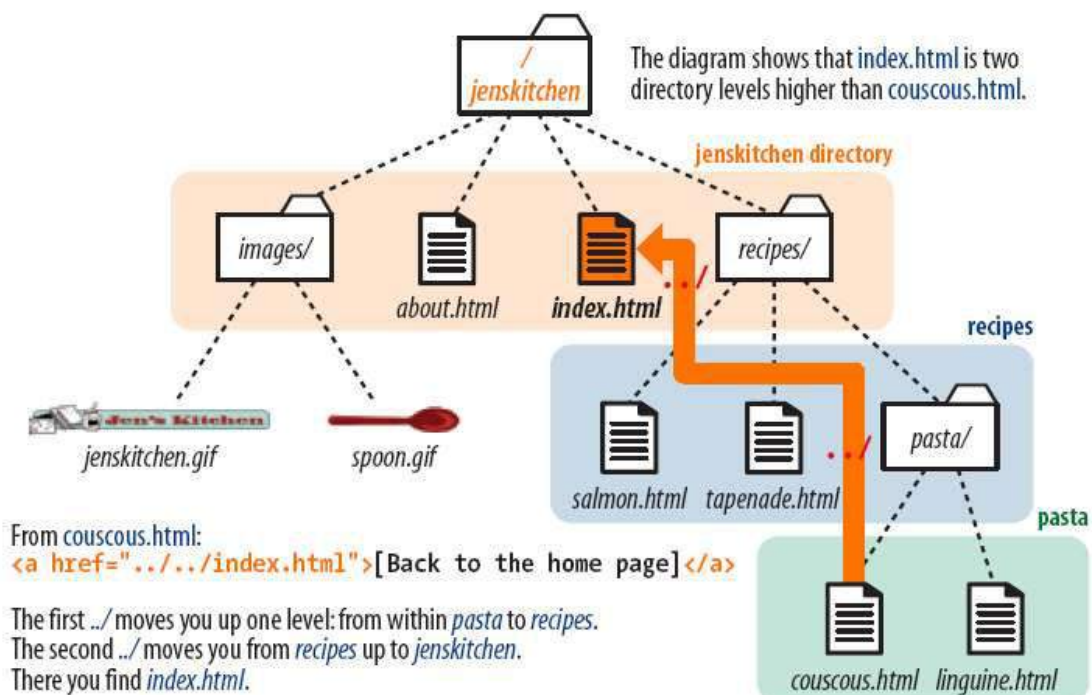


Figure 2.10: Writing a relative URL to a document that is two directory levels higher than the current document.

➤ Site root relative pathnames

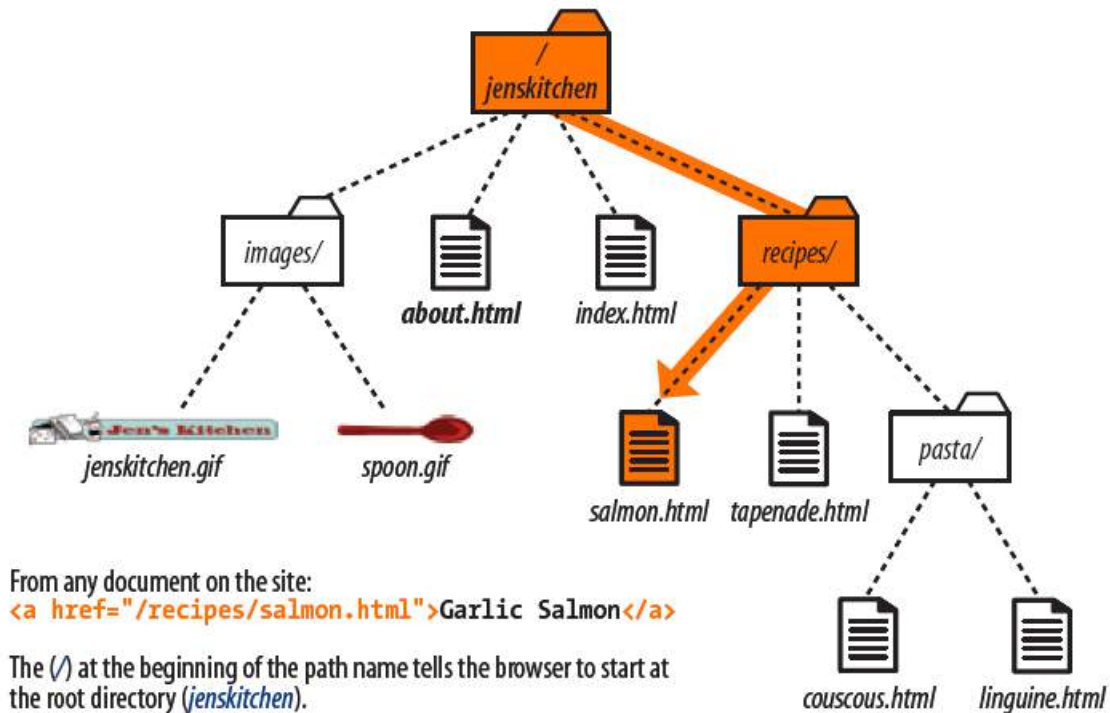


Figure 2.11: Writing a relative URL starting at the root directory.

NOTE: It's the same for images

Important Pathname Don'ts

When you are writing relative pathnames, it is critical that you follow these rules to avoid common errors:

- Don't use backslashes (\). Web URL pathnames use forward slashes (/) only.
- Don't start with the drive name (D:, C:, etc.)._ Although your pages will link to each other successfully while they are on your own computer, once they are uploaded to the web server, the drive name is irrelevant and will break your links.
- Don't start with file://. This also indicates that the file is local and causes the link to break when it is on the server.
- The ../ (or multiples of them) always appears at the beginning of the pathname and never in the middle. If the pathnames you write have ../ in the middle, you've done something wrong.

2.10 Images

It is common to store all the graphics in their own directory (usually called images or graphics). You can make one images directory to store all the graphics for the whole site or create an images directory in each subdirectory (subsection) of the site.

Most Web browsers support either directly or through extension a variety of image formats, such as GIF, JPEG, Flash, and PNG.

- GIF—Use the GIF format for flat color images. These are images with just a few colors.
- JPG—Use the JPG format for photographic images. These are images with millions of colors.
- PNG—Use the PNG format if you don't need your images to display on mobile devices. They are good for both flat color and photographic images. It's best to save your images as both PNG and either JPG or GIF and then use the version that is smaller.

When a browser downloads an image file, it stores it in the disk cache (a space for temporarily storing files on the hard disk). That way, if it needs to redisplay the page, it can just pull up a local copy of the source document and image files without making a new trip out to the remote server.

When you use the same image repetitively in a page or a site, the browser only needs to download the image once. Every subsequent instance of the image is grabbed from the local cache, which means less traffic for the server and faster display for the end user.

The image formats can be separated into two general categories: bitmap (or raster) images and vector images. Raster images describe each individual pixel and its color, while vector images describe an image generally as a collection of mathematical directions used to draw—or more precisely, render—the image. Regardless of storage format, all images become bitmaps onscreen. The fundamental difference between the two general images formats is shown here:

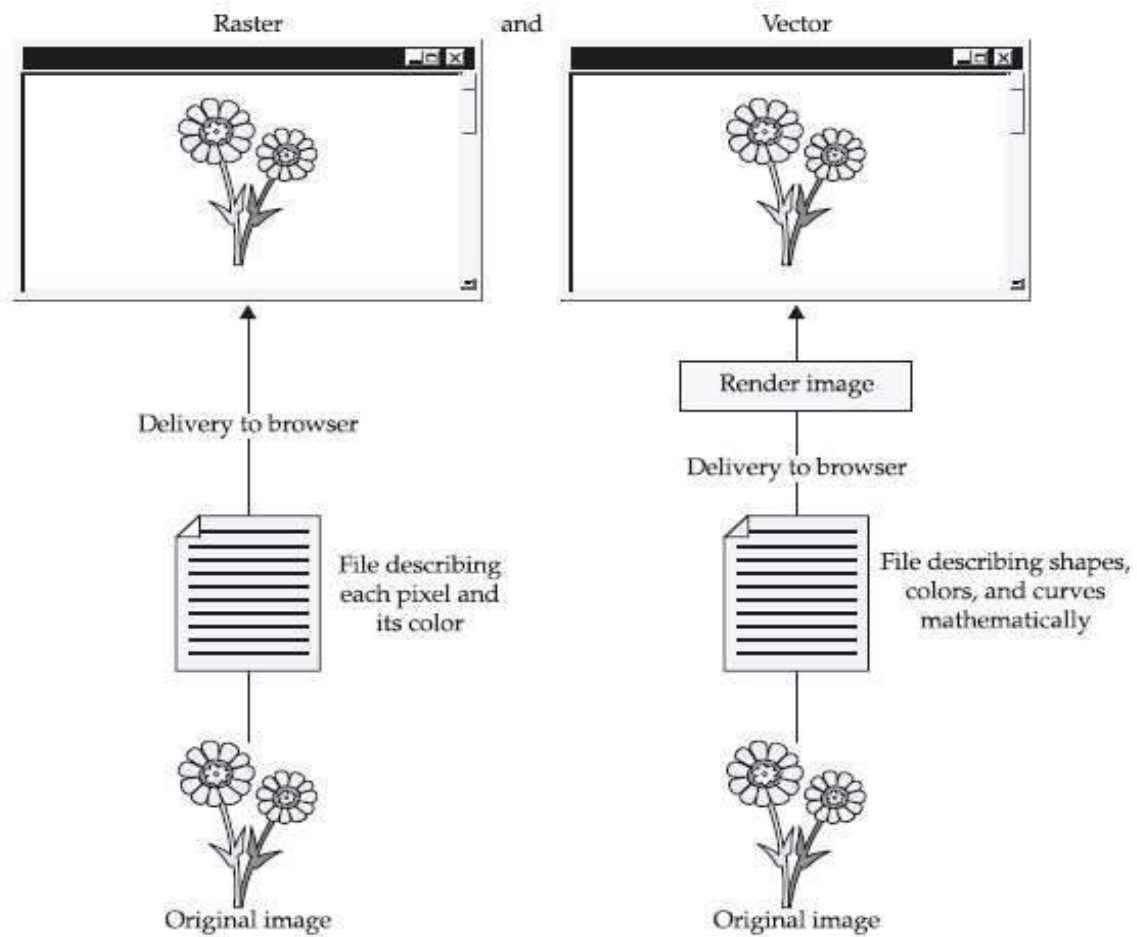


Figure 2.12: Raster and vector images

3

Topics in Web Design: Tables, Forms, and the Standards

3.1 Using Layout Tables

Complex tables were once the norm for creating interesting web page layouts, but now that style sheets offer an alternative, this use of (X)HTML tables is discouraged. Not only are they not semantically sound, but they can be a real hindrance to accessibility. The professional web design community is leaving layout tables in the dust.

If you still choose use table elements to create the grid of the page, follow these guidelines:

- Use only the minimal table elements (table, tr, and td).
- Avoid nesting tables within tables.
- Avoid tricks like empty rows and transparent GIF images used solely for adjusting the spacing.
- Use style sheets to control all presentational aspects of the table and its contents, such as colors, alignment, spacing, and column width.
- Make sure that your content still reads in a logical order in the source document when all of the table markup is removed. Tables that read in a logical order are said to linearize well. This is the way visitors with screen readers will encounter the page.

3.2 Advanced Table Elements

There are some table elements and attributes that offer more complex semantic descriptions and improve the accessibility of tabular content. Take a look of the following sample code:

```

<table summary="A listing of calorie and fat
content for each of the most popular menu items">
  <caption>Nutritional Information</caption>
  <thead>
    <tr>
      <th scope="column">Menu item</th>
      <th scope="column">Calories</th>
      <th abbr="fat" scope="column">Fat (g)</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Chicken noodle soup</td>
      <td>120</td>
      <td>2</td>
    </tr>
    <tr>
      <td>Caesar salad</td>
      <td>400</td>
      <td>26</td>
    </tr>
  </tbody>
</table>

```

Figure 3.1: Advanced Table Elements

Row group elements

You can describe rows or groups of rows as belonging to a header, footer, or the body of a table using the **thead**, **tfoot**, and **tbody** elements respectively. Some user agents (another word for a browsing device) may repeat the header and footer rows on tables that span multiple pages. Authors may also use these elements to apply styles to various regions of a table.

Column group elements

Columns may be identified with the **col** element or put into groups using the **colgroup** element. This is useful for adding semantic context to information in columns and may be used to calculate the width of tables more quickly.

Accessibility features

Accessibility features such as **captions** and **summaries** for providing descriptions of table content, and the **scope** and **headers** attributes for explicitly connecting headers with their respective content.

3.3 How Forms Work

There are two parts to a working form. The first part is the form that you see on the page itself. Forms are made up of buttons, text fields, and pull-down menus (collectively known as form controls) used to collect information from the user. Forms may also contain text and other elements.

The other component of a web form is an application or script on the server that processes the information collected by the form and returns an appropriate response. It's what makes the form work. In other words, putting up an (X)HTML page with form elements isn't enough. Web applications and scripts require programming know-how that is beyond the scope of this course.

In other words, HTML forms allow users to interact with Web documents by providing GUI controls for data entry. The HTML side of forms simply collects the data. A separate handler, usually a script of some sort, is used to do something useful with the data. A typical interaction with an HTML form resembles that shown in Figure 3.2.

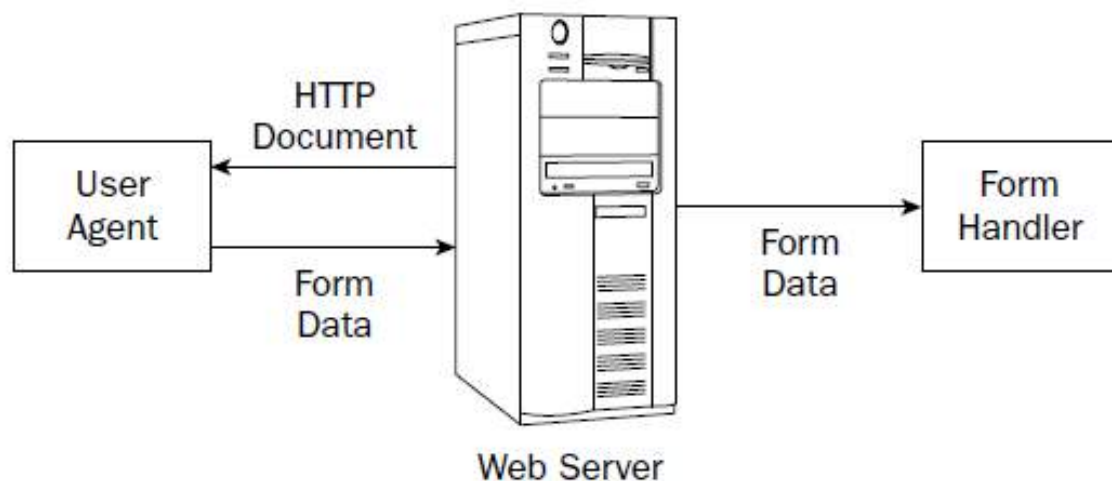


Figure 3.2: How Forms Work.

The steps in the flow are as follows:

- ❶ The Web server sends the HTML document (containing the form) to the user agent.

- ❷ The user uses the form's GUI controls to enter data and submits the completed form.
- ❸ The form is submitted to a specified server (typically the same server that delivered the form document) to be passed to a handler.
- ❹ The server passes the data stream to a specified handler, which uses the data in a prescribed method.

3.3.1 From Data Entry to Response

If you are going to be creating web forms, it is beneficial to understand what is happening behind the scenes. This example traces the steps of a transaction using a simple form that gathers names and email addresses for a mailing list; however, it is typical of the process for most forms.

- ❶ Your visitor, let's call her Sally, opens the page with a web form in the browser window. The browser sees the form control elements in the markup and replaces them with the appropriate form controls, including two text entry fields and a submit button (shown in Figure 3.2).
- ❷ Sally would like to sign up for this mailing list, so she enters her name and email address into the fields and submits the form by clicking the "Submit" button.
- ❸ The browser collects the information she entered, encodes it, and sends it to the web application on the server.
- ❹ The web application accepts the information and processes it (that is, does whatever it is programmed to do with it). In this example, the name and email address are added to a database.
- ❺ The web application also returns a response. The kind of response sent back depends on the content and purpose of the form. Here, the response is a simple web page that contains a thank you for signing up for the mailing list. Other applications might respond by reloading the (X)HTML form page with updated information, by moving the user on to another related form page, or by issuing an error message if the form is not filled out correctly, to name only a few examples.
- ❻ The server sends the web application's response back to the browser where it is displayed. Sally can see that the form worked and that she has been added to the mailing list.

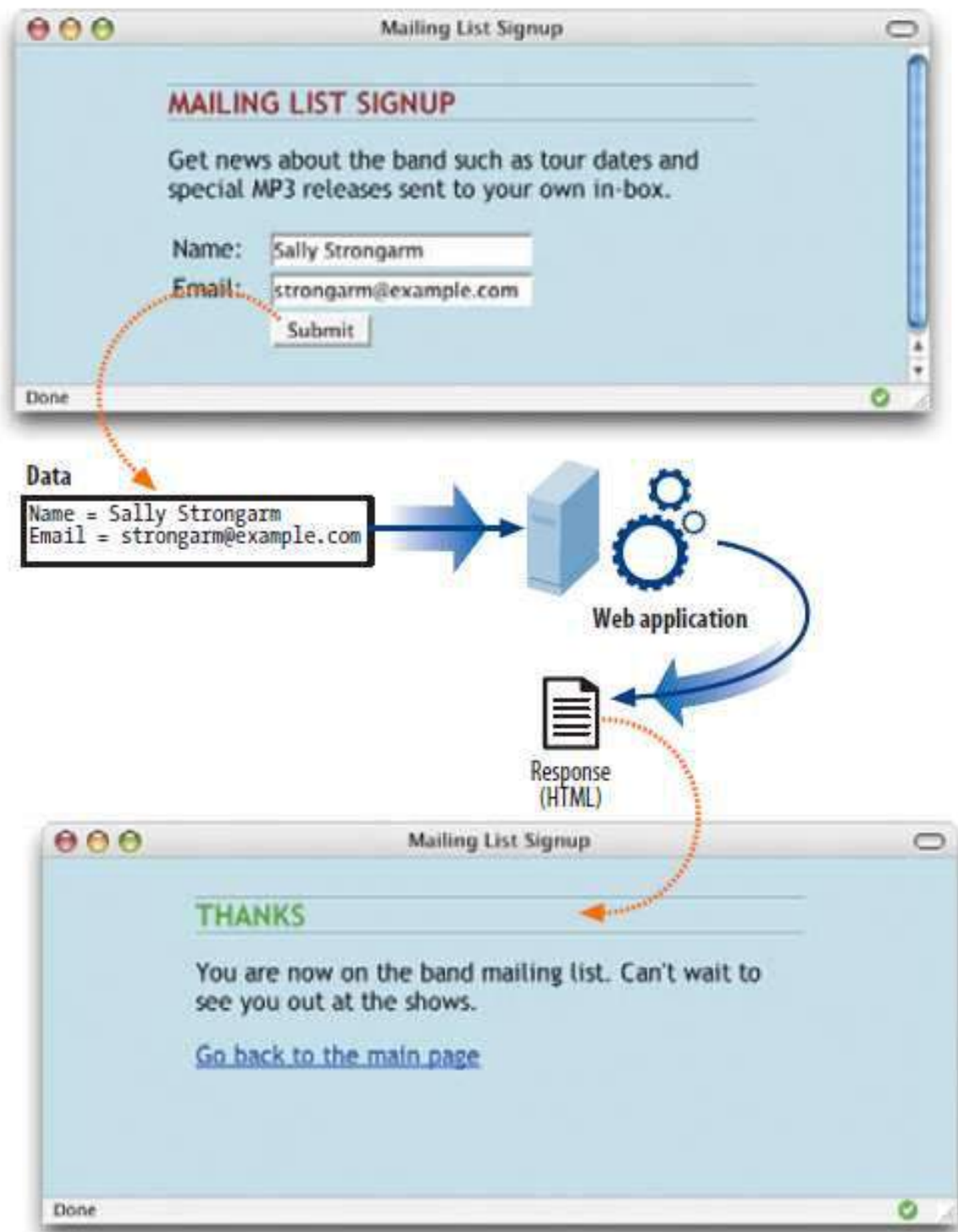


Figure 3.3: What happens behind the scenes when a web form is submitted.

3.3.2 The Action Attribute

The action attribute provides the location (URL) of the application or script (sometimes called the action page) that will be used to process the form. The action attribute in this example sends the data to a script called mailinglist.pl. The script is in the cgi-bin directory on the same server as the HTML document (you can tell because the URL is site root relative).

```
<form action="/cgi-bin/ mailinglist.pl" method="post">...</form>
```

The .pl suffix indicates that this form is processed by a Perl script (Perl is a scripting language). It is also common to see web applications that end with the following:

- .php, indicating that a PHP program is doing the work. PHP is an open source scripting language most commonly used with the Apache web server.
- .asp, for Microsoft's ASP (Active Server Pages) programming environment for the Microsoft Internet Information Server (IIS).
- .jsp, for JavaServer Pages, a Java-based technology similar to ASP.

3.3.3 The Method Attribute

The method attribute specifies how the information should be sent to the server. Let's use this data gathered from the sample form in Figure 3.3 as an example.

```
name = Sally Strongarm  
email = strongarm@example.com
```

When the browser encodes that information for its trip to the server, it looks like this:

```
name=Sally%20Strongarm&email=strongarm%40example.com
```

There are only two methods for sending this encoded data to the server: POST or GET indicated using the method attribute in the form element. We'll look at the difference between the two methods in the following sections. Our example uses the POST method, as shown here:

```
<form action="/cgi-bin/ mailinglist.pl" method="post">...</form>
```

The POST method

When the form's method is set to POST, the browser sends a separate server request containing some special headers followed by the data. Only the server sees the content of this request, thus it is the best

method for sending secure information such as credit card or other personal information.

The POST method is also preferable for sending a lot of data, such as a lengthy text entry, because there is no character limit as there is for GET.

Some other notes on POST requests:

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

The GET method

With the GET method, the encoded form data gets tacked right onto the URL sent to the server. A question mark character separates the URL from the following data, as shown here:

`get http://www.bandname.com/cgi-bin/maillinglist.pl?name=Sally%20Strongarm&email=strongarm%40example.com`

The GET method is appropriate if you want users to be able to bookmark the results of a form submission (such as a list of search results). Because the content of the form is in plain sight, GET is not appropriate for forms with private personal or financial information. In addition, because there is a 256 character limit on what can be appended to a URL, GET may not be used for sending a lot of data or when the form is used to upload a file.

Some notes on GET requests:

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests should be used only to retrieve data

3.4 The Standards

Professional web designers know that the best way to ensure consistency and accessibility across multiple browsers and devices is to write standards compliant web documents. Standards compliance simply means that your documents abide by all of the rules in the latest Recommendations published by the World Wide Web Consortium (the W3C). That includes HTML and XHTML for markup, but also other standards for style sheets (CSS) and accessibility.

3.4.1 HTML Version History

The original HTML draft. Tim Berners-Lee based his original markup language for hypertext documents on the syntax and elements in SGML (Standard Generalized Markup Language). He took many elements such as (p) and (h1) through (h6) right from SGML, then invented the anchor (a) element for adding hypertext links.

HTML +. This version of HTML, written by Dave Raggett in 1993 and 1994, builds upon Berners-Lee's original version, adding elements such as figures, tables, and forms. Many of the ideas developed here made it into later versions, but the specific elements (such as fig for figures) were left behind.

HTML 2.0. This was a proposed standard developed by Tim Berners-Lee and the HTML Working Group at IETF (Internet Engineering Task Force) in 1995. At this point, the Web was still in its infancy. In fact, Microsoft had not yet released its Internet Explorer browser. Netscape, however, had emerged on the scene and was busy adding elements to HTML that worked only on its browser.

HTML 3.2. This is the first Recommendation released by the newly-formed W3C in 1996. It is a snapshot of all the HTML elements in common use at the time and includes many extensions to HTML that were the result of the notorious Browser War between Netscape and Microsoft. Many of these extensions are presentational, and in hindsight, most would say they should have never been incorporated in the standard.

HTML 4.0 and 4.01. HTML 4.0, released as a Recommendation in 1998, got HTML back on track by acknowledging that all matters of presentation should be handled with CSS and remain separate from document markup. Many of the presentational elements and attributes introduced in HTML 3.2 were kept because they were in widespread use, however, they were labeled as deprecated. HTML 4.0 also introduced a number of accessibility and internationalization features. The updated HTML 4.01 Recommendation fixed some small issues and was released in 1999.

Rewriting HTML. W3C had a vision of an XML-based Web with many specialized markup languages working together. The first thing they needed to do was rewrite HTML, the cornerstone of the Web, according to the stricter rules of XML.

The result is XHTML. The “X” stands for “eXtensible” and it indicates its connection to XML. The first version, XHTML 1.0, is nearly identical to HTML 4.01. It shares the same elements and attributes.

XHTML 1.0. is the first in a growing family of XHTML document types that have been released or are being developed by the W3C.

XHTML 1.1. You can think of this as a stricter version of XHTML 1.0 Strict. It is the first version of XHTML to be liberated from legacy HTML by eliminating all elements and attributes that control presentation. XHTML 1.1 documents also must identify themselves as XML applications, not as HTML (as XHTML 1.0 documents may do). Unfortunately, not all browsers support XHTML 1.1 documents, which is why developers opt for XHTML 1.0 as of this writing.

XHTML 2.0. XHTML 2.0 is a rethinking of HTML that includes new elements and new ways of doing things in order to be purely semantic and highly accessible. Because it is an XML language, it is supported by browsers that support XML.

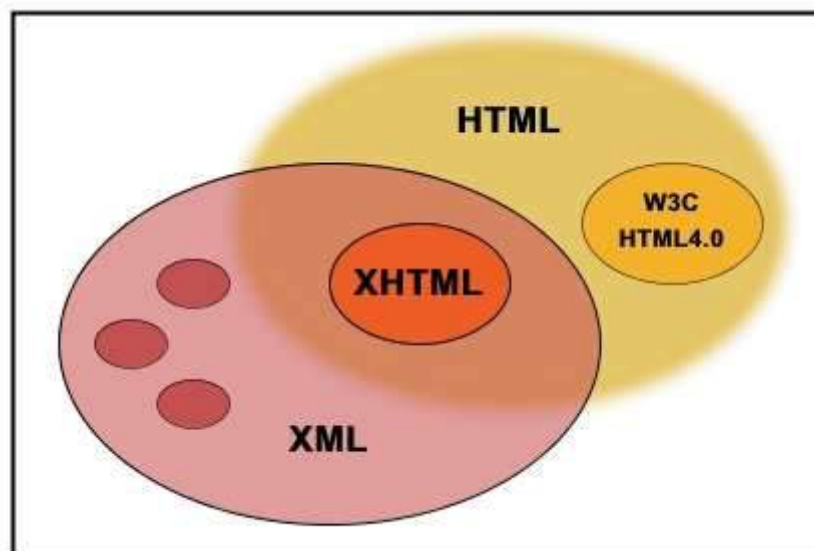


Figure 3.4: HTML, XHTML, and XML

HTML5. HTML5 will be the new standard for HTML. The previous version of HTML, HTML 4.01, came in 1999. The web has changed a lot since then. HTML5 is still a work in progress. However, the major browsers support many of the new HTML5 elements and APIs.

HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).

WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML. And here are some rules for HTML5 were established:

- New features should be based on HTML, CSS, DOM, and JavaScript
- Reduce the need for external plugins (like Flash)
- Better error handling
- More markup to replace scripting
- HTML5 should be device independent
- The development process should be visible to the public

3.4.2 XHTML Syntax

What makes XHTML documents different is that because XHTML is an XML language, correct syntax is critical. Elements must be closed and properly nested, attributes must be in quotation marks, all in the interest of eliminating confusion. Documents with correct XML syntax are said to be well-formed. The following is a checklist of the requirements of XHTML documents.

Element and attribute names must be lowercase.

In HTML, element and attribute names are not case-sensitive, which means that you could write ``, ``, or `` and it's all the same. Not so in XML. When XHTML was written, all element and attribute names were defined as lowercase. Attribute values do not need to be lowercase, except in cases where a predefined list of values is provided for the attribute.

All elements must be closed (terminated).

Although it is okay to omit the closing tag of certain HTML elements (p and li, for example), in XHTML, every element must be closed (or terminated, to use the proper term).

Empty elements must be terminated too.

This termination rule extends to empty elements as well. To do this, simply add a slash before the closing bracket, indicating the element's ending.

In XHTML, a line break is entered as `
`. Some browsers have a problem with this syntax, so to keep your XHTML digestible to all browsers, add a space before the slash (`
`) and the terminated empty element will slide right through.

Attribute values must be in quotation marks.

In XHTML, all attribute values must be contained in quotation marks (in HTML, certain attribute values could get away without them). Single or double quotation marks are acceptable as long as they are used consistently.

Furthermore, there should be no extra white space before or after the attribute value inside the quotation marks.

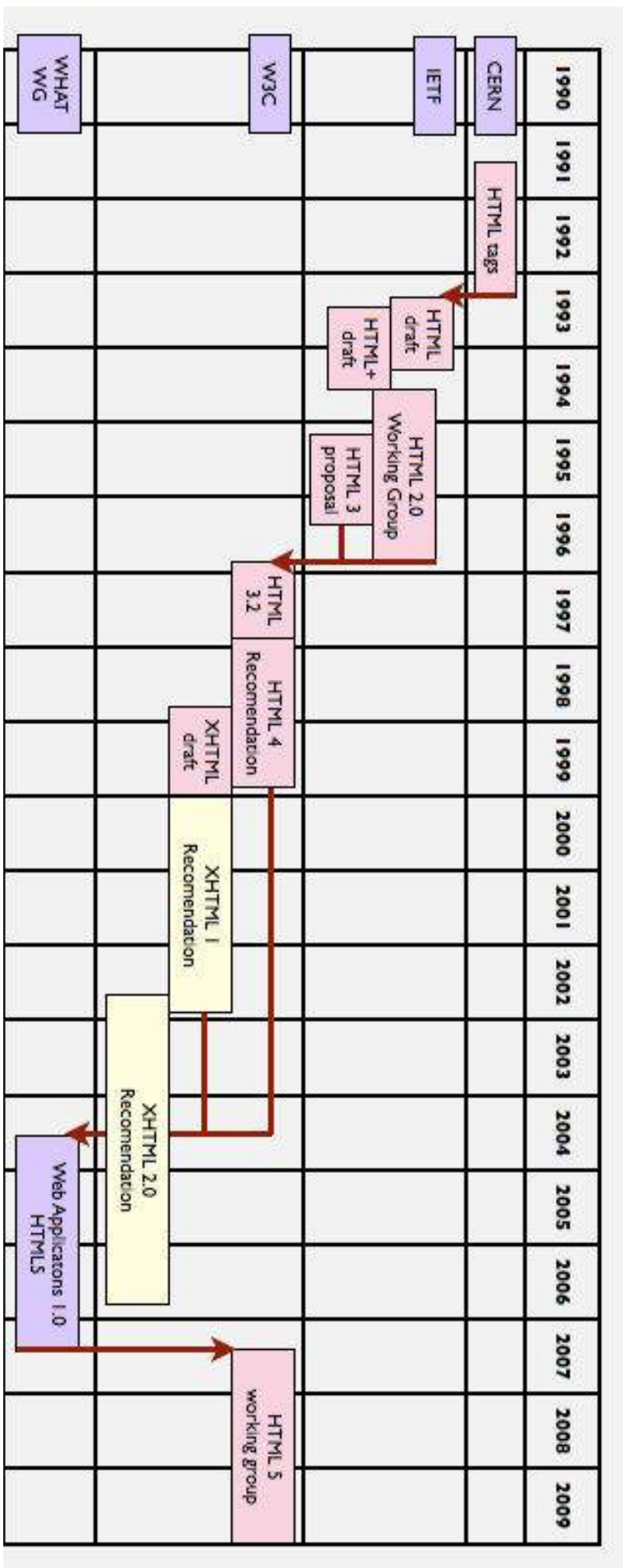


Figure 3.5: HTML History

All attributes must have explicit attribute values.

XML (and therefore XHTML) does not support attribute minimization, the SGML practice in which certain attributes can be reduced to just the attribute value. So, while in HTML you can write `checked` to indicate that a form button be checked when the form loads, in XHTML you need to explicitly write out `checked="checked"`.

Elements must be nested properly.

Although it has always been a rule in HTML that elements should be properly nested, in XHTML the rule is strictly enforced. Be sure that the closing tag of a contained element appears before the closing tag of the element that contains it. This will be more clear with an example. Be sure to do this:

<code><p>I can fly</p></code>
--

and not this:

<code><p>I can fly</p>.</code>

Follow additional nesting restrictions.

In HTML, there are some basic nesting restrictions (for example, don't put a `p` inside another `p` or put a block-level element in an inline element).

XHTML adds a few more nesting restrictions:

- **a** must not contain other **a** elements.
- **pre** must not contain the **img**, **object**, **big**, **small**, **sub**, or **sup** elements.
- **button** must not contain the **input**, **select**, **textarea**, **label**, **button**, **form**, **fieldset**, **iframe** or **isindex** elements.
- **label** must not contain other **label** elements.
- **form** must not contain other **form** elements.

3.4.3 HTML in three flavors

The authors of the HTML 4.01 Recommendation had a dilemma on their hands. They had a vision of how HTML should work, given the standardization of style sheets and scripting languages. Unfortunately, the practical fact was, by that time, millions of web pages had been written in legacy HTML.

They could not enforce a radical change to the standard overnight. To address this problem, they created several versions of HTML.

Transitional

The "Transitional" version includes most of the presentational extensions to HTML that were in common use. This option was made available to ease web authors as well as browser and tools

developers out of their old habits. The presentational elements (like **center**) and attributes (like **bgcolor** and **align**) were marked as deprecated, indicating that they would be removed from future versions of HTML. The HTML 4.01 Recommendation urges the web community to avoid using deprecated elements and attributes. Now that CSS is well supported by virtually all browsers, that is finally easy to do.

Strict

At the same time, the W3C created a “Strict” version of HTML 4.01 that omits all of the deprecated elements and gets HTML into the state they ultimately wanted it to be in.

Frameset

The third version of HTML is the “Frameset” version, which describes the content of framed documents. Frames make it possible to divide the browser into multiple windows, each displaying a different HTML document. Frames are constructed with a frameset document that defines the frame structure and the content of each frame. Frameset documents are fundamentally different from other HTML documents because they use the frameset element instead of body. This technicality earned them their own HTML specification.

3.4.4 Declaring the Document Type

If you’ve made the effort to write a standards compliant document, it makes sense that you’d want it displayed in the browser’s Standards mode. To do this, simply tell the browser what type of (X)HTML document it is by identifying the (X)HTML version (that is, the DTD (Document Type Definition)) that you followed in a document type (DOCTYPE) declaration at the beginning of the document.

- The `<!DOCTYPE>` declaration must be the very first thing in your HTML document, before the `<html>` tag.
- The `<!DOCTYPE>` declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in.
- In HTML 4.01, the `<!DOCTYPE>` declaration refers to a DTD (Document Type Definition), because HTML 4.01 was based on SGML. The DTD specifies the rules for the markup language, so that the browsers render the content correctly.
- HTML5 is not based on SGML, and therefore does not require a reference to a DTD.

Tip: Always add the `<!DOCTYPE>` declaration to your HTML documents, so that the browser knows what type of document to expect.

This is an example of a DOCTYPE declaration that indicates the document has been written according to the rules of the HTML 4.01 Strict DTD. DOCTYPE declarations must appear before the opening <html> tag.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/HTML4.01/strict.dtd">
<html>
```

...document continues...

The most commonly used DTDs are also presented here as a reference.

HTML DTDs

Strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
```

XHTML 1.0 DTDs

Strict

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

Transitional

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

Frameset

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

3.4.5 Character Encoding

Because the Web is worldwide, there are hundreds of written languages with a staggering number of unique character shapes that may need to be displayed on a web page. These include not only the various alphabets (Western, Hebrew, Arabic, and so on), but also **ideographs** (characters that indicate a whole word or concept) for languages such as Chinese, Japanese, and Korean.

To display an HTML page correctly, the browser must know what character-set to use.

The character-set for the early world wide web was ASCII. ASCII supports the numbers from 0–9, the uppercase and lowercase English alphabet, and some special characters.

Since many countries use characters which are not a part of ASCII, the default character-set for modern browsers is ISO-8859-1.

If a web page uses a different character-set than ISO-8859-1, it should be specified in the <meta> tag.

ISO Character Sets

It is the International Standards Organization (ISO) that defines the standard character-sets for different alphabets/languages.

The different character-sets being used around the world are listed below:

Character set	Description	Covers
ISO-8859-1	Latin alphabet part 1	North America, Western Europe, Latin America, the Caribbean, Canada, Africa
ISO-8859-2	Latin alphabet part 2	Eastern Europe
ISO-8859-3	Latin alphabet part 3	SE Europe, Esperanto, miscellaneous others
ISO-8859-4	Latin alphabet part 4	Scandinavia/Baltics (and others not in ISO-8859-1)
ISO-8859-5	Latin/Cyrillic part 5	The languages that are using a Cyrillic alphabet such as Bulgarian, Belarusian, Russian and Macedonian
ISO-8859-6	Latin/Arabic part 6	The languages that are using the Arabic alphabet
ISO-8859-7	Latin/Greek part 7	The modern Greek language as well as mathematical symbols derived from the Greek
ISO-8859-8	Latin/Hebrew part 8	The languages that are using the Hebrew alphabet
ISO-8859-9	Latin 5 part 9	The Turkish language. Same as ISO-8859-1 except Turkish characters replace Icelandic ones
ISO-8859-10	Latin 6 Lappish, Nordic, Eskimo	The Nordic languages

ISO-8859-15	Latin 9 (aka Latin 0)	Similar to ISO 8859-1 but replaces some less common symbols with the euro sign and some other missing characters
ISO-2022-JP	Latin/Japanese part 1	The Japanese language
ISO-2022-JP-2	Latin/Japanese part 2	The Japanese language
ISO-2022-KR	Latin/Korean part 1	The Korean language

The Unicode Standard

Because the character-sets listed above are limited in size, and are not compatible in multilingual environments, the Unicode Consortium developed the Unicode Standard.

The Unicode Standard covers all the characters, punctuations, and symbols in the world.

Unicode enables processing, storage and interchange of text data no matter what the platform, no matter what the program, no matter what the language.

The Unicode Consortium

The Unicode Consortium develops the Unicode Standard. Their goal is to replace the existing character-sets with its standard Unicode Transformation Format (UTF).

The Unicode Standard has become a success and is implemented in XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML, etc. The Unicode standard is also supported in many operating systems and all modern browsers.

The Unicode Consortium cooperates with the leading standards development organizations, like ISO, W3C, and ECMA.

Unicode can be implemented by different character-sets. The most commonly used encodings are UTF-8 and UTF-16:

Character-set	Description
UTF-8	A character in UTF8 can be from 1 to 4 bytes long. UTF-8 can represent any character in the Unicode standard. UTF-8 is backwards compatible with ASCII. UTF-8 is the preferred encoding for e-mail and web pages
UTF-16	16-bit Unicode Transformation Format is a variable-length character encoding for Unicode, capable of encoding the entire Unicode repertoire. UTF-16 is used in major operating systems and environments, like Microsoft Windows 2000/XP/2003/Vista/CE and the Java and .NET byte code environments

Tip: The first 256 characters of Unicode character-sets correspond to the 256 characters of ISO-8859-1.

Tip: All HTML 4 processors already support UTF-8, and all XHTML and XML processors support UTF-8 and UTF-16!

Specifying the character encoding

In HTML 4.01 and XHTML 1.0 documents, character encoding is indicated using a **meta** element. The **meta** element is an empty element that provides information about the document, such as its creation date, author, copyright information, and, as we'll focus on in this section, the character encoding and the type of file.

The **meta** element goes in the **head** of the document, as shown in this XHTML example (note the trailing slash in the empty **meta** element):

```
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Sample document</title>
</head>
```

The **http-equiv** attribute identifies that this **meta** element is providing information about the content type of the document.

The **content** attribute provides the details of the content type in a two-part value. The first part says that this is an HTML text file (in technical terms, it identifies its **media type** as **text/html**).

Finally, we get to the second part that specifies the character encoding for this document as **utf-8**.

For another look, here is a **meta** element for an (X)HTML document that uses the Latin-1 character encoding.

```
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
```

And for Arabic you can also use

```
<meta http-equiv='Content-Type' content='text/html; charset=windows-1256'/>
```

1- Information Architecture

Information architecture (IA) refers to the structure or organization of your website. It describes the ways in which the different pages of your site relate to one another and ensures information is organized in a consistent and predictable way on each page. It involves steps such as assessing existing and needed content, organizing the pages, providing clues to help use the site efficiently, and developing navigational structure.

We can define the information architect as: 1) the individual who organizes the patterns inherent in data, making the complex clear; 2) a person who creates the structure or map of information which allows others to find their personal paths to knowledge; 3) the emerging 21st century professional occupation addressing the needs of the age focused upon clarity, human understanding and the science of the organization of information. Structure and organization are about building rooms.

1.2. Defining Information Architecture.

Let's start by clarifying what we mean by information architecture:

1. The structural design of shared information environments.
2. The synthesis of organization, labeling, search, and navigation systems within digital, physical, and cross-channel ecosystems.
3. The art and science of shaping information products and experiences to support usability, findability, and understanding.
4. An emerging discipline and community of practice focused on bringing principles of design and architecture to the digital landscape.

2- Website Usability:

Some factors can influence web usability such as language and localization. World Wide Web was originally created for English content, and the accepted standards of web usability that followed were also targeted towards English websites.

Usability means that people who use a system can do so quickly and easily to accomplish their own tasks. Even though a system may perform well by other measures, without satisfactory usability a system is not successful because it does not meet the users' needs.

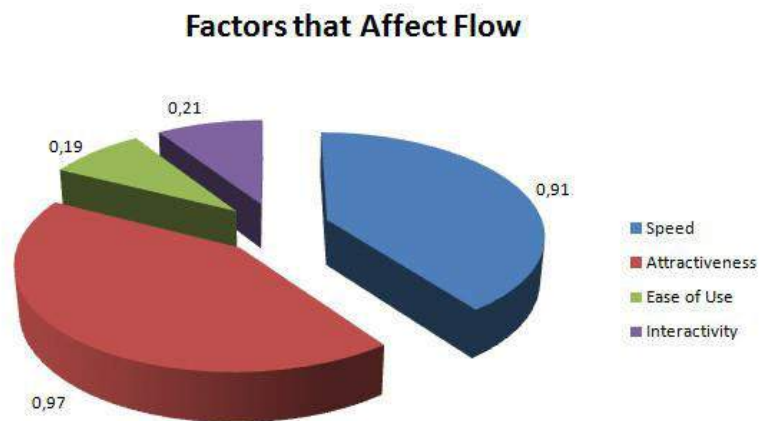
Web usability has also been defined as the degree of ease with which users can complete various tasks using a website interface with which they are unfamiliar.

Table 2. 1. Variety of expert-review methods for website evaluation.

Expert-review method	Explanation
Heuristic evaluation	Evaluators critique the design by conforming with a list of consisting usability elements.
Guidelines review	Evaluator follows organizational checklists or other well known guidelines. This may take a long time for evaluation.
Consistency inspection	The purpose of a consistency inspection is to check the extent of the consistency of the website elements such as color, fonts, output forms, consistency of terminology etc.
Cognitive walkthrough	Evaluators stimulate users navigating a website to conduct certain tasks to identify errors in website's interface.
Formal usability inspection	The reviewer holds a formal meeting with a website design team to present the interface and to discuss its merits and weaknesses. This type requires a great deal of time for preparation.

Table 2.1 Categories of Website Usability Testing

No.	Category	Description
1.	Ease of use	Usability, accessibility, navigability, and logical structure
2.	Responsiveness	Accessibility of service, e-mail service, reply to customer, contact information, and intuitive online help
3.	Fulfillment	Order process, accuracy of service promise, billing accuracy, online booking process, and confirmation, on-time delivery
4.	Security/Privacy	Information protection, online purchase security, and privacy statement
5.	Personalization	Individualized attention, customization of offerings and information
6.	Visual appearance	Attract attention, convey image, and aesthetics
7.	Information quality	Variety, scope, currency, conciseness, accuracy, authority, reliability, and uniqueness
8.	Trust	Brand recognition, consistency, intentions, and credibility
9.	Interactivity	Interactive features and communication (FAQs, guest books, chat)
10.	Advertising/persuasion	Marketing, promotional content, suggested products, recommendation, and incentives
11.	Playfulness	Enjoyment, fun, pleasure, and flow
12.	Technology integration	New technology and integration



Source: Skadberg & Kimmel 2004.

Fig 3. Factors that affect flow in websites.

Web page optimization streamlines your content to maximize display speed. Fast display speed is the key to success with your website . It increases profits, decreases costs, and improves customer satisfaction (not to mention search engine rankings, accessibility, and maintainability). To maximize web page display speed, you can employ the following 10 techniques: minimize HTTP requests, resize and optimize images, optimize multimedia, convert JavaScript behavior to code Cascading Style Sheets (CSS), use server-side sniffing, optimize JavaScript for execution speed and file size, convert table layout to code Cascading Style Sheets (CSS) layout, replace inline style with code Cascading Style Sheets (CSS) rules, minimize initial display time, and load JavaScript wisely.

2.1. Usability engineering

Usability engineers are experts at testing and evaluating how systems work. For information systems, they measure such criteria as how long it takes users to learn how to use a system, how long it takes them to find information in a system, and how many errors they make along the way. Of all the disciplines we list, usability engineering is probably the most scientific in its view of users and the quality of their experiences with information systems. Keep in mind that usability engineers concentrate on measuring a system's performance, not in designing or redesigning a system. (Of course, measurements of a site's performance can greatly determine how redesign should proceed.)

Organizing Information

1. Organizing Web Sites and Intranets

The organization of information in web sites and intranets is a major factor in determining success, and yet many web development teams lack the understanding necessary to do the job well. Our goal in this chapter is to provide a foundation for tackling even the most challenging information organization projects. Organization systems are composed of *organization schemes* and *organization structures*. **An organization scheme defines the shared characteristics of content items and influences the logical grouping of those items. An organization structure defines the types of relationships between content items and groups.**

Before diving in, it's important to understand information organization in the context of web site development.

Organization is closely related to navigation, labeling, and indexing. The hierarchical organization structures of web sites often play the part of primary navigation system. The labels of categories play a significant role in defining the contents of those categories. Manual indexing is ultimately a tool for organizing content items into groups at a very detailed level. Despite these closely knit relationships, it is both possible and useful to isolate the design of organization systems, which will form the foundation for navigation and labeling systems. By focusing solely on the logical grouping of information, you avoid the distractions of implementation details and design a better web site.

2. Personalization and Customization

Personalization involves serving up information to the user based upon a model of the behavior, needs, or preferences of that individual. In contrast, *customization* involves giving the user direct control over some combination of presentation, navigation, and content options. **In short, with personalization, we guess what the user wants, and with customization, the user tells us what he wants.** Both personalization and customization can be used to refine or supplement existing navigation systems. Unfortunately, however, both have been hyped by consultants and software vendors as the solution to all navigation problems. The reality is that personalization and customization:

- Typically play important but limited roles
- Require a solid foundation of structure and organization
- Are really difficult to do well
- Can make it more difficult to collect metrics and analyze user behavior

Amazon is the most cited example of successful personalization, and some of the things it's done are truly valuable. It's nice that Amazon remembers our names, and it's great that it remembers our address and credit card information. It's when Amazon starts trying to recommend products based on past purchases that the system breaks down (see [Figure 4](#)). In this example, Jorge already owns two of the top five recommended books, but the system doesn't know this because he purchased them elsewhere and (obviously) not as Kindle books. And this ignorance is not the exception but the rule. Because we don't have time to teach our systems, or because we prefer to maintain our privacy, we often don't share enough information to drive effective personalization. In addition, in many cases, it's really hard to guess what people will want to do or learn or buy tomorrow. As they say in the financial world, past performance is no guarantee of future results. In short, personalization works really well in limited contexts, but fails when you try to expand it to drive the entire user experience.



Figure 4. Amazon's personalized recommendations

Customization introduces a similar set of promises and perils. The idea of giving users control and thereby alleviating some of the pressures on design is obviously very compelling. And customization can sometimes deliver great value. For example, Gmail allows the user to set the visibility and order of labels—a critical element in the structuring of the user's mail in the system by dragging and dropping them within a global navigation structure (Figure 5).

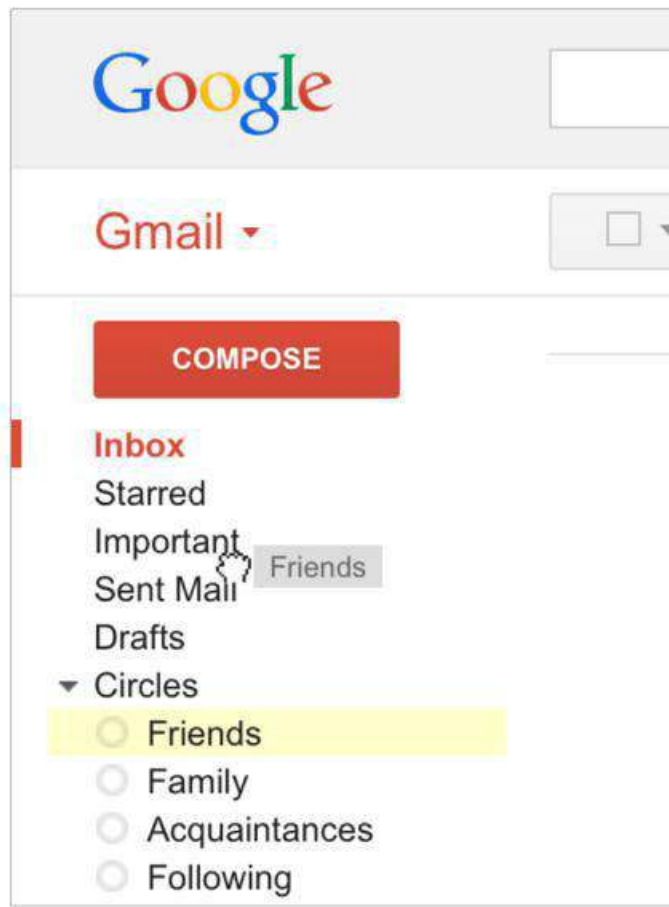


Figure 5. Customization in Gmail

The problem with customization is that most people don't want to spend much (if any) time customizing, and will do this work only on a small handful of sites that are most important to them. Because corporate intranets have a captive audience of repeat visitors, customization has a much better chance of being used there than it does on most public websites. However, there's another problem. Even users themselves don't always know what they will want to know or do tomorrow. Customization works great for tracking the sports scores of your favourite baseball team or monitoring the value of stocks you own, but not so well when it comes to broader news and research needs. One day you want to know the results of the French elections; the next day you want to know when dogs were first domesticated. Do you really know what you might need next month?

3. Visualization

Since the advent of the Web, people have struggled to create useful tools that enable users to navigate in a more visual way. First came the metaphor-driven attempts to display online museums, libraries, shopping malls, and other websites as physical places. Then came the dynamic, fly-through “sitemaps” that tried to show relationships between pages on a website. Both looked very cool and stretched our imaginations. But neither proved to be very useful. Visualization has proven most useful when the user must select among a result set of elements that she knows by their looks, as in the case of shopping for physical goods (Figure 6).

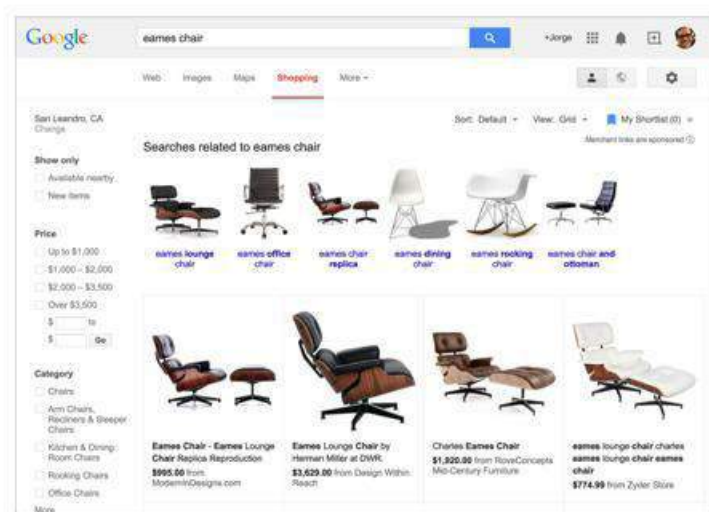


Figure 6. Google Shopping's visual search results

4. Search Algorithms

Search engines find information in many ways. In fact, there are about 40 different retrieval algorithms alone, most of which have been around for decades. We're not going to cover them all here; if you'd like to learn more, read any of the standard texts on information retrieval. We bring up the topic because it's important to realize that a retrieval algorithm is essentially a tool, and just like other tools, specific algorithms help solve specific problems. And as retrieval algorithms are at the heart of search engines, it's important to note that there is absolutely no single search engine that will meet all of your users' information needs. Remember that fact the next time you hear a search engine vendor claim that his product's brand-new proprietary algorithm is the solution to all information problems.

5. Pattern-Matching Algorithms

Most retrieval algorithms employ pattern matching; that is, they compare the user's query with an index of, typically, the full texts of your system's documents, looking for the same string of text. When a matching string is found, the source document is added to the retrieval set. So a user types the textual query "electric guitar," and documents that include the text string "electric guitar" are retrieved. It all sounds quite simple. But this matching process can work in many different ways to produce different results.

6. Advanced Search

In the past, many websites provided advanced search interfaces as crutches for under-featured or poorly configured search systems. In stark contrast to the search box, advanced search interfaces allow much more manipulation of the search system and are typically used by two types of users: advanced searchers (librarians, lawyers, doctoral students, medical researchers), and frustrated searchers who need to revise their initial search (often users who found that the search box didn't meet their needs). As search engines have improved, advanced search interfaces are increasingly focused on serving the former. While they are less common today, advanced search interfaces provide flexibility and power to users who understand the structure of the information they are looking for. For example, the U.S. Congress website allows knowledgeable users to configure extremely sophisticated searches using Boolean operators (Figure 7).



Figure 7. Congress.gov allows advanced users to build complex searches using Boolean operators

If your system could benefit from advanced search, a good rule of thumb is to expose your search engine's various heavy-duty search functions on the advanced page for those few users who want to have a go at them, but design your search system with the goal of making it unnecessary for the vast majority of searchers to ever need to go to the advanced search page.

7. Metadata

When it comes to definitions, metadata is a slippery fish. Describing it as “data about data” isn’t very helpful. The following excerpt from Wikipedia takes us a little further:

Metadata (metacontent) is defined as the data providing information about one or more aspects of the data, such as:

- *Means of creation of the data*
- *Purpose of the data*
- *Time and date of creation*
- *Creator or author of the data*
- *Location on a computer network where the data was created*
- *Standards used*

For example, a digital image may include metadata that describe how large the picture is, the color depth, the image resolution, when the image was created, and other data. A text document’s metadata may contain information about how long the document is, who the author is, when the document was written, and a short summary of the document.

Metadata tags are used to describe documents, pages, images, software, video and audio files, and other content objects for the purposes of improved navigation and retrieval. The keywords attribute of the HTML <meta> tag used by many websites provides a simple.

4

CSS for Presentation

4.1 Cascading Style Sheets Orientation

Cascading Style Sheets (CSS) is the W3C standard for defining the presentation of documents written in HTML, XHTML, and, in fact, any XML language. Presentation, again, refers to the way the document is displayed or delivered to the user, whether on a computer screen, a cell phone display, or read aloud by a screen reader.

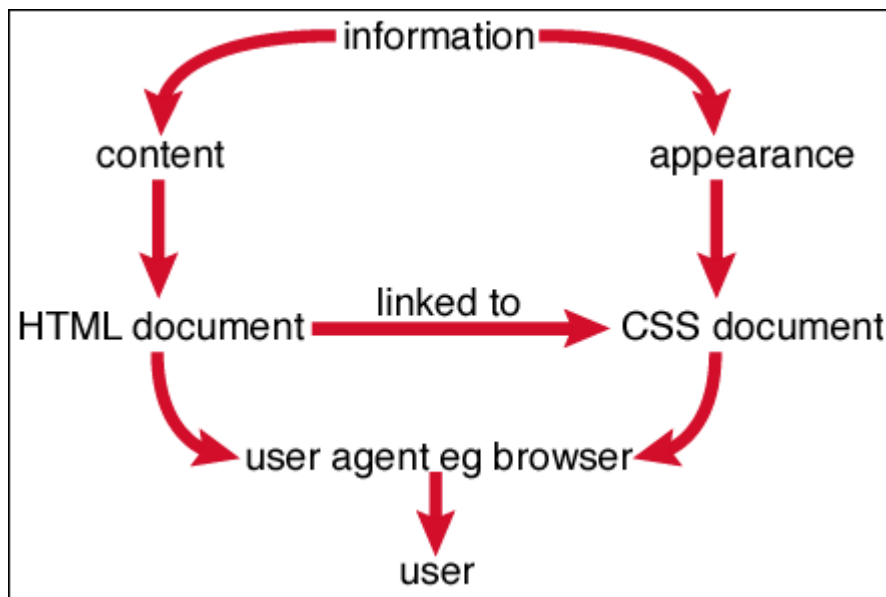


Figure 4.1: Separation between the content and the presentation

4.2 The Benefits of CSS

Here is a list of the benefits of using style sheets:

Better type and layout controls. Presentational (X)HTML never gets close to offering the kind of control over type, backgrounds, and layout that is possible with CSS.

Less work. You can change the appearance of an entire site by editing one style sheet. Making small tweaks and even entire site redesigns with style sheets is much easier than when presentation instructions are mixed in with the markup.

Potentially smaller documents and faster downloads. Old school practices of using redundant font elements and nested tables make for bloated documents. Not only that, you can apply a single style sheet document to all the pages in a site for further byte savings.

More accessible sites. When all matters of presentation are handled by CSS, you can mark up your content meaningfully, making it more accessible for nonvisual or mobile devices.

Reliable browser support. Nearly every browser in current use supports all of CSS Level 1 and the majority of CSS Level 2.

4.3 CSS Levels

Cascading Style Sheets does not have versions in the traditional sense; instead it has *levels*. Each level of CSS builds on the previous, refining definitions and adding features. The feature set of each higher level is a superset of any lower level, and the behavior allowed for a given feature in a higher level is a subset of that allowed in the lower levels. A user agent conforming to a higher level of CSS is thus also conformant to all lower levels.

CSS1:

- The first official version of CSS (the CSS Level 1 Recommendation) was officially released in 1996.
- It used to include properties for adding font, color, and spacing instructions to page elements.
- Unfortunately, lack of dependable browser support prevented the widespread adoption of CSS for several years.

CSS2:

- CSS Level 2 (CSS2) was released in 1998.
- It most notably added properties for positioning that allowed CSS to be used for page layout.
- It also introduced styles for other media types (such as print, handheld, and aural) and more sophisticated methods for selecting elements for styling.

CSS2.1:

- CSS Level 2, Revision 1 (CSS2.1) makes some minor adjustments to CSS2.
- Fortunately, most current browsers support the majority of the CSS1, CSS2, and CSS2.1 specifications.

CSS3:

- The CSS3 specification is still under development by W3C.
- It adds support for vertical text, improved handling of tables and international languages
- Better integration with other XML technologies.
- Provides little privileges like multiple background images in a single element and a larger list of color names.
- Many of the new CSS3 properties have been implemented in modern browsers.

4.4 How Style Sheets Work

1. Start with a document that has been marked up in HTML or XHTML.
2. Write style rules for how you'd like certain elements to look.
3. Attach the style rules to the document. When the browser displays the document, it follows your rules for rendering elements.

4.5 Writing the rules

A style sheet is made up of one or more style instructions (called rules) that describe how an element or group of elements should be displayed. The first step in learning CSS is to get familiar with the parts of a rule. As you'll see, they are fairly intuitive to follow. Each rule selects an element and declares how it should look.

The following example contains two rules. The first makes all the h1 elements in the document green; the second specifies that the paragraphs should be in a small, sans-serif font.

```
h1 { color: green; }  
p { font-size: small; font-family: sans-serif; }
```

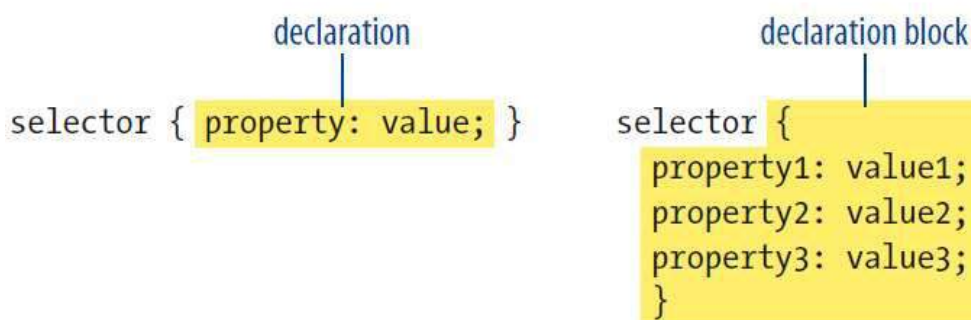


Figure 4.2: The parts of a style sheet rule.

4.5.1 Providing Measurement Values

When providing measurement values, the unit must immediately follow the number, like this:

```
{ margin: 2em; }
```



Adding a space before the unit will cause the property not to work.

```
INCORRECT: { margin: 2 em; }
```



It is acceptable to omit the unit of measurement for zero values:

```
{ margin: 0; }
```

Note that: 1em is equal to the current font size. 2em means 2 times the size of the current font. E.g., if an element is displayed with a font of 12 pt, then '2em' is 24 pt. The 'em' is a very useful unit in CSS, since it can adapt automatically to the font that the reader uses

4.6 Attaching the styles to the document

There are three ways to style information that can be applied to an (X)HTML document.

External style sheets. An external style sheet is a separate, text-only document that contains a number of style rules. It must be named with the .css suffix. The .css document is then linked to or imported into one or more (X)HTML documents. In this way, all the files in a web site may share the same style sheet. This is the most powerful and preferred method for attaching style sheets to content.

Embedded style sheets. It is placed in a document using the style element and its rules apply only to that document. The style element must be placed in the **head** of the document and it must contain a type attribute that identifies the content of the style element as “text/css”. This example also includes a comment.

```
<head>
  <title>Required document title here</title>
  <style type="text/css">
    /* style rules go here */
  </style>
</head>
```

The style element may also include the media attribute used to target specific media such as screen, print, or handheld devices.

Inline styles. You can apply properties and values to a single element using the style attribute in the element itself, as shown here:

```
<h1 style="color: red">Introduction</h1>
```

To add multiple properties, just separate them with semicolons, like this:

```
<h1 style="color: red; margin-top: 2em">Introduction</h1>
```

Inline styles apply only to the particular element in which they appear.

Inline styles should be avoided, unless it is absolutely necessary to override styles from an embedded or external style sheet.

4.7 The Concepts

There are a few big ideas that you need to get your head around to be comfortable with how Cascading Style Sheets behave.

Inheritance

(X)HTML elements pass down certain style properties to the elements they contain. For example, when we styled the `p` elements in a small, sans-serif font, the `em` element in the second paragraph became small and sans-serif as well, even though we didn't write a rule for it specifically. That is because it inherited the styles from the paragraph it is in.

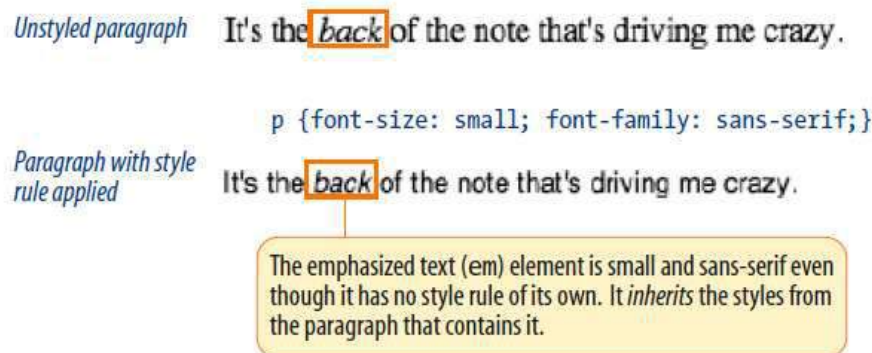


Figure 4.3: The `em` element inherits styles that were applied to the paragraph

Document structure

This is where an understanding of your document's structure comes in. (X)HTML documents have an implicit structure or hierarchy.

For example, an (X)HTML page has an `html` root element that contains a `head` and a `body`, and the `body` contains heading and paragraph elements. A few of the paragraphs, in turn, contain inline elements like images (`img`) and emphasized text (`em`). You can visualize the structure as an upside-down tree, branching out from the root, as shown in Figure 4.5.

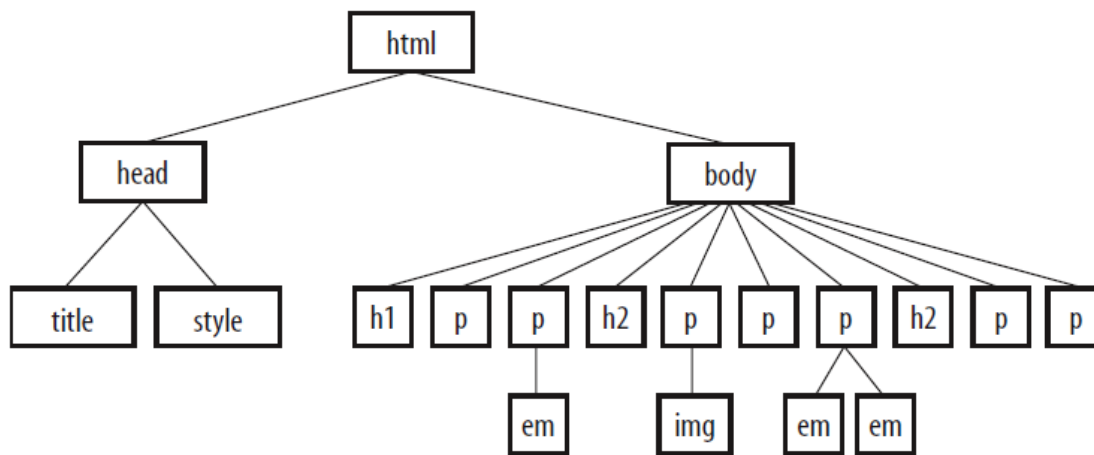


Figure 4.4: The document tree structure of the sample document, twenties.html.

Parents and children

The document tree becomes a family tree when it comes to referring to the relationship between elements.

- All the elements contained within a given element are said to be its **descendants**. For example, all of the h1, h2, p, em, and img elements in the document in Figure 4.5 are descendants of the body element.
- An element that is directly contained within another element (with no intervening hierarchical levels), is said to be the **child** of that element.
- Conversely, the containing element is the **parent**. For example, the em element is the child of the p element, and the p element is its parent.
- All of the elements higher than a particular element in the hierarchy are its **ancestors**.
- Two elements with the same parent are **siblings**.

Pass it on

When you write a font-related style rule using the p element as a selector, the rule applies to all of the paragraphs in the document as well as the inline text elements they contain. We've seen the evidence of the em element inheriting the style properties applied to its parent (p) back in Figure 4.3. Figure 4.5 demonstrates what's happening in terms of the document structure diagram. Note that the img element is excluded because font-related properties do not apply to images

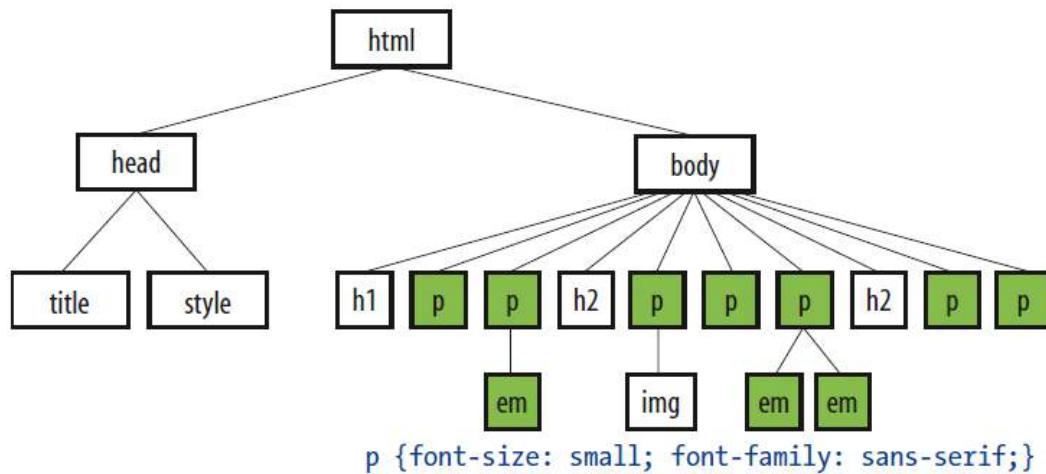


Figure 4.5: Certain properties applied to the `p` element are inherited by their children.

It's important to note that some style sheet properties inherit and others do not.

- In general, properties related to the styling of text—font size, color, style, etc.—are passed down.
- Properties such as borders, margins, backgrounds, and so on that affect the boxed area around the element tend not to be passed down.

You can use inheritance to your advantage when writing style sheets. For example, if you want all text elements to be rendered in the Verdana font face, you could write separate style rules for every element in the document and set the font-face to Verdana. A better way would be to write a single style rule that applies the font-face property to the body element, and let all the text elements contained in the body inherit that style (Figure 4.6).

Note that: Any property applied to a specific element will override the inherited values for that property.

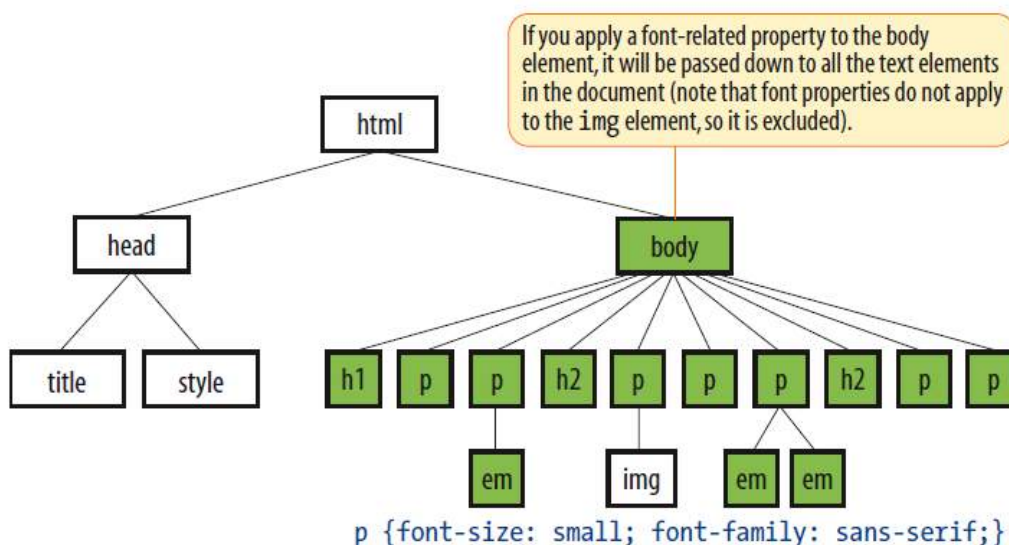


Figure 4.6: All the elements in the document inherit certain properties applied to the body element.

4.8 Conflicting styles: the cascade

CSS allows you to apply several style sheets to the same document, which means there are bound to be conflicts. For example, what should the browser do if a document's imported style sheet says that **h1** elements should be **red**, but its embedded style sheet has a rule that makes **h1**s **purple**?

The folks who wrote the style sheet specification anticipated this problem and devised a hierarchical system that assigns different weights to the various sources of style information. The cascade refers to what happens when several sources of style information vie for control of the elements on a page: **style information is passed down until it is overridden by a style command with more weight**. For example:

- If you don't apply any style information to a web page, it will be rendered according to the browser's internal style sheet (we've been calling this the default rendering).
- If the web page's author provides a style sheet for the document, that has more weight and overrides the browser's styles.

As we know, there are three ways to attach style information to the source document, and they have a cascading order as well. Generally speaking, the closer the style sheet is to the content, the more weight it is given.

- Embedded style sheets that appear right in the document in the style element have more weight than external style sheets.
- Inline styles have more weight than embedded style sheets because you can't get any closer to the content than a style right in the element's opening tag.

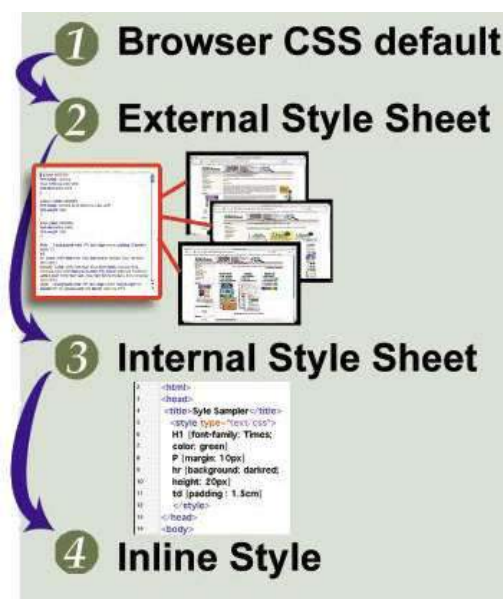


Figure 4.7: How styles Cascade

4.8.1 Specificity

Once the applicable style sheet has been chosen, there may still be conflicts; therefore, the cascade continues at the rule level. When two rules in a single style sheet conflict, the type of selector is used to determine the winner. The more specific the selector, the more weight it is given to override conflicting declarations. We will discuss this in more details later.

4.8.2 Rule order

Finally, if there are conflicts within style rules of identical weight, whichever one comes last in the list “wins.” Take these three rules, for example:

```
<style type="text/css">
p { color: red; }
p { color: blue; }
p { color: green; }
</style>
```

In this scenario, paragraph text will be green because the last rule in the style sheet, that is, the one closest to the content in the document, overrides the earlier ones.

4.8.3 Assigning Importance

If you want a rule not to be overridden by a subsequent conflicting rule, include the **!important** indicator just after the property value and before the semicolon for that rule. For example, to make paragraph text blue always, use the following rule:

```
p {color: blue !important;}
```

Even if the browser encounters an inline style later in the document (which should override a document-wide style sheet), like this one:

```
<p style="color: red">
```

that paragraph will still be blue, because the rule with the **!important** indicator cannot be overridden by other styles in the author’s style sheet.

4.9 The box model

The easiest way to think of the box model is that browsers see every element on the page (both block and inline) as being contained in a little rectangular box. You can apply properties such as borders, margins, padding, and backgrounds to these boxes, and even reposition them on the page.

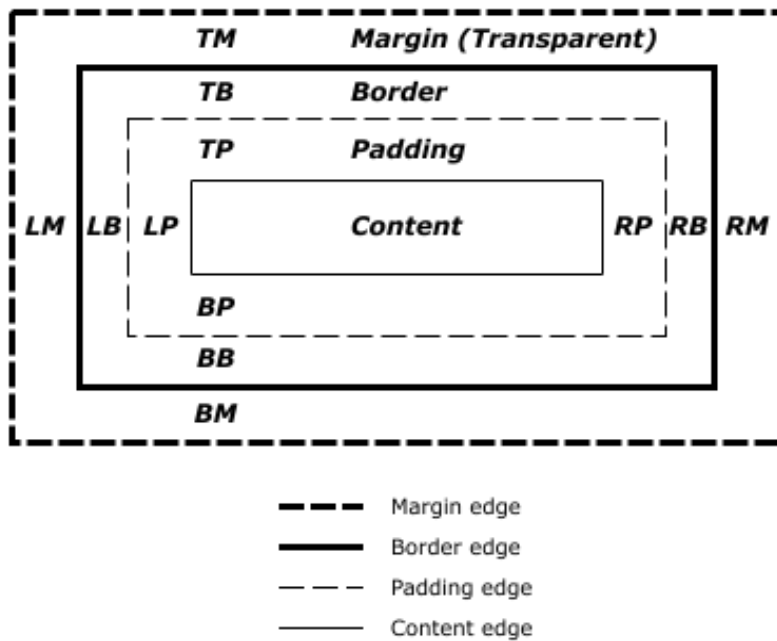


Figure 4.8: The box model

To see the elements roughly the way the browser sees them, I've written style rules that add borders around every content element in our sample article.

```
h1 { border: 1px solid blue; }
h2 { border: 1px solid blue; }
p { border: 1px solid blue; }
em { border: 1px solid blue; }
img { border: 1px solid blue; }
```

Figure 4.9 shows the results. The borders reveal the shape of each block element box. There are boxes around the inline elements (`em` and `img`), as well.

Notice that the block element boxes expand to fill the available width of the browser window, which is the nature of block elements in the normal document flow. Inline boxes encompass just the characters or image they contain.

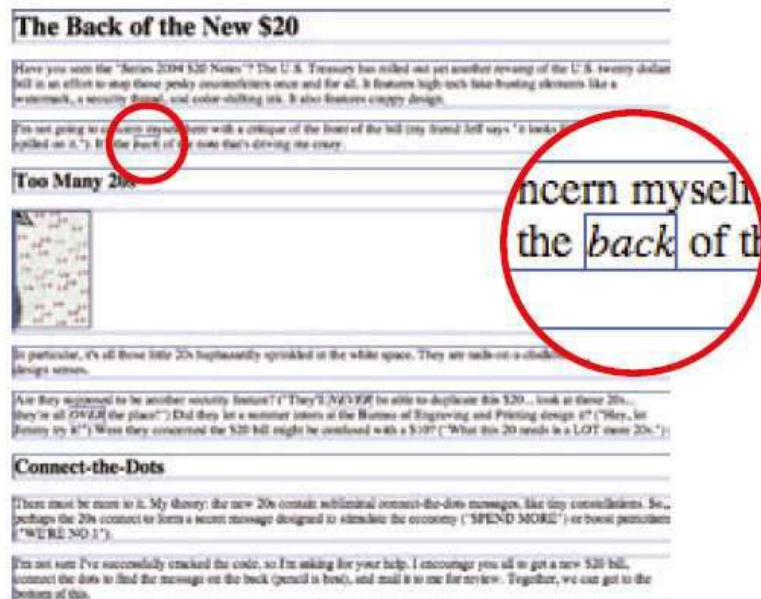


Figure 4.9: Rules around all the elements reveal their element boxes

There is a difference in how width is interpreted between the W3C and Internet Explorer box models, Figure 4.10 explains that.

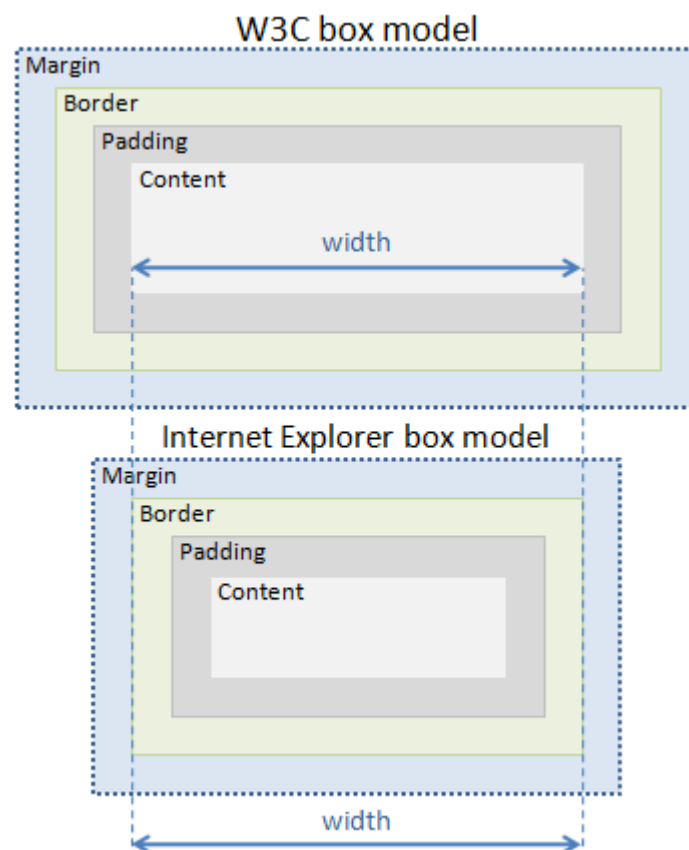


Figure 4.10: W3C and Internet Explorer box models

5

Page Layout with CSS

5.1 Page Layout Strategies

There is no way of knowing exactly how wide, skinny, tall, or short a user's browser window will be. Users may set their browsers to fill the monitor at one of the standard resolutions or have them set to some other comfortable dimension. The precise size of any given web page is an unknown.

In addition, there is no way of knowing how large your text will be. You may prefer small tidy text, but a portion of your users will make their text larger, possibly much larger, to make it comfortable to read. Changing text size is likely to have an impact on the layout of your page.

Over time, three general page layout approaches have emerged to deal with these inescapable facts.

- **Liquid pages** resize along with the browser window.
- **Fixed pages** put the content in a page area that stays a specific pixel width regardless of the browser's dimensions.
- **Elastic pages** have areas that get larger or smaller when the text is resized.

5.1.1 Liquid page design

Liquid page layouts (also called *fluid* layouts)

- Follow the default behavior of the normal flow.
- In other words, the page area and/or columns within the page are allowed to get wider or narrower to fill the available space in the browser window.
- There is no attempt to control the width of the content or line breaks—the text is permitted to reflow as required.

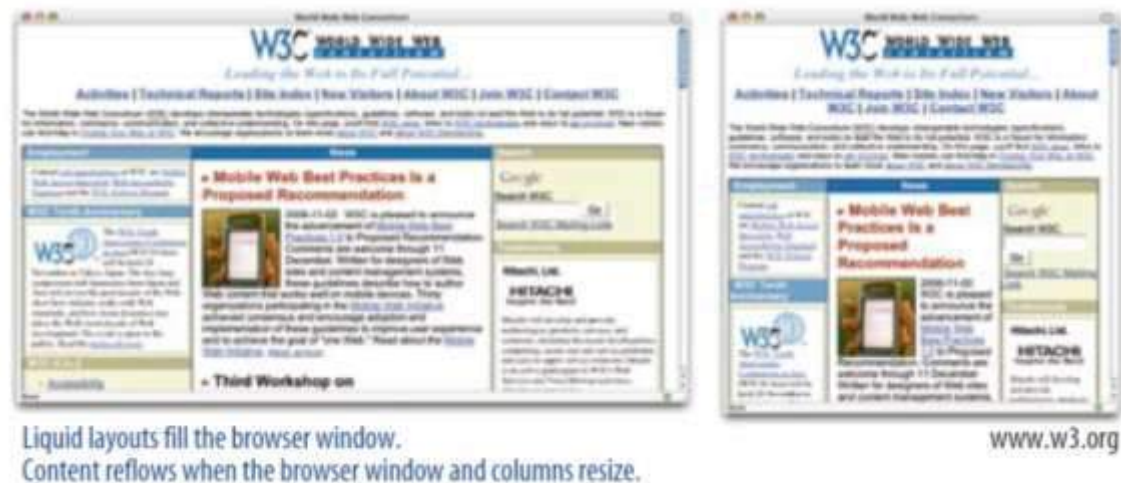


Figure 5.1: Example of a liquid layout

<i>Advantages</i>	<i>Disadvantages</i>
<ul style="list-style-type: none"> • You don't have to design for a specific monitor resolution. • You avoid potentially awkward empty space because the text fills the window. • Liquid pages keep with the spirit and nature of the medium. 	<ul style="list-style-type: none"> • On large monitors, line lengths can get very long and uncomfortable to read. • They are less predictable. Elements may be too spread out or too cramped at extreme browser dimensions.

Optimal Line Length

Line length is a measure of the number of words or characters in a line text. The rule of thumb is that the optimal line length is **10-12 words** or between **60-75 characters**.

When line lengths grow too long, the text becomes more difficult to read. Not only is it hard to focus long enough to get to the end of a long line, it is also requires extra effort to find the beginning of the next line again.

Now, consider the following html page together with the styles listed in the coming examples:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>Untitled 1</title>
<link href="css.css" rel="stylesheet" type="text/css" />
```



```

</head>
<body>
<div id="wrapper">
<div id="main">

<h1>Main Column</h1>
<p>Line length is a measure of the number of words or characters in a
line text. The rule of thumb is that the optimal line length is 10-12 words
or between 60-75 characters.
When line lengths grow too long, the text becomes more difficult to read.
Not only is it hard to focus long enough to get to the end of a long line, it
is also requires extra effort to find the beginning of the next line again.
</p>
<p>Line length is a measure of the number of words or characters in a
line text. The rule of thumb is that the optimal line length is 10-12 words
or between 60-75 characters.
When line lengths grow too long, the text becomes more difficult to read.
Not only is it hard to focus long enough to get to the end of a long line, it
is also requires extra effort to find the beginning of the next line again.
</p>
</div>

<div id="extras">
<h1>Extras Column</h1>
<p>Line length is a measure of the number of words or characters in a
line text. The rule of thumb is that the optimal line length is 10-12 words
or between 60-75 characters.
When line lengths grow too long, the text becomes more difficult to read.
Not only is it hard to focus long enough to get to the end of a long line, it
is also requires extra effort to find the beginning of the next line again.
</p>
</div>
</div>
</body>
</html>

```

Example 1:

```

div#main {
    width: 70%;
    margin-right: 5%;
    float: left;
    background: yellow;}
div#extras {
    width: 25%;
    float: left;
    background: orange;}

```

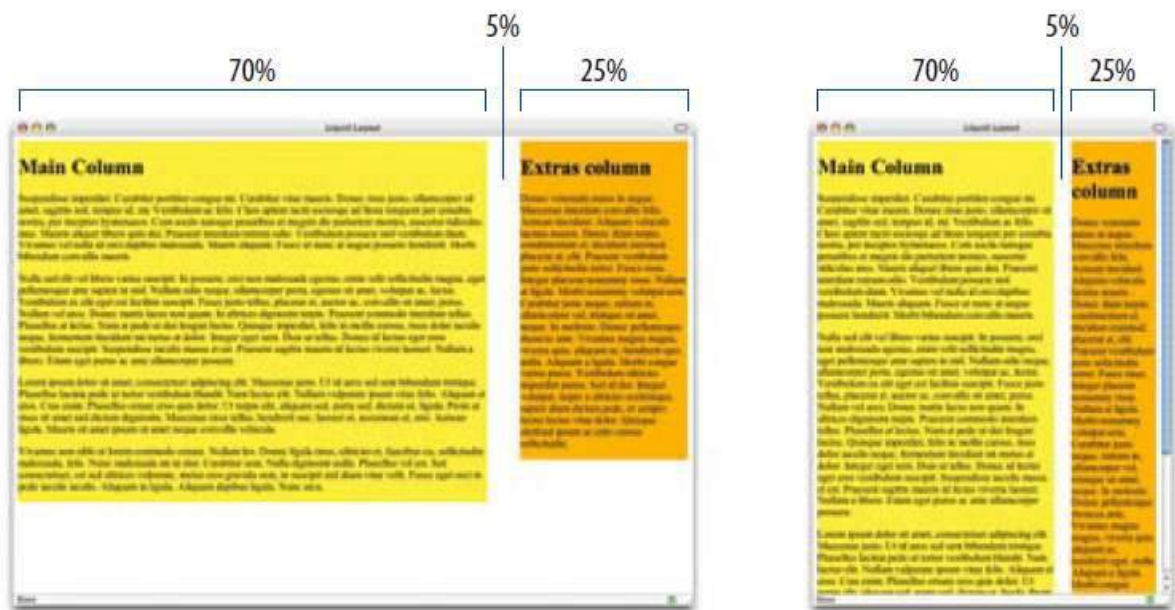


Figure 5.2: Liquid layout using percentage values

Example 2:

```
div#main { width: auto;
           position: absolute;
           top: 0;
           left: 225px;
           background: yellow; }
div#extras { width: 200px;
            position: absolute;
            top: 0;
            left: 0;
            background: orange; }
```

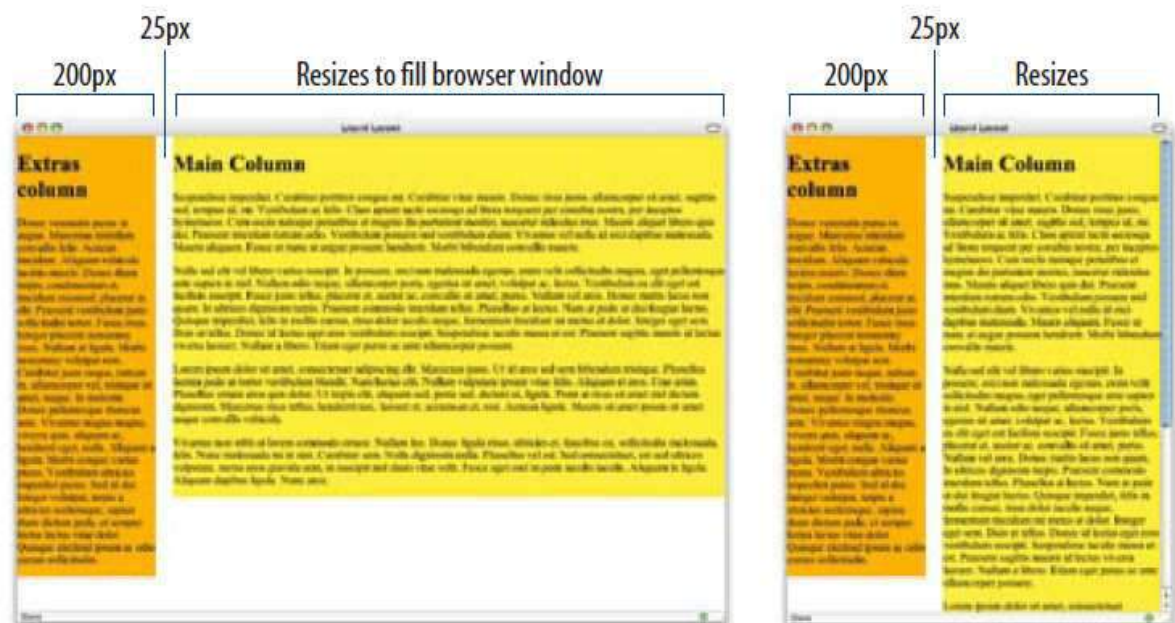


Figure 5.3: Liquid layout combining fixed-width and auto sized columns

5.1.2 Fixed Layouts

- Fixed-width layouts use a specified pixel width as determined by the designer.
- You need to decide a couple of things.
 - **First**, you need to pick a page width, usually based on common monitor resolutions. Most fixed-width web pages such as 800×600 pixel browser window, although more and more sites are venturing into a (roughly) 1000 pixel page width.
 - **Second**, you need to decide where the fixed-width layout should be positioned in the browser window. By default, it stays on the left edge of the browser, with the extra space to the right of it. Some designers opt to center the page, splitting the extra space over left and right margins.

Example 3:

```
#wrapper { width: 750px;
           position: absolute;
           margin-left: auto;
           margin-right: auto;
           border: 1px solid black;
           padding: 0px;}
#extras { position: absolute;
          top: 0px;
          left: 0px;
          width: 200px;
          background: orange; }
#main { margin-left: 225px;
        background-color: yellow;}
```

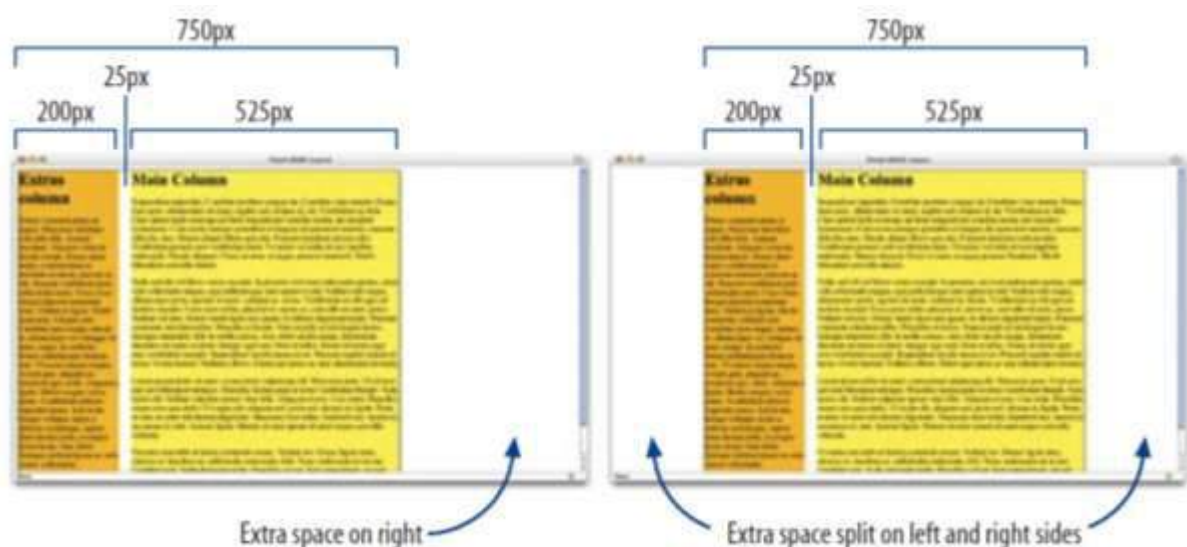


Figure 5.4: Examples of fixed layouts (left-aligned and centered)

One of the main concerns with using fixed-width layouts is that if the user's browser window is not as wide as the page, the content on the right edge of the page will not be visible. Although it is possible to scroll horizontally, it may not always be clear that there is more content there in the first place.

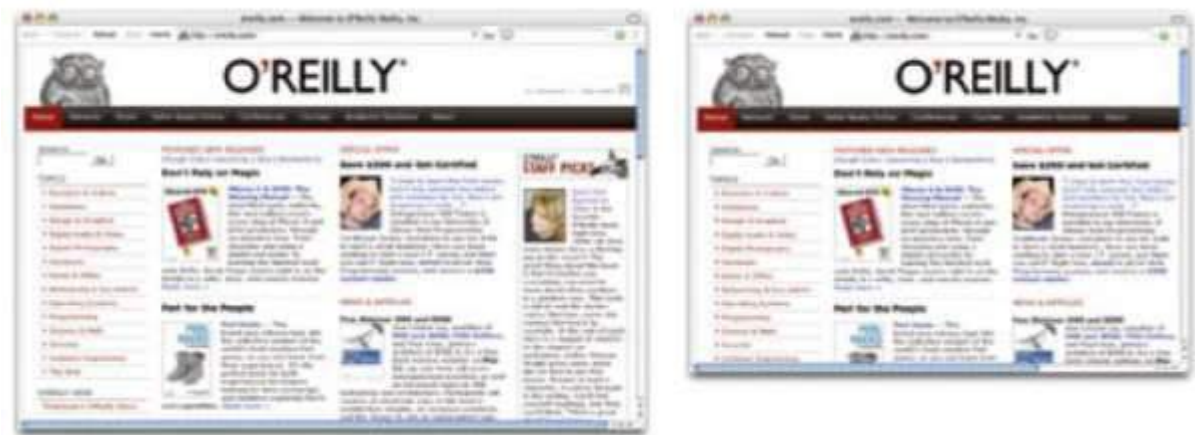


Figure 5.5: Users may miss out on content on the right edge of a fixed layout if the browser is not as wide as the page area

<i>Advantages</i>	<i>Disadvantages</i>
<ul style="list-style-type: none">• The layout is predictable.• It offers better control over line length.• Trends come and go; however, it is worth noting that many of the most well-known web designers use fixed-width designs.	<ul style="list-style-type: none">• Content on the right edge will be hidden if the browser window is smaller than the page.• Text elements still reflow if the user resizes the font size, so it doesn't guarantee the layout will stay exactly the same.• Line lengths may grow awkwardly short at very large text sizes.

5.1.3 Elastic Layouts

- A third layout approach is growing in popularity because it marries resizable text with predictable line lengths.
- *Elastic* (also called *jello*) layouts expand or contract with the size of the text. If the user makes the text larger, the box that contains it expands proportionally. Likewise, if the user likes her text size very small, the containing box shrinks to fit.
- The result is that line lengths (in terms of words or characters per line) stay the same regardless of the text size. This is an advantage over liquid layouts where line lengths can get too long, and fixed layouts where very large text may result in awkwardly few characters per line.



Figure 5.6: A classic example of elastic page layout

<i>Advantages</i>	<i>Disadvantages</i>
<ul style="list-style-type: none"> • Provide a consistent layout experience while allowing flexibility in text size. • Tighter control over line lengths than liquid and fixed layouts. 	<ul style="list-style-type: none"> • Images don't lend themselves to rescaling along with the text and the rest of the layout. • The width of the layout might exceed the width of the browser window at largest text sizes. This can be prevented with proper planning and/or the max-width property (unsupported in IE6 and earlier).

The key to elastic layouts is the **em**, a unit of measurement that is based on size of the text. For example, for an element with 12-pixel text, an em is 12 pixels. We learned that it is always preferable to specify font size in ems because it allows the text to be resized with the zoom feature in all modern browsers (remember that IE6 and earlier do not zoom text sized in pixels). In elastic layouts, the dimensions of containing elements are specified in ems as well. That is how the widths can respond to the text size.

For example, if the text size is set to 76% (equal to about 12 pixels on most browsers), and the page is set to 40 em, the resulting page width would be 480 pixels (40 em x 12px/em).

If the user resizes the text up to 24 pixels, the page grows to 960 pixels. Note that this is getting close to the available canvas space in browsers on 1024-pixel wide monitors. If the page were set much more than 40 ems wide, there would be the risk of the right edge extending beyond the browser window at extremely large text sizes.

5.2 Page Layout Templates

In this section we will discuss how to create two- and three columns layouts using CSS and **absolutely no tables**. This section provides templates and techniques for the following:

- Multicolumn layouts using floats (two- and three-column)
- Multicolumn layouts using positioning (two- and three-column, with and without footer)
- A centered fixed width page

5.2.1 Multicolumn Layouts Using Floats

The most popular way to create a column is to float an element to one side and let the remaining content wrap around it. A wide margin is used to keep the area around the floated column clear.

- The **advantage** of floats is that it is easier to start page elements such as a footer below the columned area of the page.
- The **drawback** of float-based layouts is that they are dependent on the order in which the elements appear in the source.

5.2.1.1 Two-column with footer

Method: FLOAT Layout: LIQUID

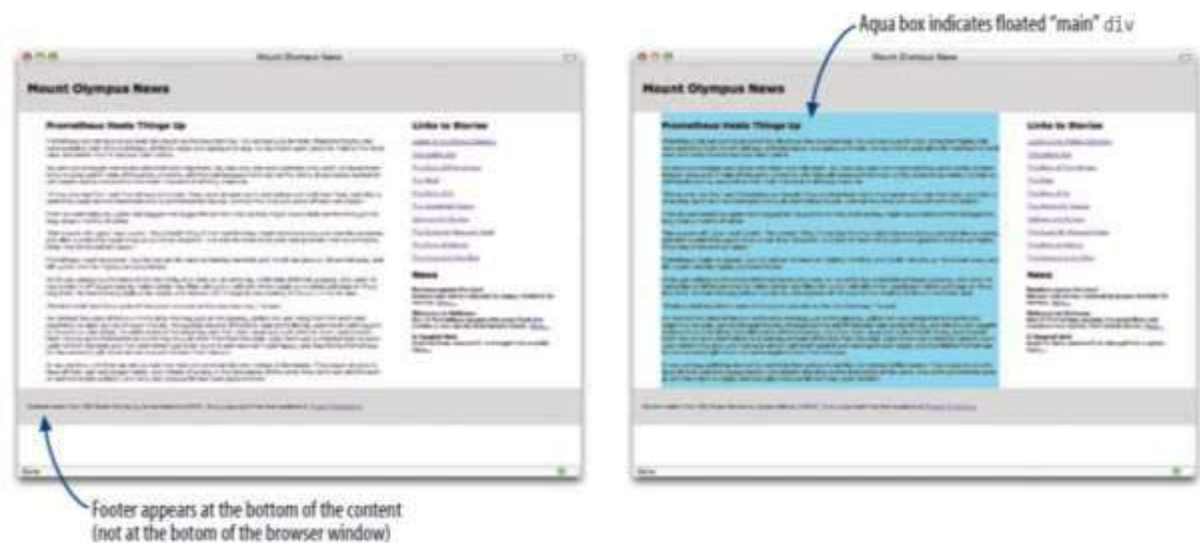


Figure 5.7: Two-column layout with footer

The Markup



```
<div id="header">
    Masthead and headline
</div>
<div id="main">
    Main article...
</div>
<div id="extras">
    List of links and news
```

```
</div>
<div id="footer">
    Copyright information
</div>
```

Markup Notes:

As you see, the source document has been divided into four **divs**, one each for the header, content, extras, and footer.

- A** The main content appears before the extras in the source document so that it is accessed first by users with non-graphical browsers. That means that we can't float the "extras" **div** because it will not float above the preceding block element to the top of the page. Instead, the main content **div** is floated and the following text (the "extras" **div**) wraps around it.

The Style Sheet

```
#header {
    background: #CCC;
    padding: 15px; }

#main {
    background-color: aqua;
    float: left; /* floats the whole main article to the left */
    width: 60%;
    margin-right: 5%; /* adds space between columns */
    margin-left: 5%; }

#footer {
    clear: left; /* starts the footer below the floated
    content */
    padding: 15px;
    background: #CCC; }

#extras {
    margin-right: 5%; /* space on the right of the side
    column */

body {
    font-family: verdana, sans-serif;
    margin: 0; /* clears default browser margins */
    padding: 0; }

li {
    list-style: none;
    margin: 10px 0; }
```

Style Sheet Notes:

- B** The main content **div** is floated to the left and set to 60% of the page width. A margin is applied to the left and right sides of the floated "main" **div** to add space between columns.

- C** The “footer” **div** is cleared (with the **clear** property) so that it starts below the floated main content column.
- D** A margin is added on the right edge of the “extras” **div** to add space between it and the browser window.
- E** The margin and padding on the **body** element have been set to zero to clear the default browser settings. This allows the shaded header and footer areas to go right up to the edge of the browser window without any white gaps.

5.2.1.2 Three-column with footer

Method: **FLOAT** Layout: **LIQUID**

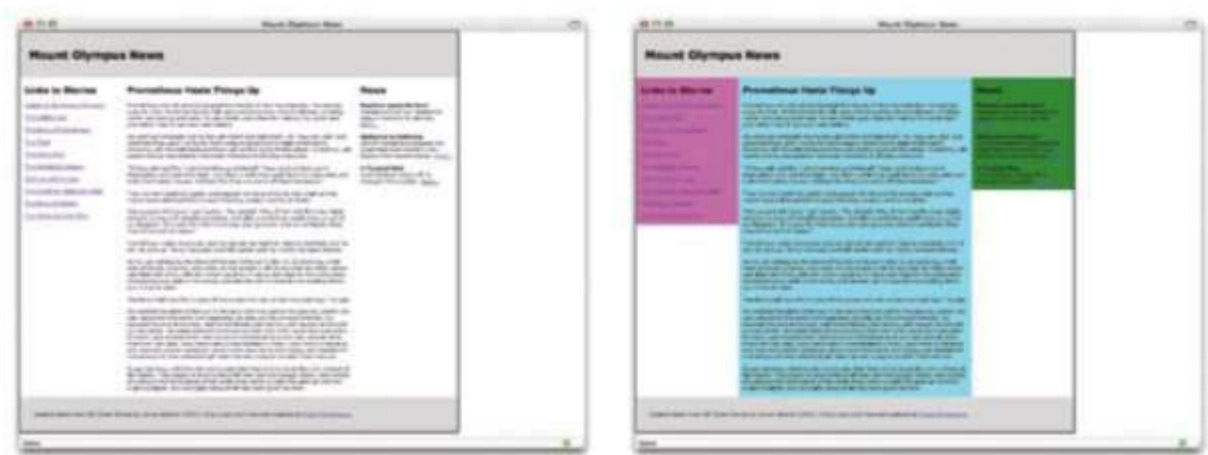


Figure 5.8: Three-column layout using floats

The Markup

A `<div id="container">`

`<div id="header">`
Masthead and headline
`</div>`

B `<div id="links">`
List of links
`</div>`

`<div id="main">`
Main article...
`</div>`

`<div id="news">`
News and announcements...
`</div>`

`<div id="footer">`
Copyright information
`</div>`

`</div>`

Markup Notes

- A** All of the content elements in the document have been placed in a “container” **div** to which the fixed-width measurement will be applied.
- B** Remember that with floating, the order that the elements appear in the source document is significant. To get the narrow sidebars on either side of the content, I needed to move the “links” **div** before the “content” **div** to keep the style sheet straightforward.

The Style Sheet

C	<pre>#container { width: 750px; border: solid 1px; }</pre>
D	<pre>#header { background: #CCC; padding: 15px; }</pre>
D	<pre>#links { background-color: fuchsia; float: left; width: 175px; }</pre>
E	<pre>#main { background-color: aqua; float: left; width: 400px; }</pre>
D	<pre>#news { background-color: green; float: left; width: 175px; }</pre>
E	<pre>#footer { clear: both; /* starts the footer below the floated content */ padding: 15px; background: #CCC; }</pre>
F	<pre>body { font-family: verdana, sans-serif; font-size: small; margin: 0; padding: 0; }</pre>
G	<pre>h2, ul, p { padding: 0px 8px; } /* adds space around content */ li { list-style: none; margin: 10px 0; }</pre>

Style Sheet Notes

- C** A border has been added to the container to reveal its edges in this demonstration, but it can easily be removed.
- D** The style sheet floats the “links” “main,” and “news” **divs** to the left. The result is that they accumulate against the left edge of the containing block, thus creating three columns.
- E** Because there are no padding, border, or margin settings for each floated element, the sum of their widths is equal to the width of the outer container.
- F** The **clear: both** property has been added to the footer to make sure it starts below all of the floated elements.
- G** Space within each content **div** is added by applying padding on the elements it contains (**h2**, **ul**, **p**, etc.).

5.2.2 Layouts Using Absolute Positioning

Absolute positioning can also be used to create a multicolumn page.

- The **advantage** is that the order of the source document is not as critical as it is in the float method, because element boxes can be positioned anywhere.
- The **disadvantage** is that you run the risk of elements overlapping and content being obscured. This makes it tricky to implement full-width elements below columns (such as the footer in the previous example), because it will get overlapped if a positioned column grows too long.

5.2.2.1 Two-column with narrow footer

Method: **POSITIONED**

Layout: **LIQUID**

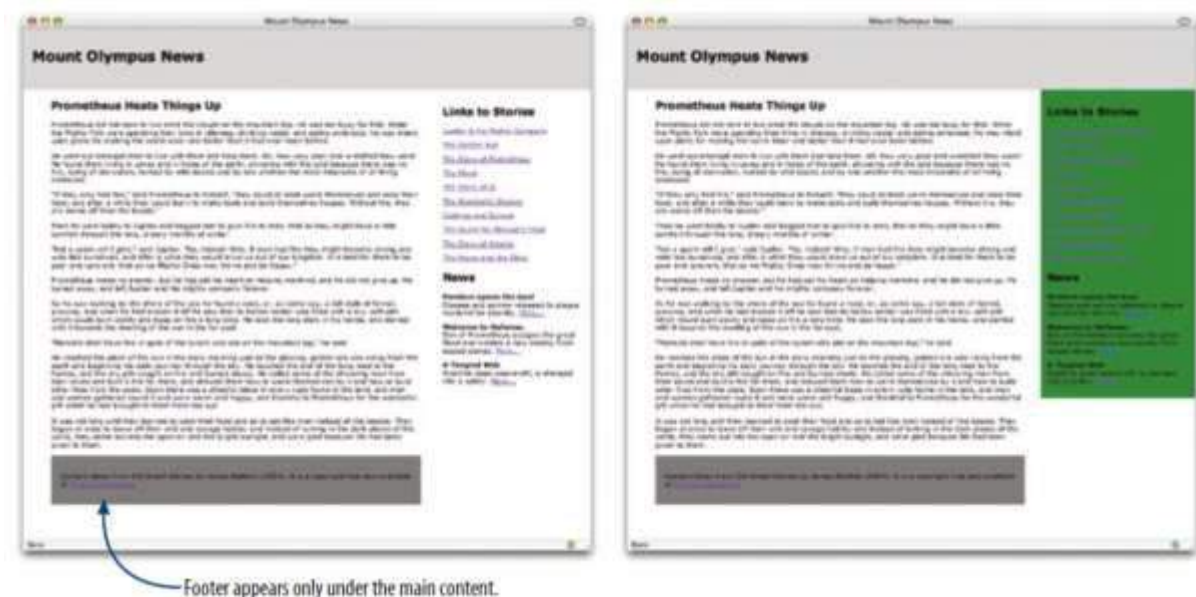


Figure 5.9: Two-column layout with absolute positioning

The Markup

```
<div id="header">
    Masthead and headline
</div>
<div id="main">
    Main article...
</div>
<div id="extras">
    List of links and news
</div>
<div id="footer">
    Copyright information
</div>
```

A

Markup Notes

A

This style sheet absolutely positions the “extras” **div** element against the right side of the page and 100 pixels down from the top to leave room for the header element.

The “main” content **div** is given a right margin wide enough to make a space for the newly positioned box.

The Style Sheet

B

```
#header {
    height: 70px;
    padding: 15px; /* height of header = 100 (15+ 70+ 15)
    */
    background: #CCC;}
```

C

```
#main {
    margin-right: 30%; /* makes room for the positioned
    sidebar */
    margin-left: 5%; }
```

D

```
#extras {
    position: absolute;
    top: 100px; /* places the extras div below the header
    */
    right: 0px; /* places it against right edge of the
    window */
    width: 25%;
    background: green;
    padding: 10px;} /* adds space within colored box */
```

E

```
#footer {
    margin-right: 30%; /* keeps the footer aligned with
    content */
    margin-left: 5%;
    padding: 15px;
    background: #666; }

body {
```

```
font-family: verdana, sans-serif;  
font-size: small;  
margin: 0;  
padding: 0;}  
ul { padding: 0px; }  
li {  
  
    list-style: none;  
    margin: 10px 0; }
```

Style Sheet Notes

- B** In this example, we know that the header is exactly 100 pixels tall (70 height plus 30 pixels of padding).
- C** The 30% right margin makes space for the column that is 25% of the page plus 5% space between the columns.
- D** The “extras” **div** is positioned absolutely 0 pixels from the right edge of the browser and exactly 100 pixels down from the top.
- E** The margins applied to the main content were also applied to the footer **div**. That is to prevent the footer from being overlapped by a long sidebar.

5.2.2.2 Three-column (narrow footer)

Method: POSITIONED

Layout: LIQUID

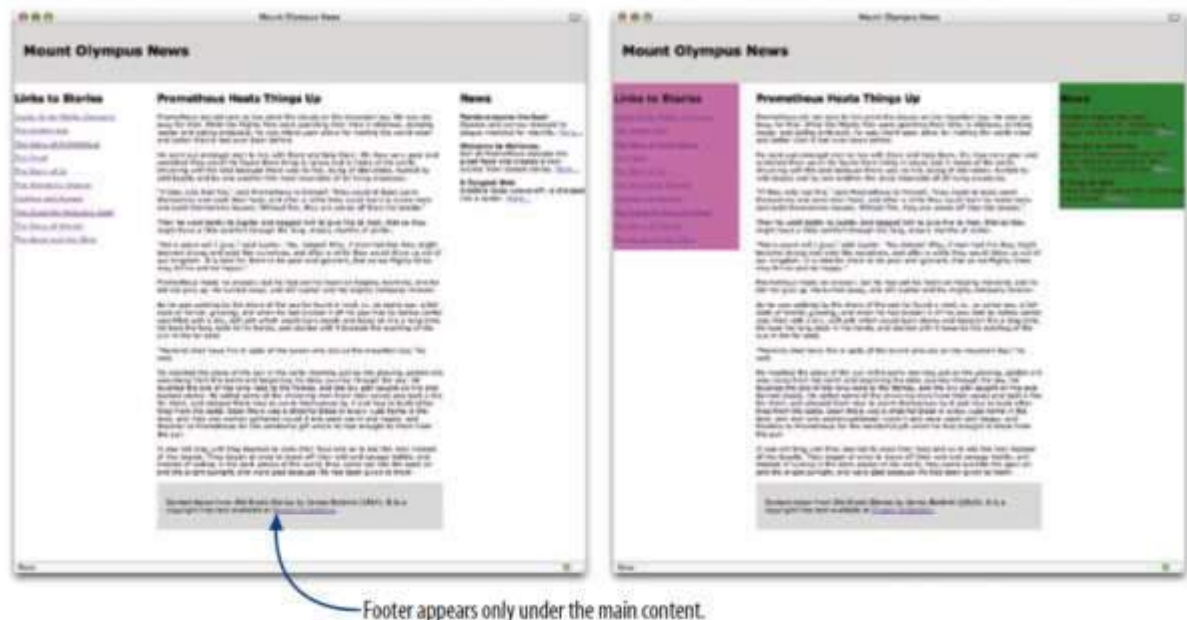


Figure 5.10: Positioning two sidebars in a three-column layout

The Markup

```
<div id="header">  
    Masthead and headline  
</div>  
<div id="main">
```

A

	<i>Main article...</i>
	</div>
B	<div id="links">
	<i>List of links</i>
	</div>
B	<div id="news">
	<i>News and announcements...</i>
	</div>
	<div id="footer">
	<i>Copyright information</i>
	</div>

Markup Notes

- A Because absolute positioning is not order-dependent, the main content **div** can appear in its preferable position first in the document source.
- B Only the “links” and “news” **div** elements are positioned in this layout.

The Style Sheet

	#header {
	height: 70px;
	padding: 15px; /* height of header = 100 (15+ 70+ 15)
	*/
	background: #CCC;
	}
	#main {
	margin-left: 25%; /* makes room for the left sidebar */
	margin-right: 25%; /* makes room for the right sidebar
	*/
	}
C	#links {
	position: absolute;
	top: 100px; /* places the sidebar below the header */
	left: 0px; /* places it against left edge of the window
	*/
D	width: 22%; /* less than main margins to add #
	between cols */
	background: fuchsia;
	}
C	#news {
	position: absolute;
	top: 100px; /* places the sidebar below the header */
	right: 0px; /* places it against right edge of the
	window */
D	width: 22%;

E

```
background: green;
}
#footer {
margin-right: 25%; /* keeps the footer aligned with
content */
margin-left: 25%;
padding: 15px;
background: #CCC;
}
```

Style Sheet Notes

The style sheet is essentially the same as that for the previous example, with the exception that margins have been applied to both sides of the “main” and “footer” **div** elements to make room for columns on both sides. The comments within the style sheet provide information on what key properties are doing.

C

The “links” and “news” **divs** are positioned against the left and right edges of the browser window (technically, it’s the initial-containing block), respectively.

D

The width of the positioned columns is narrower than the margins on the main content **div** to allow space between columns.

E

The footer gets the same margin treatment as the main content column to make sure the side columns do not overlap it.

5.2.3 Centering a Fixed-Width Page

In CSS, the proper way to center a fixed-width element is to specify a **width** for the **div** element that holds all the page’s contents, then set the left and right margins to **auto**. According to the CSS visual formatting model, this will have the net effect of centering the element in the initial containing block.

```
#container {
position: relative;
width: 750px;
margin-right: auto;
margin-left: auto;
}
```

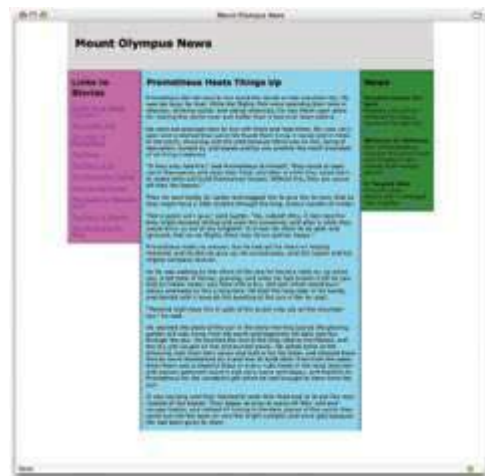


Figure 5.11: Centering a fixed-width page element

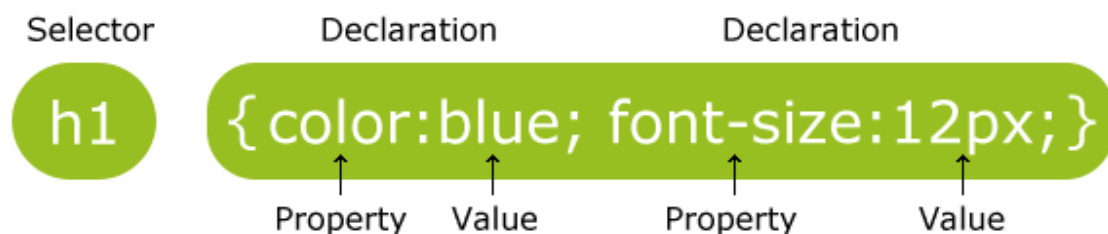
CSS Basic

1. What is CSS?

- CSS stands for Cascading Style Sheets
- Styles define **how to display** HTML elements
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**

2. CSS Syntax

A CSS rule has two main parts: a selector, and one or more declarations:



- The selector is normally the HTML element you want to style.
- Each declaration consists of a property and a value.
- The property is the style attribute you want to change. Each property has a value.

3. CSS Example–

CSS declarations always ends with a semicolon, and declaration groups are surrounded by curly brackets:

```
p {color:red;text-align:center;}
```

To make the CSS more readable, you can put one declaration on each line, like this:

```
p {  
color:red;  
text-align:center;  
}
```

4. CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers. A CSS comment begins with "/*", and ends with "*/", like this:

```
/*This is a comment*/
```

```
p
{
text-align:center;
/*This is another comment*/
color:black;
font-family:arial;
}
```

5. The id and class Selectors

In addition to setting a style for a HTML element, CSS allows you to specify your own selectors called "id" and "class".

5.1 The id Selector

The id selector is used to specify a style for a single, unique element. The id selector uses the id attribute of the HTML element, and is defined with a "#".

The style rule below will be applied to the element with id="para1":

```
#para1
{
text-align:center;
color:red;
}
```

i Do NOT start an ID name with a number! It will not work in Mozilla/Firefox.

5.2 The class Selector

The class selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements.

This allows you to set a particular style for any HTML elements with the same class.

The class selector uses the HTML class attribute, and is defined with a "."

In the example below, all HTML elements with class="center" will be center-aligned:

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

You can also specify that only specific HTML elements should be affected by a class.

In the example below, all p elements with class="center" will be center-aligned:

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


i Do NOT start a class name with a number! This is only supported in Internet Explorer

6. Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

6.1 External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a .css extension. An example of a style sheet file is shown below:

```
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
```

Do not leave spaces between the property value and the units! "margin-left:20 px" (instead of "margin-left:20px") will work in IE, but not in Firefox or Opera.

6.2 Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, by using the <style> tag, like this:

```
<head>
<style type="text/css">
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

6.3 Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly!

To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

6.4 Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

For example, an external style sheet has these properties for the h3 selector:

```
h3
{
color:red;
text-align:left;
font-size:8pt;
}
```

And an internal style sheet has these properties for the h3 selector:

```
h3
{
text-align:right;
font-size:20pt;
}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

```
color:red;
text-align:right;
font-size:20pt;
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

6.5 Multiple Styles Will Cascade into One

Styles can be specified:

- inside an HTML element
- inside the head section of an HTML page
- in an external CSS file

Tip: Even multiple external style sheets can be referenced inside a single HTML document.

6.6 Cascading order

What style will be used when there is more than one style specified for an HTML element?

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

1. Browser default
2. External style sheet
3. Internal style sheet (in the head section)
4. Inline style (inside an HTML element)

So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or in a browser (a default value).

Note: If the link to the external style sheet is placed after the internal style sheet in HTML <head>, the external style sheet will override the internal style sheet!

7. CSS Background

CSS background properties are used to define the background effects of an element.

CSS properties used for background effects:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

7.1 Background Color

The background-color property specifies the background color of an element.

The background color of a page is defined in the body selector:

```
body {background-color:#b0c4de;}
```

The background color can be specified by:

- name – a color name, like "red"
- RGB – an RGB value, like "rgb(255,0,0)"
- Hex – a hex value, like "#ff0000"

In the example below, the h1, p, and div elements have different background colors:

```
h1 {background-color:#6495ed;}  
p {background-color:#e0ffff;}  
div {background-color:#b0c4de;}
```

7.2 Background Image

The `background-image` property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element. The background image for a page can be set like this:

```
body {background-image:url('paper.gif');}
```

7.3 Background Image – Repeat Horizontally or Vertically

By default, the `background-image` property repeats an image both horizontally and vertically.

Some images should be repeated only horizontally or vertically, or they will look strange, like this:

```
body
{
background-image:url('gradient2.png');
}
```

If the image is repeated only horizontally (`repeat-x`), the background will look better:

```
body
{
background-image:url('gradient2.png');
background-repeat:repeat-x;
}
```

7.4 Background Image – Set position and no-repeat

When using a background image, use an image that does not disturb the text.

Showing the image only once is specified by the `background-repeat` property:

```
body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;
}
```

In the example above, the background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much.

The position of the image is specified by the `background-position` property:

```
body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;
background-position:right top;
}
```

7.5 Background – Shorthand property

As you can see from the examples above, there are many properties to consider when dealing with backgrounds.

To shorten the code, it is also possible to specify all the properties in one single property. This is called a shorthand property.

The shorthand property for background is simply "background":

```
body {background:#ffffff url('img_tree.png') no-repeat right top;}
```

When using the shorthand property the order of the property values are:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

It does not matter if one of the property values are missing, as long as the ones that are present are in this order.

8. CSS Text

8.1 Text Color

The color property is used to set the color of the text. The color can be specified by:

- name – a color name, like "red"
- RGB – an RGB value, like "rgb(255,0,0)"
- Hex – a hex value, like "#ff0000"

The default color for a page is defined in the body selector.

```
body {color:blue;}
h1 {color:#00ff00;}
h2 {color:rgb(255,0,0);}
```

For W3C compliant CSS: If you define the color property, you must also define the background-color property.

8.2 Text Alignment

The text-align property is used to set the horizontal alignment of a text.

Text can be centered, or aligned to the left or right, or justified.

When text-align is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

```
h1 {text-align:center;}  
p.date {text-align:right;}  
p.main {text-align:justify;}
```

8.3 Text Decoration

The text-decoration property is used to set or remove decorations from text.

The text-decoration property is mostly used to remove underlines from links for design purposes:

```
a {text-decoration:none;}
```

It can also be used to decorate text:

```
h1 {text-decoration:underline;}  
h2 {text-decoration:line-through;}  
h3 {text-decoration:underline;}  
h4 {text-decoration:blink;}
```

It is not recommended to underline text that is not a link, as this often confuses users.

8.4 Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

```
p.uppercase {text-transform:uppercase;}  
p.lowercase {text-transform:lowercase;}  
p.capitalize {text-transform:capitalize;}
```

8.5 Text Indentation

The text-indentation property is used to specify the indentation of the first line of a text.

```
p {text-indent:50px;}
```

9. CSS Font

CSS font properties define the font family, boldness, size, and the style of a text.

Difference between Serif and Sans-serif Fonts

On computer screens, sans-serif fonts are considered easier to read than serif fonts.



9.1 CSS Font Families

In CSS, there are two types of font family names:

- **generic family** – a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** – a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

9.2 Font Family

The font family of a text is set with the font-family property.

The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like font-family: "Times New Roman".

More than one font family is specified in a comma-separated list:

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

9.3 Font Style

The font-style property is mostly used to specify italic text. This property has three values:

- normal – The text is shown normally
- italic – The text is shown in italics

- oblique – The text is "leaning" (oblique is very similar to italic, but less supported)

```
p.normal {font-style:normal;}  
p.italic {font-style:italic;}  
p.oblique {font-style:oblique;}
```

9.4 Font Size

The font-size property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Always use the proper HTML tags, like <h1> – <h6> for headings and <p> for paragraphs.

The font-size value can be an absolute or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

9.4.1 Set Font Size With Pixels

Setting the text size with pixels, gives you full control over the text size:

```
h1 {font-size:40px;}  
h2 {font-size:30px;}  
p {font-size:14px;}
```

The example above allows Firefox, Chrome, and Safari to resize the text, **but not Internet Explorer**.

The text can be resized in all browsers using the zoom tool (however, this resizes the entire page, not just the text).

9.4.2 Set Font Size With Em

To avoid the resizing problem with Internet Explorer, many developers use em instead of pixels. The em size unit is recommended by the W3C.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula:
 $pixels/16=em$

```
h1 {font-size:2.5em;} /* 40px/16=2.5em */  
h2 {font-size:1.875em;} /* 30px/16=1.875em */  
p {font-size:0.875em;} /* 14px/16=0.875em */
```

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

Unfortunately, there is still a problem with IE. When resizing the text, it becomes larger than it should when made larger, and smaller than it should when made smaller.

9.4.3 Use a Combination of Percent and Em

The solution that works in all browsers is to set a default font-size in percent for the body element:

```
body {font-size:100%;}  
h1 {font-size:2.5em;}  
h2 {font-size:1.875em;}  
p {font-size:0.875em;}
```

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

10. CSS Links

Links can be styled in different ways.

10.1 Styling Links

Links can be style with any CSS property (e.g. color, font-family, background-color).

Special for links are that they can be styled differently depending on what state they are in.

The four links states are:

- a:link – a normal, unvisited link
- a:visited – a link the user has visited
- a:hover – a link when the user mouse over it
- a:active – a link the moment it is clicked

```
a:link {color:#FF0000;} /* unvisited link */  
a:visited {color:#00FF00;} /* visited link */  
a:hover {color:#FF00FF;} /* mouse over link */  
a:active {color:#0000FF;} /* selected link */
```

When setting the style for several link states, there are some order rules:

- `a:hover` MUST come after `a:link` and `a:visited`
- `a:active` MUST come after `a:hover`

10.2 Common Link Styles

In the example above the link changes color depending on what state it is in.

Let's go through some of the other common ways to style links:

10.2.1 Text Decoration

The `text-decoration` property is mostly used to remove underlines from links:

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}
```

10.2.2 Background Color

The `background-color` property specifies the background color for links:

```
a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

11.CSS Lists

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker

11.1 List

In HTML, there are two types of lists:

- unordered lists – the list items are marked with bullets
- ordered lists – the list items are marked with numbers or letters

With CSS, lists can be styled further, and images can be used as the list item marker.

11.2 Different List Item Markers

The type of list item marker is specified with the `list-style-type` property:

```
ul.a {list-style-type: circle;}
ul.b {list-style-type: square;}
ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
```

Some of the property values are for unordered lists, and some for ordered lists.

Values for Unordered Lists	Value Description
none	No marker
disc	Default. The marker is a filled circle
circle	The marker is a circle
square	The marker is a square

Values for Ordered Lists	Value Description
armenian	The marker is traditional Armenian numbering
decimal	The marker is a number
decimal-leading-zero	The marker is a number padded by initial zeros (01, 02, 03, etc.)
georgian	The marker is traditional Georgian numbering (an, ban, gan, etc.)
lower-alpha	The marker is lower-alpha (a, b, c, d, e, etc.)
lower-greek	The marker is lower-greek (alpha, beta, gamma, etc.)
lower-latin	The marker is lower-latin (a, b, c, d, e, etc.)
lower-roman	The marker is lower-roman (i, ii, iii, iv, v, etc.)
upper-alpha	The marker is upper-alpha (A, B, C, D, E, etc.)
upper-latin	The marker is upper-latin (A, B, C, D, E, etc.)
upper-roman	The marker is upper-roman (I, II, III, IV, V, etc.)

Note: No versions of Internet Explorer (including IE8) support the property values "decimal-leading-zero", "lower-greek", "lowerlatin", "upper-latin", "armenian", or "georgian".

11.3 An Image as The List Item Marker

To specify an image as the list item marker, use the `list-style-image` property:

```
ul
{
list-style-image: url('sqpurple.gif');
}
```

The example above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari.

If you want the image-marker to be placed equally in all browsers, a crossbrowser solution is explained below.

11.4 Crossbrowser Solution

The following example displays the image-marker equally in all browsers:

```
ul
{
list-style-type: none;
padding: 0px;
margin: 0px;
}
li
{
background-image: url(sqpurple.gif);
background-repeat: no-repeat;
background-position: 0px 5px;
padding-left: 14px;
}
```

Example explained:

- For `ul`:
 - o Set the `list-style-type` to `none` to remove the list item marker
 - o Set both `padding` and `margin` to `0px` (for cross-browser compatibility)
- For `li`:
 - o Set the URL of the image, and show it only once (`no-repeat`)
 - o Position the image where you want it (left `0px` and down `5px`)
 - o Position the text in the list with `padding-left`

11.5 List – Shorthand property

It is also possible to specify all the list properties in one, single property. This is called a shorthand property.

The shorthand property used for lists, is the list-style property:

```
ul
{
list-style: square url("sqpurple.gif");
}
```

When using the shorthand property, the order of the values are:

- list-style-type
- list-style-position
- list-style-image

It does not matter if one of the values above are missing, as long as the rest are in the specified order.

12. CSS Tables

The look of an HTML table can be greatly improved with CSS:

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Königlich Essen	Philip Cramer	Germany

12.1 Table Borders

To specify table borders in CSS, use the border property.

The example below specifies a black border for table, th, and td elements:

```
table, th, td
{
border: 1px solid black;
}
```

Notice that the table in the example above has double borders. This is because both the table, th, and td elements have separate borders.

To display a single border for the table, use the border-collapse property.

12.2 Collapse Borders

The border-collapse property sets whether the table borders are collapsed into a single border or separated:

```
table
{
border-collapse:collapse;
}
table,th, td
{
border: 1px solid black;
}
```

12.3 Table Width and Height

Width and height of a table is defined by the width and height properties.

The example below sets the width of the table to 100%, and the height of the th elements to 50px:

```
table
{
width:100%;
}
th
{
height:50px;
}
```

12.4 Table Text Alignment

The text in a table is aligned with the text-align and vertical-align properties.

The text-align property sets the horizontal alignment, like left, right, or center:

```
td
{
text-align:right;
}
```

The vertical-align property sets the vertical alignment, like top, bottom, or middle:

```
td
{
height:50px;
vertical-align:bottom;
}
```

12.5 Table Padding

To control the space between the border and content in a table, use the padding property on td and th elements:

```
td
{
padding:15px;
}
```

12.6 Table Color

The example below specifies the color of the borders, and the text and background color of th elements:

```
table, td, th
{
border:1px solid green;
}
th
{
background-color:green;
color:white;
}
```

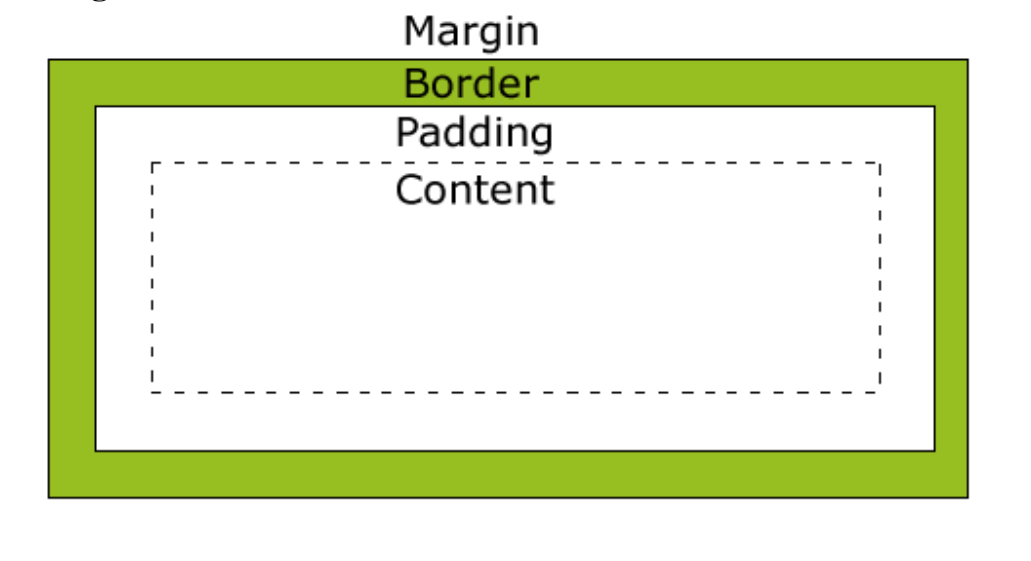
13. CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content.

The box model allows us to place a border around elements and space elements in relation to other elements.

The image below illustrates the box model:



Explanation of the different parts:

- **Margin** – Clears an area around the border. The margin does not have a background color, it is completely transparent
- **Border** – A border that goes around the padding and content. The border is affected by the background color of the box
- **Padding** – Clears an area around the content. The padding is affected by the background color of the box
- **Content** – The content of the box, where text and images appear

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

13.1 Width and Height of an Element

Important: When you specify the width and height properties of an element with CSS, you are just setting the width and height of the content area. To know the full size of the element, you must also add the padding, border and margin.

The total width of the element in the example below is 300px:

```
width:250px;
padding:10px;
border:5px solid gray;
margin:10px;
```

Let's do the math:

```
250px (width)
+ 20px (left and right padding)
+ 10px (left and right border)
+ 20px (left and right margin)
= 300px
```

Imagine that you only had 250px of space. Let's make an element with a total width of 250px:

```
width:220px;
padding:10px;
border:5px solid gray;
margin:0px;
```

The total width of an element should always be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should always be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

13.2 Browsers Compatibility Issue

If you tested the previous example in Internet Explorer, you saw that the total width was not exactly 250px.

IE includes padding and border in the width, when the width property is set, **unless a DOCTYPE is declared**.

To fix this problem, just add a DOCTYPE to the code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
div.ex {
width:220px;
padding:10px;
border:5px solid gray;
margin:0px;}
</style>
</head>
```

14. CSS Border

The CSS border properties allow you to specify the style and color of an element's border.

14.1 Border Style

The border-style property specifies what kind of border to display.

None of the border properties will have ANY effect unless the **border-style** property is set!

border-style values:

none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

groove: Defines a 3D grooved border. The effect depends on the border-color value

ridge: Defines a 3D ridged border. The effect depends on the border-color value

inset: Defines a 3D inset border. The effect depends on the border-color value

outset: Defines a 3D outset border. The effect depends on the border-color value

14.2 Border Width

The border-width property is used to set the width of the border.

The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

```
p.one
{
border-style:solid;
border-width:5px;
}
p.two
{
border-style:solid;
border-width:medium;
}
```

14.3 Border Color

The border-color property is used to set the color of the border. The color can be set by:

- name – specify a color name, like "red"
- RGB – specify a RGB value, like "rgb(255,0,0)"
- Hex – specify a hex value, like "#ff0000"

You can also set the border color to "transparent".

Note: The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

```
p.one
{
border-style:solid;
border-color:red;
}
p.two
{
border-style:solid;
border-color:#98bf21;
}
```

14.4 Border – Individual sides

In CSS it is possible to specify different borders for different sides:

```
p
{
border-top-style:dotted;
```

```
border-right-style:solid;
border-bottom-style:dotted;
border-left-style:solid;
}
```

The example above can also be set with a single property:

```
border-style:dotted solid;
```

The border-style property can have from one to four values.

- **border-style:dotted solid double dashed;**
 - o top border is dotted
 - o right border is solid
 - o bottom border is double
 - o left border is dashed
- **border-style:dotted solid double;**
 - o top border is dotted
 - o right and left borders are solid
 - o bottom border is double
- **border-style:dotted solid;**
 - o top and bottom borders are dotted
 - o right and left borders are solid
- **border-style:dotted;**
 - o all four borders are dotted

The border-style property is used in the example above. However, it also works with border-width and border-color.

14.5 Border – Shorthand property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the border properties in one property. This is called a shorthand property.

The shorthand property for the border properties is "border":

```
border:5px solid red;
```

When using the border property, the order of the values are:

- border-width
- border-style
- border-color

It does not matter if one of the values above are missing (although, border-style is required), as long as the rest are in the specified order.

15. CSS Margin

The CSS margin properties define the space around elements.

15.1 Margin

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent.

The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Possible Values

Value	Description
auto	The browser sets the margin. The result of this is dependent of the browser
length	Defines a fixed margin (in pixels, pt, em, etc.)
%	Defines a margin in % of the containing element

It is possible to use negative values, to overlap content.

15.2 Margin – Individual sides

In CSS, it is possible to specify different margins for different sides:

```
margin-top:100px;  
margin-bottom:100px;  
margin-right:50px;  
margin-left:50px;
```

15.3 Margin – Shorthand property

To shorten the code, it is possible to specify all the margin properties in one property. This is called a shorthand property.

The shorthand property for all the margin properties is "margin":

```
margin:100px 50px;
```

The margin property can have from one to four values.

- **margin:25px 50px 75px 100px;**
 - o top margin is 25px
 - o right margin is 50px
 - o bottom margin is 75px
 - o left margin is 100px
- **margin:25px 50px 75px;**
 - o top margin is 25px
 - o right and left margins are 50px
 - o bottom margin is 75px
- **margin:25px 50px;**
 - o top and bottom margins are 25px
 - o right and left margins are 50px
- **margin:25px;**
 - o all four margins are 25px

16. CSS Padding

The CSS padding properties define the space between the element border and the element content.

16.1 Padding

The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element.

The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all paddings at once.

Possible Values

Value	Description
Length	Defines a fixed margin (in pixels, pt, em, etc.)
%	Defines a margin in % of the containing element

16.2 Padding – Individual sides

In CSS, it is possible to specify different padding for different sides:

```
padding-top:25px;  
padding-bottom:25px;  
padding-right:50px;  
padding-left:50px;
```

16.3 Padding – Shorthand property

To shorten the code, it is possible to specify all the padding properties in one property. This is called a shorthand property.

The shorthand property for all the padding properties is "padding":

```
padding:25px 50px;
```

The padding property can have from one to four values.

- **padding:25px 50px 75px 100px;**
 - o top padding is 25px
 - o right padding is 50px
 - o bottom padding is 75px
 - o left padding is 100px
- **padding:25px 50px 75px;**
 - o top padding is 25px
 - o right and left paddings are 50px
 - o bottom padding is 75px
- **padding:25px 50px;**
 - o top and bottom paddings are 25px
 - o right and left paddings are 50px
- **padding:25px;**
 - o all four paddings are 25px

17. CSS Grouping and Nesting Selectors

17.1 Grouping Selectors

In style sheets there are often elements with the same style.

```
h1
{
color:green;
}
h2
{
color:green;
}
p
{
color:green;
}
```

To minimize the code, you can group selectors.

Separate each selector with a comma.

In the example below we have grouped the selectors from the code above:

```
h1,h2,p
{
color:green;
}
```

17.2 Nesting Selectors

It is possible to apply a style for a selector within a selector.

In the example below, one style is specified for all p elements, and a separate style is specified for p elements nested within the "marked" class:

```
p
{
color:blue;
text-align:center;
}
.marked
{
background-color:blue;
}
.marked p
{
color:white;
}
```

18. CSS Display and Visibility

The display property specifies if/how an element is displayed, and the visibility property specifies if an element should be visible or hidden.

Box 1



Box 2



Box 3



18.1 Hiding an Element – display:none or visibility:hidden

Hiding an element can be done by setting the display property to "none" or the visibility property to "hidden". However, notice that these two methods produce different results:

visibility:hidden hides an element, but it will still take up the same space as before. The element will be hidden, but still affect the layout.

```
h1.hidden {visibility:hidden;}
```

display:none hides an element, and it will not take up any space. The element will be hidden, and the page will be displayed as the element is not there:

```
h1.hidden {display:none;}
```

18.2 CSS Display – Block and Inline Elements

A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

- `<h1>`
- `<p>`
- `<div>`

An inline element only takes up as much width as necessary, and does not force line breaks.

Examples of inline elements:

- ``
- `<a>`

18.3 Changing How an Element is Displayed

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow web standards.

The following example displays list items as inline elements:

```
li {display:inline;}
```

The following example displays span elements as block elements:

```
span {display:block;}
```

Note: Changing the display type of an element changes only how the element is displayed, NOT what kind of element it is. For example:

An inline element set to `display:block` is not allowed to have a block element nested inside of it.

19. CSS Positioning

- Positioning can be tricky sometimes!
- Decide which element to display in front!
- Elements can overlap!

19.1 Positioning

The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element's content is too big.

Elements can be positioned using the `top`, `bottom`, `left`, and `right` properties. However, these properties will not work unless the `position` property is set first. They also work differently depending on the positioning method.

There are four different positioning methods.

Static Positioning

HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page. Static positioned elements are not affected by the top, bottom, left, and right properties.

Fixed Positioning

An element with fixed position is positioned relative to the browser window.

It will not move even if the window is scrolled:

```
p.pos_fixed
{
position:fixed;
top:30px;
right:5px;
}
```

Note: Internet Explorer supports the fixed value only if a !DOCTYPE is specified.

Fixed positioned elements are removed from the normal flow. The document and other elements behave like the fixed positioned element does not exist.

Fixed positioned elements can overlap other elements.

Relative Positioning

A relative positioned element is positioned relative to its normal position.

```
h2.pos_left
{
position:relative;
left:-20px;
}
h2.pos_right
{
position:relative;
left:20px;
}
```

The content of a relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.

```
h2.pos_top
{
position:relative;
top:-50px;
}
```

Relatively positioned element are often used as container blocks for absolutely positioned elements.

Absolute Positioning

An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>:

```
h2
{
position:absolute;
left:100px;
top:150px;
}
```

Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist.

Absolutely positioned elements can overlap other elements.

19.2 Overlapping Elements

When elements are positioned outside the normal flow, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a positive or negative stack order:

```
img
{
position:absolute;
left:0px;
top:0px;
z-index:-1
}
```

An element with greater stack order is always in front of an element with a lower stack order.

Note: If two positioned elements overlap, without a z-index specified, the element positioned last in the HTML code will be shown on top.

20. CSS Float

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.

Float is very often used for images, but it is also useful when working with layouts.

20.1 How Elements Float

Elements are floated horizontally, this means that an element can only be floated left or right, not up or down.

A floated element will move as far to the left or right as it can. Usually this means all the way to the left or right of the containing element.

The elements after the floating element will flow around it.

The elements before the floating element will not be affected.

If an image is floated to the right, a following text flows around it, to the left:

```
img
{
float:right;
}
```

20.2 Floating Elements Next to Each Other

If you place several floating elements after each other, they will float next to each other if there is room.

Here we have made an image gallery using the float property:

```
.thumbnail
{
float:left;
width:110px;
height:90px;
margin:5px;
}
```

20.3 Turning off Float – Using Clear

Elements after the floating element will flow around it. To avoid this, use the clear property.

The clear property specifies which sides of an element other floating elements are not allowed.

Add a text line into the image gallery, using the clear property:

```
.text_line
{
clear:both;
}
```

21. CSS Horizontal Align

In CSS, several properties are used to align elements horizontally.

21.1 Aligning Block Elements

A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

- <h1>
- <p>
- <div>

21.2 Center Aligning Using the margin Property

Block elements can be aligned by setting the left and right margins to "auto".

Note: Using margin:auto will not work in Internet Explorer, **unless a !DOCTYPE is declared.**

Setting the left and right margins to auto specifies that they should split the available margin equally. The result is a centered element:

Example

```
.center
{
margin-left:auto;
margin-right:auto;
width:70%;
background-color:#b0e0e6;
}
```

Tip: Aligning has no effect if the width is 100%.

Note: In IE 5 there is a margin handling bug for block elements. To make the example above work in IE5, add some extra code.

21.3 Left and Right Aligning Using the position Property

One method of aligning elements is to use absolute positioning:

Example

```
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

21.4 Crossbrowser Compatibility Issues

When aligning elements like this, it is always a good idea to predefine margin and padding for the <body> element. This is to avoid visual differences in different browsers.

There is also another problem with IE when using the position property. If a container element (in our case <div class="container">) has a specified width, and the !DOCTYPE declaration is missing, IE will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. Always set the !DOCTYPE declaration when using the position property:

Example

```
body
{
margin:0;
padding:0;
}
.container
{
position:relative;
width:100%;
}
.right
{
position:absolute;
right:0px;
width:300px;
background-color:#b0e0e6;
}
```

21.5 Left and Right Aligning Using the float Property

One method of aligning elements is to use the float property:

Example

```
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

21.6 Crossbrowser Compatibility Issues

When aligning elements like this, it is always a good idea to predefine margin and padding for the <body> element. This is to avoid visual differences in different browsers.

There is also another problem with IE when using the float property. If the !DOCTYPE declaration is missing, IE will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. Always set the !DOCTYPE declaration when using the float property:

Example

```
body
{
margin:0;
padding:0;
}
.right
{
float:right;
width:300px;
background-color:#b0e0e6;
}
```

22. CSS Pseudo-classes

CSS pseudo-classes are used to add special effects to some selectors.

Syntax

The syntax of pseudo-classes:

```
selector:pseudo-class {property:value;}
```

CSS classes can also be used with pseudo-classes:

```
selector.class:pseudo-class {property:value;}
```

22.1 Anchor Pseudo-classes

Links can be displayed in different ways in a CSS-supporting browser:

Example

```
a:link {color:#FF0000;} /* unvisited link */
a:visited {color:#00FF00;} /* visited link */
a:hover {color:#FF00FF;} /* mouse over link */
a:active {color:#0000FF;} /* selected link */
```

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!!

Note: a:active MUST come after a:hover in the CSS definition in order to be effective!!

Note: Pseudo-class names are not case-sensitive.

22.2 Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

```
a.red:visited {color:#FF0000;}  
<a class="red" href="css_syntax.asp">CSS Syntax</a>
```

If the link in the example above has been visited, it will be displayed in red.

22.3 CSS – The :first-child Pseudo-class

The :first-child pseudo-class matches a specified element that is the first child of another element.

Note: For :first-child to work in IE a `<!DOCTYPE>` must be declared.

22.4 Match the first <p> element

In the following example, the selector matches any <p> element that is the first child of any element:

Example

```
<html>  
<head>  
<style type="text/css">  
p:first-child  
{  
color:blue;  
}  
</style>  
</head>  
<body>  
<p>I am a strong man.</p>  
<p>I am a strong man.</p>  
</body>  
</html>
```

22.5 Match the first <i> element in all <p> elements

In the following example, the selector matches the first <i> element in all <p> elements:

Example

```
<html>  
<head>  
<style type="text/css">  
p > i:first-child  
{  
font-weight:bold;  
}  
</style>
```

```
</head>
<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
</body>
</html>
```

22.6 Match all <i> elements in all first child <p> elements

In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element:

Example

```
<html>
<head>
<style type="text/css">
p:first-child i
{
color:blue;
}
</style>
</head>
<body>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
<p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>
</body>
</html>
```

22.7 CSS – The :lang Pseudo-class

The :lang pseudo-class allows you to define special rules for different languages.

Note: Internet Explorer 8 (and higher) supports the :lang pseudo-class if a <!DOCTYPE> is specified.

In the example below, the :lang class defines the quotation marks for q elements with lang="no":

```
<html>
<head>
<style type="text/css">
q:lang(no) {quotes: "~" "~";}
</style>
</head>
<body>
<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>
</body>
</html>
```


23. CSS Pseudo-elements

CSS pseudo-elements are used to add special effects to some selectors.

The syntax of pseudo-elements:

```
selector:pseudo-element {property:value;}
```

CSS classes can also be used with pseudo-elements:

```
selector.class:pseudo-element {property:value;}
```

23.1 The :first-line Pseudo-element

The "first-line" pseudo-element is used to add a special style to the first line of a text.

In the following example the browser formats the first line of text in a p element according to the style in the "first-line" pseudo-element (where the browser breaks the line, depends on the size of the browser window):

Example

```
p:first-line
{
color:#ff0000;
font-variant:small-caps;
}
```

Note: The "first-line" pseudo-element can only be used with block-level elements.

Note: The following properties apply to the "first-line" pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

23.2 The :first-letter Pseudo-element

The "first-letter" pseudo-element is used to add a special style to the first letter of a text:

Example

```
p:first-letter
{
```

```
color:#ff0000;
font-size:xx-large;
}
```

Note: The "first-letter" pseudo-element can only be used with block-level elements.

Note: The following properties apply to the "first-letter" pseudo-element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

23.3 Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes:

```
p.article:first-letter {color:#ff0000;}
<p class="article">A paragraph in an article</p>
```

The example above will display the first letter of all paragraphs with class="article", in red.

23.4 Multiple Pseudo-elements

Several pseudo-elements can also be combined.

In the following example, the first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color:

Example

```
p:first-letter
{
color:#ff0000;
font-size:xx-large;
}
p:first-line
{
color:#0000ff;
```

```
font-variant:small-caps;
}
```

23.5 CSS – The :before Pseudo-element

The ":before" pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before each <h1> element:

```
h1:before
{
content:url(smiley.gif);
}
```

23.6 CSS – The :after Pseudo-element

The ":after" pseudo-element can be used to insert some content after the content of an element.

The following example inserts an image after each <h1> element:

Example

```
h1:after
{
content:url(smiley.gif);
}
```

24. CSS Navigation Bar

Having easy-to-use navigation is important for any web site.

With CSS you can transform boring HTML menus into good-looking navigation bars.

24.1 Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

In our examples we will build the navigation bar from a standard HTML list.

A navigation bar is basically a list of links, so using the and elements makes perfect sense:

Example

```
<ul>
<li><a href="default.asp">Home</a></li>
<li><a href="news.asp">News</a></li>
<li><a href="contact.asp">Contact</a></li>
<li><a href="about.asp">About</a></li>
</ul>
```

Now let's remove the bullets and the margins and padding from the list:

Example

```
ul
{
list-style-type:none;
margin:0;
padding:0;
}
```

Example explained:

- `list-style-type:none` – Removes the bullets. A navigation bar does not need list markers
- Setting margins and padding to 0 to remove browser default settings

The code in the example above is the standard code used in both vertical, and horizontal navigation bars.

24.2 Vertical Navigation Bar

To build a vertical navigation bar we only need to style the `<a>` elements, in addition to the code above:

Example

```
a
{
display:block;
width:60px;
}
```

Example explained:

- `display:block` – Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width
- `width: 60px` – Block elements take up the full width available by default. We want to specify a 60 px width

Note: Always specify the width for `<a>` elements in a vertical navigation bar. If you omit the width, IE6 can produce unexpected results.

24.3 Horizontal Navigation Bar

There are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items.

Both methods work fine, but if you want the links to be the same size, you have to use the floating method.

24.3.1 Inline List Items

One way to build a horizontal navigation bar is to specify the `` elements as inline, in addition to the "standard" code above:

Example

```
li
{
display:inline;
}
```

Example explained:

- `display:inline;` – By default, `` elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

24.3.2 Floating List Items

In the example above the links have different widths.

For all the links to have an equal width, float the `` elements and specify a width for the `<a>` elements:

Example

```
li
{
float:left;
}
a
{
display:block;
width:60px;
}
```

Example explained:

- `float:left` – use float to get block elements to slide next to each other
- `display:block` – Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width
- `width: 60px` – Since block elements take up the full width available, they cannot float next to each other. We specify the width of the links to 60px

25. CSS Image Gallery

CSS can be used to create an image gallery.



Add a description of the image here



Add a description of the image here



Add a description of the image here



Add a description of the image here

25.1 Image Gallery

The following image gallery is created with CSS:

Example

```

<html>
<head>
<style type="text/css">
div.img
{
margin:2px;
border:1px solid #0000ff;
height:auto;
width:auto;
float:left;
text-align:center;
}
div.img img
{
display:inline;
margin:3px;
border:1px solid #ffffff;
}
div.img a:hover img
{

```

```

border:1px solid #0000ff;
}
div.desc
{
text-align:center;
font-weight:normal;
width:120px;
margin:2px;
}
</style>
</head>
<body>
<div class="img">
<a target="_blank" href="klematis_big.htm">

</a>
<div class="desc">Add a description of the image here</div>
</div>
<div class="img">
<a target="_blank" href="klematis2_big.htm">

</a>
<div class="desc">Add a description of the image here</div>
</div>
<div class="img">
<a target="_blank" href="klematis3_big.htm">

</a>
<div class="desc">Add a description of the image here</div>
</div>
<div class="img">
<a target="_blank" href="klematis4_big.htm">

</a>
<div class="desc">Add a description of the image here</div>
</div>
</body>
</html>

```

26. CSS Image Opacity / Transparency

Creating transparent images with CSS is easy.

Note: This is not yet a CSS standard. However, it works in all modern browsers, and is a part of the W3C CSS 3 recommendation.

26.1 Example 1 – Creating a Transparent Image

First we will show you how to create a transparent image with CSS.

Regular image:



The same image with transparency:



Look at the following source code:

```

```

Firefox uses the property **opacity:x** for transparency, while IE uses **filter:alpha(opacity=x)**.

Tip: The CSS3 syntax for transparency is **opacity:x**.

In Firefox (opacity:x) x can be a value from 0.0 – 1.0. A lower value makes the element more transparent.

In IE (filter:alpha(opacity=x)) x can be a value from 0 – 100. A lower value makes the element more transparent.

26.2 Example 2 – Image Transparency – Mouseover Effect

Mouse over the images:



The source code looks like this:

```

<img src="klematis2.jpg" style="opacity:0.4;filter:alpha(opacity=40)"
```



```
onmouseover="this.style.opacity=1;this.filters.alpha.opacity=100"  
onmouseout="this.style.opacity=0.4;this.filters.alpha.opacity=40" />
```

We see that the first line of the source code is similar to the source code in Example 1. In addition, we have added an onmouseover attribute and an onmouseout attribute. The onmouseover attribute defines what will happen when the mouse pointer moves over the image. In this case we want the image to NOT be transparent when we move the mouse pointer over it.

The syntax for this in Firefox is: **this.style.opacity=1** and the syntax in IE is: **this.filters.alpha.opacity=100**.

When the mouse pointer moves away from the image, we want the image to be transparent again. This is done in the onmouseout attribute.

26.3 Example 3 – Text in Transparent Box

This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box. This is some text that is placed in the transparent box.

The source code looks like this:

```
<html>  
<head>  
<style type="text/css">  
div.background  
{  
width:500px;  
height:250px;  
background:url(klematis.jpg) repeat;  
border:2px solid black;  
}  
div.transbox  
{  
width:400px;  
height:180px;  
margin:30px 50px;  
background-color:#ffffff;  
border:1px solid black;  
/* for IE */  
filter:alpha(opacity=60);  
/* CSS3 standard */  
opacity:0.6;  
}  
div.transbox p
```

```

{
margin:30px 40px;
font-weight:bold;
color:#000000;
}
</style>
</head>
<body>
<div class="background">
<div class="transbox">
<p>This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
This is some text that is placed in the transparent box.
</p>
</div>
</div>
</body>
</html>

```

First, we create a div element (class="background") with a fixed height and width, a background image, and a border. Then we create a smaller div (class="transbox") inside the first div element. This div also have a fixed width, a background image, and a border. In addition we make this div transparent.

Inside the transparent div, we add some text inside a p element.

27. CSS Image Sprites

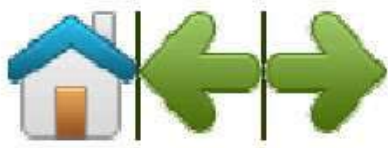
An image sprite is a collection of images put into a single image.

A web page with many images can take a long time to load and generates multiple server requests.

Using image sprites will reduce the number of server requests and save bandwidth.

27.1 Image Sprites – Simple Example

Instead of using three separate images, we use this single image ("img_navsprites.gif"):



With CSS, we can show just the part of the image we need.
In the following example the CSS specifies which part of the "img_navsprites.gif" image to show:

Example

```
img.home
{
width:46px;
height:44px;
background:url(img_navsprites.gif) 0 0;
}
```

Example explained:

- `` - Only defines a small transparent image because the src attribute cannot be empty. The displayed image will be the background image we specify in CSS
- `width: 46px;height:44px;` - Defines the portion of the image we want to use
- `background:url(img_navsprites.gif) 0 0;` - Defines the background image and its position (left 0px, top 0px)

This is the easiest way to use image sprites, now we want to expand it by using links and hover effects.

27.2 Image Sprites – Create a Navigation List

We want to use the sprite image ("img_navsprites.gif") to create a navigation list.

We will use an HTML list, because it can be a link and also supports a background image:

Example

```
#navlist{position:relative;}
#navlist li{margin:0;padding:0;list-style:none;position:absolute;top:0;}
#navlist li, #navlist a{height:44px;display:block;}
#home{left:0px;width:46px;}
#home{background:url('img_navsprites.gif') 0 0;}
#prev{left:63px;width:43px;}
#prev{background:url('img_navsprites.gif') -47px 0;}
#next{left:129px;width:43px;}
#next{background:url('img_navsprites.gif') -91px 0;}
```

Example explained:

- `#navlist{position:relative;}` - position is set to relative to allow absolute positioning inside it

- `#navlist li{margin:0;padding:0;list-style:none;position:absolute;top:0;}` – margin and padding is set to 0, list-style is removed, and all list items are absolute positioned
- `#navlist li, #navlist a{height:44px;display:block;}` – the height of all the images are 44px

Now start to position and style for each specific part:

- `#home{left:0px;width:46px;}` – Positioned all the way to the left, and the width of the image is 46px
- `#home{background:url(img_navsprites.gif) 0 0;}` – Defines the background image and its position (left 0px, top 0px)
- `#prev{left:63px;width:43px;}` – Positioned 63px to the right (`#home` width 46px + some extra space between items), and the width is 43px.
- `#prev{background:url('img_navsprites.gif') -47px 0;}` – Defines the background image 47px to the right (`#home` width 46px + 1px line divider)
- `#next{left:129px;width:43px;}` – Positioned 129px to the right (start of `#prev` is 63px + `#prev` width 43px + extra space), and the width is 43px.
- `#next{background:url('img_navsprites.gif') no-repeat -91px 0;}` – Defines the background image 91px to the right (`#home` width 46px + 1px line divider + `#prev` width 43px + 1px line divider)

27.3 Image Sprites – Hover Effect

Now we want to add a hover effect to our navigation list.

Our new image ("img_navsprites_hover.gif") contains three navigation images and three images to use for hover effects:



Because this is one single image, and not six separate files, there will be **no loading delay** when a user hovers over the image.

We only add three lines of code to add the hover effect:

Example

```
#home a:hover{background: url('img_navsprites_hover.gif') 0 -45px;}
#prev a:hover{background: url('img_navsprites_hover.gif') -47px -45px;}
```

```
#next a:hover{background: url('img_navsprites_hover.gif') -91px -45px;}
```

Example explained:

- Since the list item contains a link, we can use the :hover pseudo-class
- #home a:hover {background: transparent url (img_navsprites_hover.gif) 0 - 45px;} - For all three hover images we specify the same background position, only 45px further down

28. CSS Media Types

Media Types allow you to specify how documents will be presented in different media. The document can be displayed differently on the screen, on the paper, with an aural browser, etc.

28.1 Media Types

Some CSS properties are only designed for a certain media. For example the "voice-family" property is designed for aural user agents. Some other properties can be used for different media types. For example, the "font-size" property can be used for both screen and print media, but perhaps with different values. A document usually needs a larger font-size on a screen than on paper, and sans-serif fonts are easier to read on the screen, while serif fonts are easier to read on paper.

28.2 The @media Rule

The @media rule allows different style rules for different media in the same style sheet.

The style in the example below tells the browser to display a 14 pixels Verdana font on the screen. But if the page is printed, it will be in a 10 pixels Times font. Notice that the font-weight is set to bold, both on screen and on paper:

```
<html>
<head>
<style>
@media screen
{
p.test {font-family:verdana,sans-serif;font-size:14px;}
}
@media print
{
p.test {font-family:times,serif;font-size:10px;}
```

```

}
@media screen,print
{
p.test {font-weight:bold;}
}
</style>
</head>
<body>
....
</body>
</html>

```

See it yourself ! If you are using Mozilla/Firefox or IE 5+ and print this page, you will see that the paragraph under "Media Types" will be displayed in another font, and have a smaller font size than the rest of the text.

28.3 Different Media Types

Note: The media type names are not case-sensitive.

Media Type	Description
all	Used for all media type devices
aural	Used for speech and sound synthesizers
braille	Used for braille tactile feedback devices
embossed	Used for paged braille printers
handheld	Used for small or handheld devices
print	Used for printers
projection	Used for projected presentations, like slides
screen	Used for computer screens
tty	Used for media using a fixed-pitch character grid, like teletypes and terminals
tv	Used for television-type devices

29. CSS Attribute Selectors

It is possible to style HTML elements that have specific attributes, not just class and id.

Note: Internet Explorer 7 (and higher) supports attribute selectors only if a !DOCTYPE is specified. Attribute selection is **NOT** supported in IE6 and lower.

29.1 Attribute Selector

The example below styles all elements with a title attribute:

Example

```
[title]
```

```
{  
color:blue;  
}
```

29.2 Attribute and Value Selector

The example below styles all elements with title="W3Schools":

Example

```
[title=W3Schools]  
{  
border:5px solid green;  
}
```

29.3 Attribute and Value Selector – Multiple Values

The example below styles all elements with a title attribute that contains a specified value. This works even if the attribute has space separated values:

Example

```
[title~=hello] { color:blue; }
```

The example below styles all elements with a lang attribute that contains a specified value. This works even if the attribute has hyphen (-) separated values:

Example

```
[lang|=en] { color:blue; }
```

29.4 Styling Forms

The attribute selectors are particularly useful for styling forms without class or ID:

Example

```
input[type="text"]  
{  
width:150px;  
display:block;  
margin-bottom:10px;  
background-color:yellow;  
}  
input[type="button"]  
{  
width:120px;  
margin-left:35px;  
display:block;  
}
```

30. CSS Don't

Here are some technologies you should try to avoid when using CSS.

30.1 Internet Explorer Behaviors

What is it? Internet Explorer 5 introduced behaviors. Behaviors are a way to add behaviors to HTML elements with the use of CSS styles.

Why avoid it? The behavior attribute is only supported by Internet Explorer.

What to use instead? Use JavaScript and the [HTML DOM](#) instead.

30.2 Example 1 – Mouseover Highlight

The following HTML file has a <style> element that defines a behavior for the <h1> element:

```
<html>
<head>
<style type="text/css">
h1 { behavior:url(behavior.htc); }
</style>
</head>
<body>
<h1>Mouse over me!!!</h1>
</body>
</html>
```

The XML document "behavior.htc" is shown below:

Example (IE 5+ Only)

The behavior file contains a JavaScript and event handlers for the elements.

```
<attach for="element" event="onmouseover" handler="hig_lite" />
<attach for="element" event="onmouseout" handler="low_lite" />
<script type="text/javascript">
function hig_lite()
{
element.style.color='red';
}
function low_lite()
{
element.style.color='blue';
}
</script>
```


30.3 Example 2 – Typewriter Simulation

The following HTML file has a <style> element that defines a behavior for elements with an id of "typing":

```
<html>
<head>
<style type="text/css">
#typing
{
behavior:url(behave_typing.htc);
font-family:"courier new";
}
</style>
</head>
<body>
<span id="typing" speed="100">IE5 introduced DHTML behaviors.
Behaviors are a way to add DHTML functionality to HTML elements
with the ease of CSS.<br /><br />How do behaviors work?<br />
By using XML we can link behaviors to any element in a web page
and manipulate that element.</p>
</span>
</body>
</html>
```

The XML document "typing.htc" is shown below:

Example (IE 5+ Only)

```
<attach for="window" event="onload" handler="beginTyping" />
<method name="type" />
<script type="text/javascript">
var i,text1,text2,textLength,t;
function beginTyping()
{
i=0;
text1=element.innerText;
textLength=text1.length;
element.innerText="";
text2="";
t=window.setInterval(element.id+ ".type()",speed);
}
function type()
{
text2=text2+ text1.substring(i,i+ 1);
element.innerText=text2;
i=i+ 1;
if (i==textLength)
```

```
{  
clearInterval(t);  
}  
}  
</script>
```

Web Organization LAB



What is HTML?

- ✦ HTML is a language for describing Web pages.
- ✦ HTML stands for HyperText Markup Language.
- ✦ HTML is not a programming language, it is a markup language.
- ✦ A markup language is a collection of markup tags.
- ✦ HTML uses markup tags to describe Web pages.

What are Tags?

- ✦ HTML markup tags are usually called HTML tags or just tags.
- ✦ HTML tags are keywords surrounded by angle brackets like `<html>`.
- ✦ HTML tags normally come in pairs, like `` and ``.
- ✦ The first tag in a pair is the start tag; the second tag is the end tag.
- ✦ Start and end tags are also called opening tags and closing tags.

HTML Documents = Web Pages

```
<html>
<head>
</head>
<body>
    <h1>My First Heading</h1>
    <p>My first paragraph</p>
</body>
</html>
```

In the previous code example,

- ✦ The text between `<html>` and `</html>` describes the Web page.
- ✦ The text between `<head>` and `</head>` is head of the Web page.
- ✦ The text between `<body>` and `</body>` is the visible page content.
- ✦ The text between `<h1>` and `</h1>` is displayed as a heading.
- ✦ The text between `<p>` and `</p>` is displayed as a paragraph.

My First Heading

My first paragraph

.HTM or .HTML Extension?

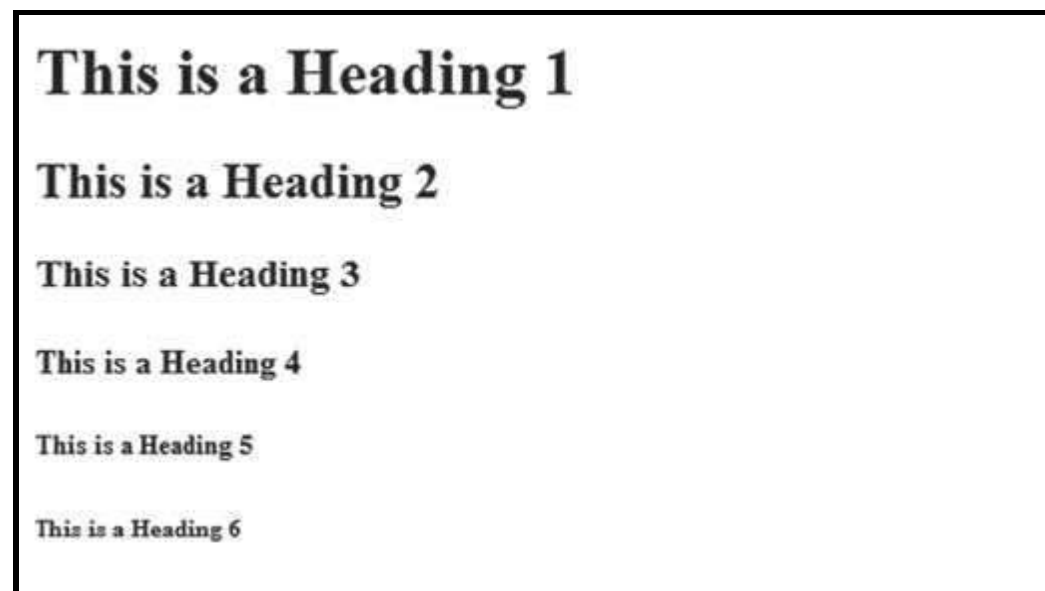
When you save an HTML file, you can use either the `.htm` or the `.html` extension.

Web Organization LAB

We use .htm in our examples. It is a habit from the past, when the software only allowed three letters in file extensions.

HTML Headings

```
<html>
<body>
    <h1>This is a Heading 1</h1>
    <h2>This is a Heading 2</h2>
    <h3>This is a Heading 3</h3>
    <h4>This is a Heading 4</h4>
    <h5>This is a Heading 5</h5>
    <h6>This is a Heading 6</h6>
</body>
</html>
```



- ✦ Use HTML headings for headings only. Don't use headings to make text BIG or bold.
- ✦ Search engines use your headings to index the structure and content of your Web pages.
- ✦ Browsers automatically add an empty line before and after headings.

HTML Paragraphs

```
<html>
<body>
    <p>This is a paragraph.</p>
    <p>This is a paragraph.</p>
    <p>This is a paragraph.</p>
</body>
</html>
```

Web Organization LAB

This is a paragraph.

This is a paragraph.

This is a paragraph.

- ✦ The `<p>` element defines a new paragraph in the HTML document.
- ✦ The element has a start tag `<p>` and an end tag `</p>`.
- ✦ Most browsers automatically add an empty line before and after paragraphs.

The `<body>` element

The `<body>` element defines the body of the HTML document.

```
<body>
```

```
  <p>This is my first paragraph</p>
```

```
</body>
```

- ✦ The element has a start tag `<body>` and an end tag `</body>`.
- ✦ The element content is another HTML element (one or more paragraphs).
- ✦ There are usually dozens of elements within the body element.

The `<html>` element

The `<html>` element defines the entire HTML document.

```
<html>
```

```
<body>
```

```
  <p>This is my first paragraph</p>
```

```
</body>
```

```
</html>
```

- ✦ The element has a start tag `<html>` and an end tag `</html>`.
- ✦ The element content is another HTML element (the body).

Don't Forget the End Tag

Most browsers will display HTML correctly even if you forget the end tag.

```
<p>This is a paragraph
```

```
<p>This is another paragraph
```

The previous example will work in most browsers, but don't rely on it. Forgetting the end tag can produce unexpected results or errors. Future versions of HTML will not allow you to skip end tags.

Empty HTML Elements

- ✦ HTML elements without content are called empty elements. Empty elements can be closed within the start tag.
- ✦ `
` is an empty element without a closing tag. It defines a line break.
- ✦ In XML and future versions of HTML, all elements must be closed.

Web Organization LAB

- ✦ Adding a slash to the end of start tag, like `
`, is the proper way of closing empty elements, accepted by HTML, and XML.
- ✦ Even if `
` works in all browsers, writing `
` instead is more future proof.

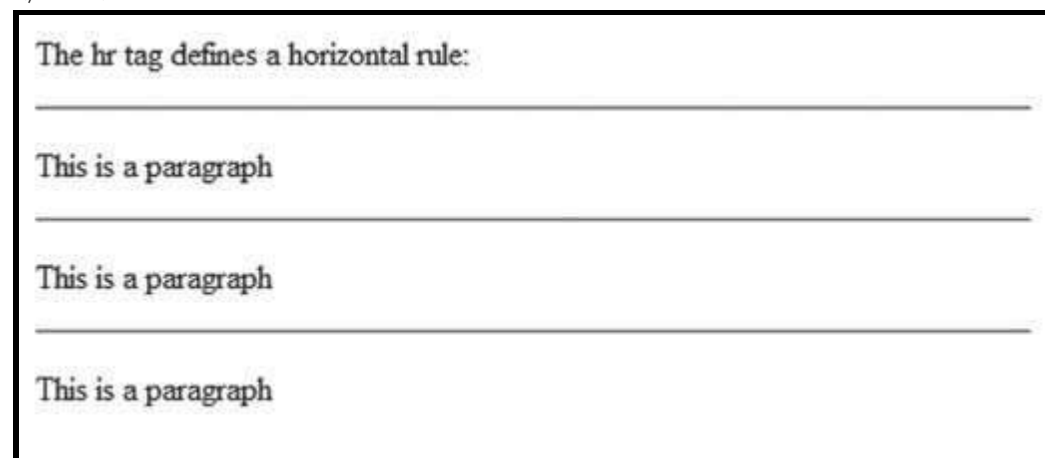
Use Lowercase Tags

- ✦ HTML tags are not case sensitive: `<P>` means the same as `<p>`. Plenty of Web sites use uppercase HTML tags in their pages.
- ✦ World Wide Web Consortium (W3C) recommends lowercase in HTML4.

HTML Rules (Lines)

The `<hr />` tag is used to create a horizontal rule (line) across the browser page.

```
<html>
<body>
  <p>The hr tag defines a horizontal rule:</p>
  <hr />
  <p>This is a paragraph</p>
  <hr />
  <p>This is a paragraph</p>
  <hr />
  <p>This is a paragraph</p>
</body>
</html>
```



HTML Comments

- ✦ Comments can be inserted in the HTML code to make it more readable and understandable.
- ✦ Comments are ignored by the browser and are not displayed

```
<html>
<body>
  <!--This comment will not be displayed-->
  <p>This is a regular paragraph</p>
</body>
</html>
```

Web Organization LAB

This is a regular paragraph

HTML Formatting Tags

```
<html>
<body>
  <p><b>This text is bold</b></p>
  <p><strong>This text is strong</strong></p>
  <p><big>This text is big</big></p>
  <p><em>This text is emphasized</em></p>
  <p><i>This text is italic</i></p>
  <p><small>This text is small</small></p>
  <p>This is<sub> subscript</sub> and <sup>superscript</sup></p>
</body>
</html>
```

This text is bold

This text is strong

This text is big

This text is emphasized

This text is italic

This text is small

This is subscript and superscript

Preformatted Text

This example demonstrates how you can control the line breaks, spaces, and character widths with the `<pre>` tag.

```
<html>
<body>
<pre>
This is
preformatted text.
It preserves      both spaces
and line breaks and shows the text in a monospace font.
</pre>
<p>The pre tag is good for displaying computer code:</p>
<pre>
for i = 1 to 10
```

Web Organization LAB

```
        print i
    next i
</pre>
</body>
</html>
```

```
This is
preformatted text.
It preserves      both spaces
and line breaks and shows the text in a monospace font.

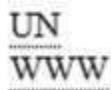
The pre tag is good for displaying computer code:

for i = 1 to 10
    print i
next i
```

Abbreviations and Acronyms

This example demonstrates how to handle an abbreviation or an acronym.

```
<html>
<body>
    <abbr title="United Nations">UN</abbr><br />
    <acronym title="World Wide Web">WWW</acronym>
    <p>The title attribute is used to show the spelled-out version when
    holding the mouse pointer over the acronym or abbreviation.</p>
</body>
</html>
```

The image shows two logos side-by-side. On the left is the United Nations logo, consisting of the letters 'UN' above a world map. On the right is the World Wide Web logo, consisting of the letters 'WWW' above a globe.

The title attribute is used to show the spelled-out version when holding the mouse pointer over the acronym or abbreviation.

Deleted and Inserted Text

This example demonstrates how to mark a text that is deleted (strikethrough) or inserted (underscore) to a document.

```
<html>
<body>
    <p>a dozen is<del>twenty</del><ins>twelve</ins>pieces</p>
    <p>Most browsers will <del>overstrike</del> deleted text and
    <ins>underscore</ins> inserted text.</p>
    <p>Some older browsers will display deleted or inserted text as
    plain text.</p>
</body>
</html>
```


Web Organization LAB

a dozen is ~~twenty~~ twelve pieces

Most browsers will ~~overstrike~~ deleted text and underscore inserted text.

Some older browsers will display deleted or inserted text as plain text.

TAG	DESCRIPTION
	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
	Defines deleted text
<s>	Deprecated. Use instead
<strike>	Deprecated. Use instead
<u>	Deprecated. Use styles instead

Defines bold text

Defines big text

Defines emphasized text

Defines italic text

Defines small text

Defines strong text

Defines subscripted text

Defines superscripted text

Defines inserted text

~~Defines deleted text~~

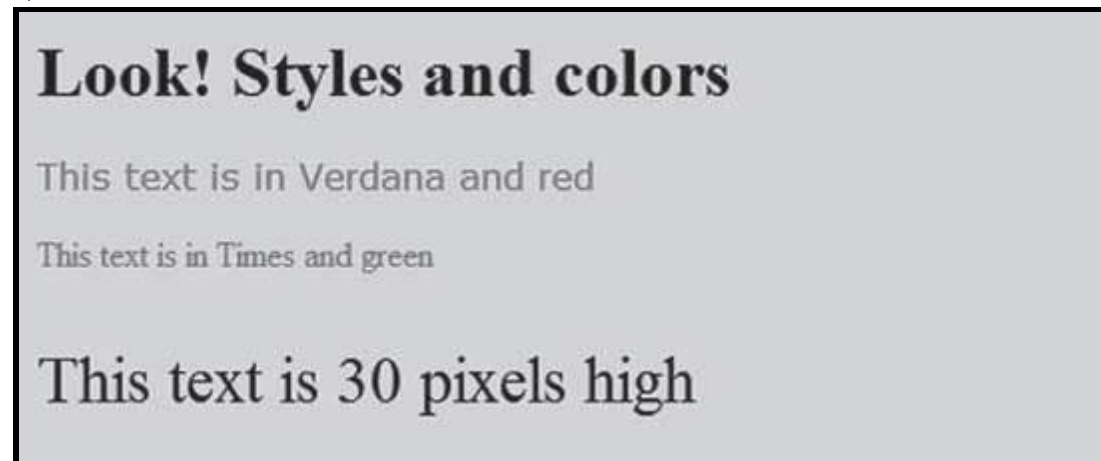


HTML Styles

The HTML Style Attribute

With HTML styles, formatting and attributes can be added to HTML elements directly by using the style attribute.

```
<html>
<body style="background-color:Gray;">
  <h1>Look! Styles and colors</h1>
  <p style="font-family:verdana;color:red">
    This text is in Verdana and red</p>
  <p style="font-family:times;color:green">
    This text is in Times and green</p>
  <p style="font-size:30px">This text is 30 pixels high</p>
</body>
</html>
```



Deprecated Tags and Attributes

In HTML 4, some tags and attributes are defined as deprecated. Deprecated means that they will not be supported in future versions of HTML and the message is clear: Avoid the use of deprecated tags and attributes.

These tags and attributes should be avoided, and styles should be used instead:

Web Organization LAB

TAGS	DESCRIPTION
<center>	Defines centered content
 and <basefont>	Defines HTML fonts
<s> and <strike>	Defines strikeout text
<u>	Defines underlined text
ATTRIBUTES	DESCRIPTION
align	Defines the alignment of text
bgcolor	Defines the background color
color	Defines the text color

Background Color

```
<body style="background-color:gray">
```

The style attribute defines a style for the <body> element.

```
<html>
```

```
<body style="background-color:gray">
```

```
<h2>Look: Colored Background!</h2>
```

```
</body>
```

```
</html>
```

Look: Colored Background!

The new style attribute makes the "old" bgcolor attribute obsolete.

```
<html>
```

```
<body bgcolor="gray">
```

```
<h2>Look: Colored Background!</h2>
```

```
<p>For future-proof HTML, use HTML styles instead:</p>
```

```
<p style="background-color:gray"></p>
```

```
</body>
```

```
</html>
```

Look: Colored Background!

For future-proof HTML, use HTML styles instead:

```
style="background-color:gray"
```

Font Family, Color, and Size

The style attribute defines a style for the <p> element

```
<html>
```

```
<body>
```

```
<h1 style="font-family:verdana">A heading</h1>
```

```
<p style="font-family:courier new; color:red; fontsize:20px;">A  
paragraph</p>
```

```
</body>
```

```
</html>
```

Web Organization LAB

A heading

A paragraph

The new style attribute makes the old tag

```
<html>
```

```
<body>
```

```
  <p><font size="2" face="Verdana">This is a paragraph.</font></p>
```

```
  <p><font size="5" face="Times" color="red">This is another  
  paragraph.</font></p>
```

```
</body>
```

```
</html>
```

This is a paragraph.

This is another paragraph.

Text Alignment

```
<h1 style="text-align:center">
```

The style attribute defines a style for the <h1> element

```
<html>
```

```
<body>
```

```
  <h1 style="text-align:center">This is heading 1</h1>
```

```
  <p>The heading above is aligned to the center of this page.
```

```
  The heading above is aligned to the center of this page.
```

```
  The heading above is aligned to the center of this page.</p>
```

```
</body>
```

```
</html>
```

This is heading 1

The heading above is aligned to the center of this page. The heading above is aligned to the center of this page. The heading above is aligned to the center of this page.

Although they display similarly in the browser, the new style attribute makes the old align attribute obsolete.

```
<html>
```

```
<body>
```

```
  <h1 align="center">This is heading 1</h1>
```

```
  <p>The heading above is aligned to the center of this page.
```

Web Organization LAB

The heading above is aligned to the center of this page.

The heading above is aligned to the center of this page. </p>

</body>

</html>

This is heading 1

The heading above is aligned to the center of this page. The heading above is aligned to the center of this page. The heading above is aligned to the center of this page.

HTML Links

The start tag contains attributes about the link.

Link text

The element content (Link text) defines the part to be displayed. The element content doesn't have to be text. You can link from an image or any other HTML element.

Hyperlinks, Anchors, and Links

- ✦ In Web terms, a hyperlink is a reference (an address) to a resource on the Web.
- ✦ Hyperlinks can point to any resource on the Web: an HTML page, an image, a sound file, a movie, and so on.
- ✦ An HTML anchor is a term used to define a hyperlink destination inside a document.
- ✦ The anchor element <a> defines both hyperlinks and anchors.

href Attribute

The href attribute defines the link "address". The following code will display in a browser:

Visit w3schools!

Visit w3schools!

The target Attribute

The target attribute defines where the linked document will be opened.

<html>

<body>

Visit w3schools!

<p>If you set the target attribute of a link to "_blank", the link will open in a new window.</p>

</body>

</html>

Web Organization LAB

You can use the following options for the target attribute:

OPTION	DESCRIPTION
_blank	Opens the linked document in a new window
_self	Opens the linked document in the same frame as it was clicked (this is default)
_parent	Opens the linked document in the parent frameset
_top	Opens the linked document in the full body of the window
framename	Opens the linked document in a named frame

The name Attribute

When the name attribute is used, the <a> element defines a named anchor inside an HTML document. Named anchors are not displayed in any special way by the browser because they are invisible to the reader.

Named anchors are sometimes used to create a table of contents at the beginning of a large document. Each chapter within the document is given a named anchor, and links to each of these anchors are put at the top of the document.

If a browser cannot find a named anchor that has been specified, it goes to the top of the document. No error occurs.

Named anchor syntax:

```
<a name="label">Any content</a>
```

The link syntax to a named anchor:

```
<a href="#label">Any content</a>
```

The # in the href attribute defines a link to a named anchor. A named anchor inside an HTML document:

```
<a name="tips">Useful Tips Section</a>
```

A link to the "Useful Tips Section" from elsewhere in the same document:

```
<a href="#tips">Jump to the Useful Tips Section</a>
```

A link to the "Useful Tips Section" from another document:

```
<a href="http://www.w3schools.com/html_tutorial.htm#tips">Jump to the Useful Tips Section</a>
```

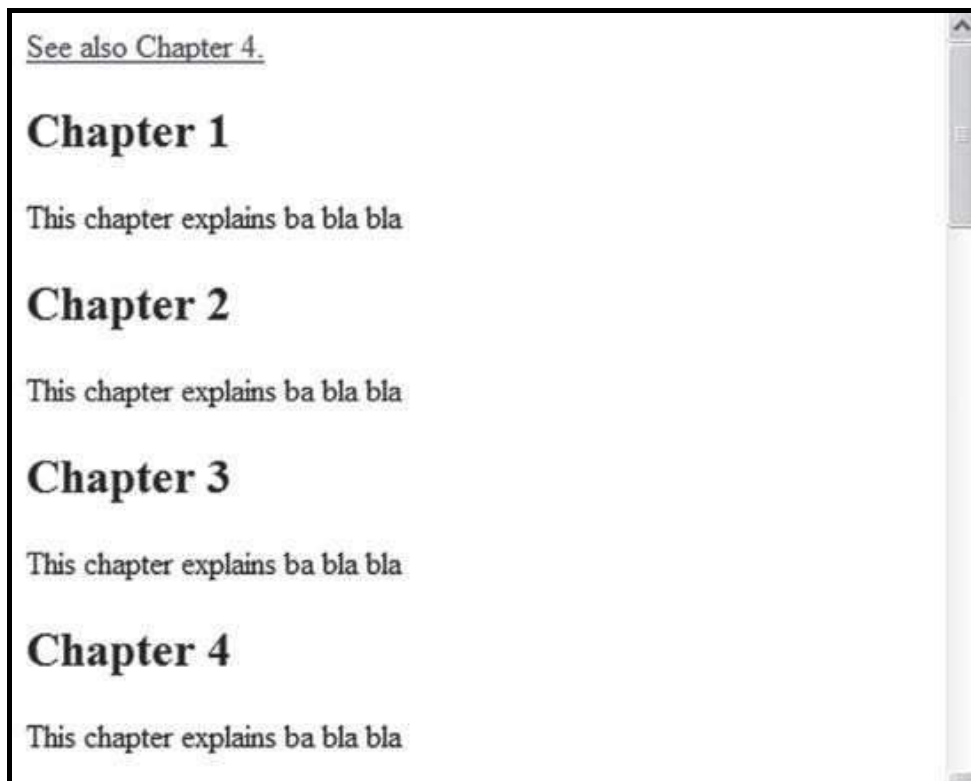
TIP: Always add a trailing slash to subfolder references. If you link like this: href="http://www.w3schools.com/html", you will generate two HTTP requests to the server because the server will add a slash to the address and create a new request like this: href="http://www.w3schools.com/html/"

Links on the Same Page

The following code example demonstrates how to use a link to jump to another part of a document.

Web Organization LAB

```
<html>
<body>
  <p><a href="#C4">See also Chapter 4.</a></p>
  <h2>Chapter 1</h2>
  <p>This chapter explains ba bla bla</p>
  <h2>Chapter 2</h2>
  <p>This chapter explains ba bla bla</p>
  <h2>Chapter 3</h2>
  <p>This chapter explains ba bla bla</p>
  <h2><a name="C4">Chapter 4</a></h2>
  <p>This chapter explains ba bla bla</p>
</body>
</html>
```



Creating a mailto: Link

The following example demonstrates how to link to an e-mail address and generate a new e-mail message in your default e-mail application (this works only if you have mail installed).

```
<html>
<body>
  <p>This is a mail link:
  <a href="mailto:someone@microsoft.com?subject=Hello%20again">Send
  Mail</a></p>
  <p><b>Note:</b> Spaces between words should be replaced by %20
  to <b>ensure</b> that the browser will display your text
  properly.</p>
</body>
</html>
```

Web Organization LAB

This is a mail link: [Send Mail](#)

Note: Spaces between words should be replaced by %20 to ensure that the browser will display your text properly.

The following example demonstrates a more complicated mailto: link. This link not only generates a new e-mail, it adds a cc, bcc, a subject line, and the message body.

```
<html>
<body>
  <p>This is another mailto link:
  <a
    href="mailto:someone@microsoft.com?cc=someoneelse@microsoft.com&bcc=
    andsomeoneelse2@microsoft.com&subject=Summer%20Party&body=You%20are%
    20invited%20to%20a%20big%20summer%20party!">Send mail!</a></p>
  <p><b>Note:</b> Spaces between words should be replaced by %20to
  <b>ensure</b> that the browser will display your text properly.</p>
</body>
</html>
```

This is another mailto link: [Send mail!](#)

Note: Spaces between words should be replaced by %20 to ensure that the browser will display your text properly.

Creating an Image Link

The following example demonstrates how to use an image as a link. Click on the image to go to the linked page.

```
<html>
<body>
  <p>Create a link attached to an image:<a href="default.htm">
  
  </a></p>
  <p>No border around the image, but still a link:
  <a href="default.htm">
  </a></p>
</body>
</html>
```

Create a link attached to an image:



No border around the image, but still a link:





HTML Images

img Tag and the src Attribute

- ✦ In HTML, images are defined with the `` tag.
- ✦ The **img** tag is empty, which means that it contains attributes only and it has no closing tag.
- ✦ To display an image on a page, you need to use the **src** attribute. **src** stands for “source”. The value of the **src** attribute is the URL of the image you want to display on your page.
- ✦ The syntax of defining an image:

```

```

The URL points to the location or address where the image is stored. An image file named "boat.gif" located in the directory "images" on "www.w3schools.com" has the URL:

```
http://www.w3schools.com/images/boat.gif
```

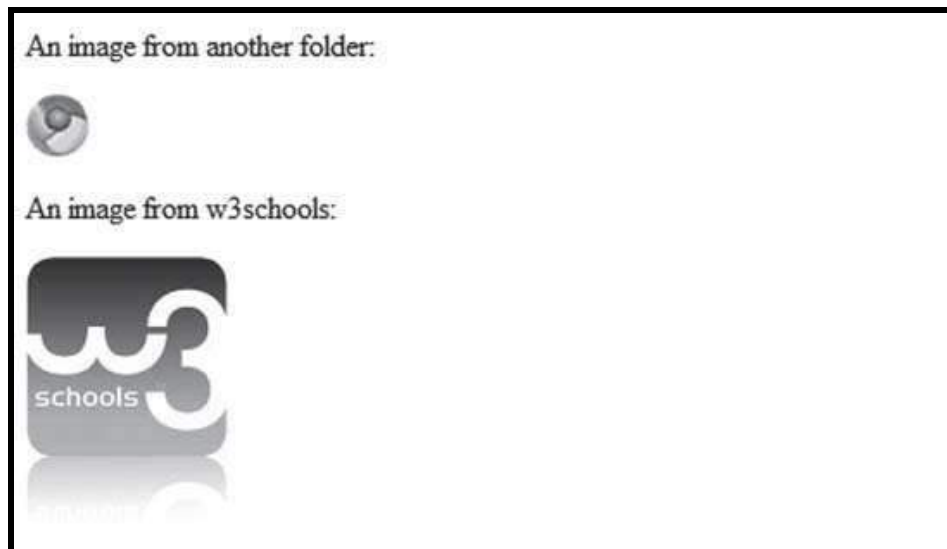
```
<html>
<body>
  <p>
    An image:
    
  </p>
</body>
</html>
```



Insert Images from Different Locations

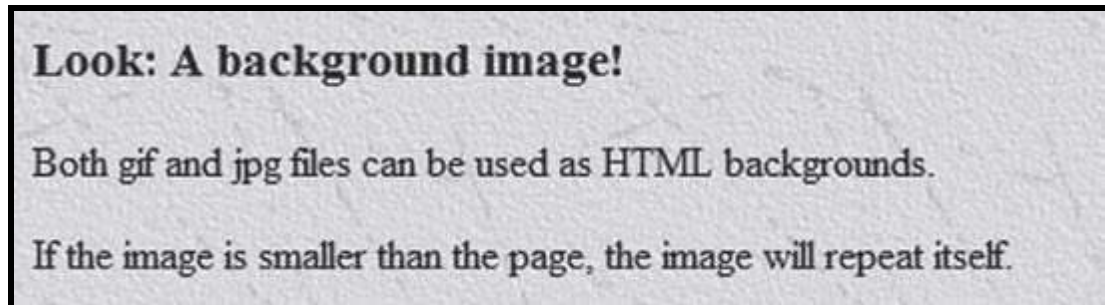
```
<html>
<body>
  <p>An image from another folder:</p>
  
  <p>An image from w3schools:</p>
  
</body>
</html>
```

Web Organization LAB



Background Images


```
<html>
<body background="background.jpg">
  <h3>Look: A background image!</h3>
  <p>Both gif and jpg files can be used as HTML backgrounds.</p>
  <p>If the image is smaller than the page, the image will repeat
  itself.</p>
</body>
</html>
```





Aligning Images

```
<html>
<body>
  <p>The text is aligned with the image
  
  at the bottom.</p>
  <p>The text is aligned with the image
  
  in the middle.</p>
  <p>The text is aligned with the image
  
  at the top.</p>
  <p><b>Note:</b> The bottom alignment is the default!</p>
</body>
</html>
```

Web Organization LAB


The text is aligned with the image  at the bottom.


The text is aligned with the image  in the middle.


The text is aligned with the image  at the top.

Note: The bottom alignment is the default!

```
<html>
<body>
  <p>This image appears
  
  exactly where it is placed in the code.</p>
  <p>
  This image appears exactly where it is placed in the code.</p>
  <p>This image appears exactly where it is placed in the code.
  </p>
</body>
</html>
```

This image appears  exactly where it is placed in the code.

 This image appears exactly where it is placed in the code.

This image appears exactly where it is placed in the code. 

Floating Images

```
<html>
<body>
  <p>
    
```

Web Organization LAB

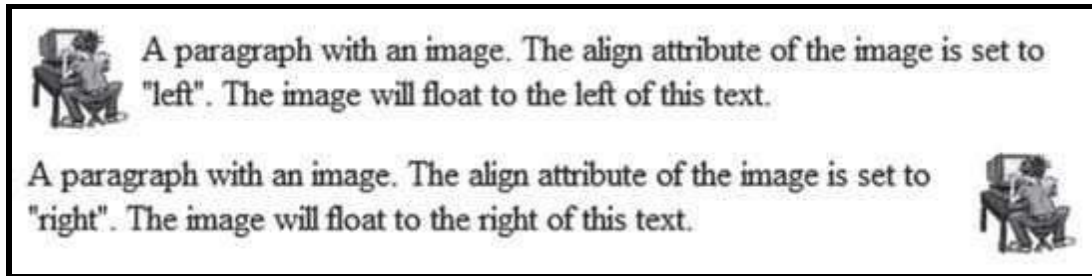
A paragraph with an image. The align attribute of the image is set to "left". The image will float to the left of this text.</p>

<p>

A paragraph with an image. The align attribute of the image is set to "right". The image will float to the right of this text.</p>

</body>

</html>



Adjusting Image Sizes

- ✦ The **width** and **height** attributes allow the page to render properly and more efficiently before the image is downloaded. Without them, the page will render once, then re-render when each image is loaded.
- ✦ The image will be scaled to fit the stated height and width. Sometimes this can have a desired effect, other times it's disastrous.

<html>

<body>

<p></p>

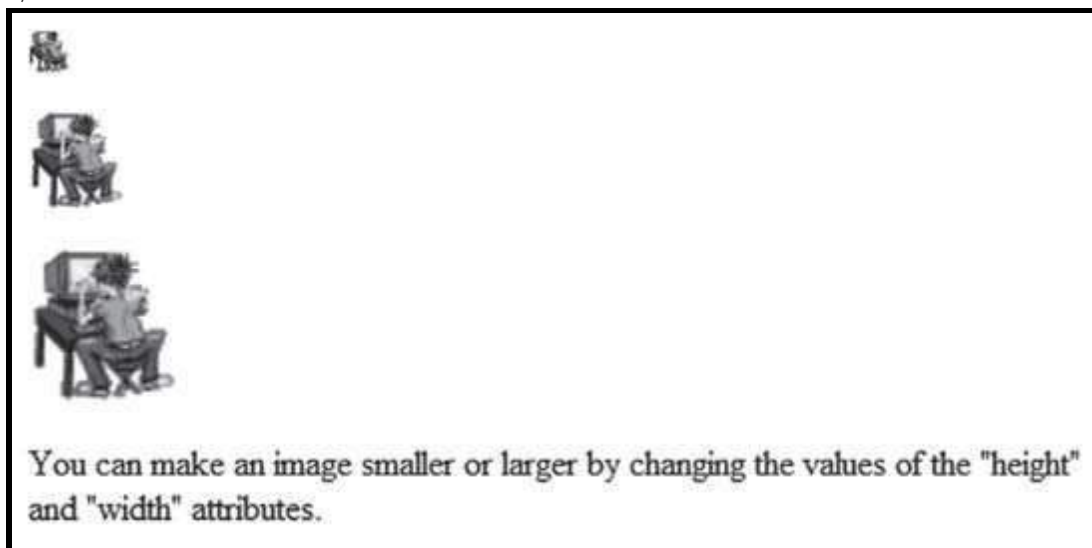
<p></p>

<p></p>

<p>You can make an image smaller or larger by changing the values of the height and width attributes.</p>

</body>

</html>

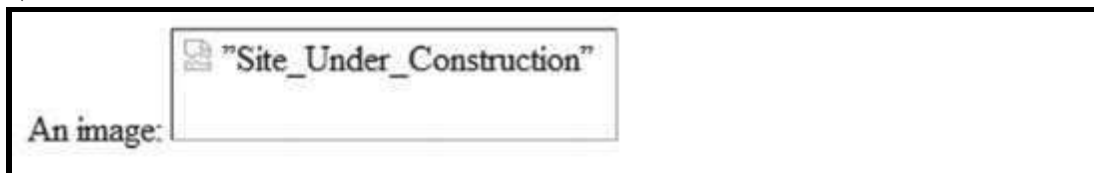


Web Organization LAB

alt Attribute

- ★ The **alt** attribute is used to define an alternate text for an image. The **alt** attribute tells the reader what he or she is missing on a page if the browser can't load images.
- ★ The browser will then display the alternate text instead of the image.
- ★ The value of the **alt** attribute is an author-defined text:
``
- ★ It is a good practice to include alternate text for every image on a page to improve the display and usefulness of your document for people who have text-only browsers.

```
<html>
<body>
<p>An image: </p>
</body>
</html>
```

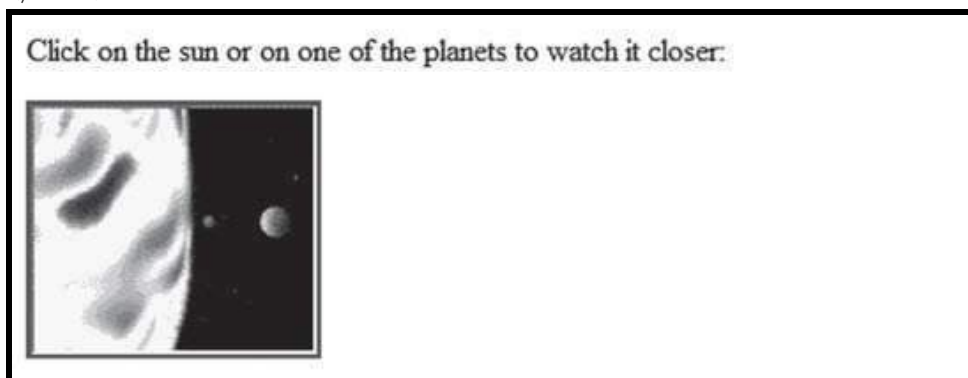


Creating an Image Map

The following example demonstrates how to create an image map with clickable regions.

```
<html>
<body>
<p>Click on the sun or on one of the planets to watch it closer:</p>

<map name="planetmap">
<area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm" />
<area shape="circle" coords="90,58,3" alt="Mercury" href=
"mercur.htm" />
<area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm"
/>
</map>
</body>
</html>
```



Web Organization LAB

HTML Tables

Creating HTML Tables

Tables are an excellent way to organize and display information on a page. Tables are defined using the `<table>` tag.

A table is divided into rows with the `<tr>` tag, and each row is divided into data cells using the `<td>` tag. The letters **td** stand for “table data,” which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, and so on.

Example 1:

```
<html>
<body>
    <h4>One column:</h4>
    <table border="1">
        <tr>
            <td>100</td>
        </tr>
    </table>
</body>
</html>
```

Example 2:

```
<html>
<body>
    <table border="1">
        <tr>
            <td>100</td>
            <td>200</td>
            <td>300</td>
        </tr>
    </table>
</body>
</html>
```

Example 3:

```
<html>
<body>
    <table border="1">
        <tr>
            <td>100</td>
            <td>200</td>
            <td>300</td>
        </tr>
        <tr>
            <td>400</td>
            <td>500</td>
        </tr>
    </table>
</body>
</html>
```

Web Organization LAB

```
        <td>600</td>
      </tr>
    </table>
  </body>
</html>
```

One column

100

The following code creates a table with one row and three columns.

100	200	300
-----	-----	-----

The following code creates a table with two rows and three columns.

100	200	300
400	500	600

Table Borders

The **border** attribute controls the appearance of the table's borders or lines. The default border is 0, so if you do not specify a **border** attribute, the table is displayed without any borders. Sometimes this is useful, but most of the time, you want the borders to be visible.

```
<html>
<body>
  <h4>With a normal border:</h4>
  <table border="1">
    <tr>
      <td>First</td>
      <td>Row</td>
    </tr>
    <tr>
      <td>Second</td>
      <td>Row</td>
    </tr>
  </table>
  <h4>With a thick border:</h4>
  <table border="8">
    <tr>
      <td>First</td>
      <td>Row</td>
    </tr>
    <tr>
```

Web Organization LAB

```
<td>Second</td>
<td>Row</td>
</tr>
</table>
<h4>With a very thick border:</h4>
<table border="15">
  <tr>
    <td>First</td>
    <td>Row</td>
  </tr>
  <tr>
    <td>Second</td>
    <td>Row</td>
  </tr>
</table>
</body>
</html>
```

With a normal border:

First	Row
Second	Row

With a thick border:

First	Row
Second	Row

With a very thick border:

First	Row
Second	Row



HTML Table's properties

Table with No Border

If you don't provide a border attribute, the default is none.

```
<html>
<body>
<h4>This table has no borders:</h4>
  <table>
    <tr>
      <td>100</td>
      <td>200</td>
      <td>300</td>
    </tr>
    <tr>
      <td>400</td>
      <td>500</td>
      <td>600</td>
    </tr>
  </table>
<h4>This table also has no borders:</h4>
  <table border="0">
    <tr>
      <td>100</td>
      <td>200</td>
      <td>300</td>
    </tr>
    <tr>
      <td>400</td>
      <td>500</td>
      <td>600</td>
    </tr>
  </table>
</body>
</html>
```

This table has no borders:

100 200 300

400 500 600

This table also has no borders:

100 200 300

400 500 600

Web Organization LAB

Headings in a Table

Table headings are defined with the <th> tag.

```
<html>
<body>
<table border="1">
  <tr>
    <th>Heading</th>
    <th>Another Heading</th>
  </tr>
  <tr>
    <td>row 1, cell 1</td>
    <td>row 1, cell 2</td>
  </tr>
  <tr>
    <td>row 2, cell 1</td>
    <td>row 2, cell 2</td>
  </tr>
</table>
<h4>Vertical headers:</h4>
<table border="1">
  <tr>
    <th>First Name:</th>
    <td>Bill Gates</td>
  </tr>
  <tr>
    <th>Telephone:</th>
    <td>555 777 1854</td>
  </tr>
  <tr>
    <th>Telephone:</th>
    <td>555 777 1855</td>
  </tr>
</table>
</body>
</html>
```

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Vertical headers:

First Name:	Bill Gates
Telephone:	555 777 1854
Telephone:	555 777 1855

Empty Cells in a Table

Table cells with no content do not display very well in most browsers. Notice that the borders around the empty table cell are missing (except when using Mozilla Firefox).

To avoid this, add a nonbreaking space () to empty data cells to ensure the borders are visible.

```
<html>
<body>
<table border="1">
  <tr>
```

```
<html>
<body>
<table border="1">
  <tr>
```

Web Organization LAB

```
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td></td>
</tr>
</table>
</body>
</html>
```

row 1, cell 1	row 1, cell 2
row 2, cell 1	

```
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>&nbsp;</td>
</tr>
</table>
</body>
</html>
```

row 1, cell 1	row 1, cell 2
row 2, cell 1	

TIP The `<thead>`, `<tbody>`, and `<tfoot>` elements are seldom used, because of bad browser support. Expect this to change in future versions of HTML.

Table with a Caption

The following example demonstrates how to create a table with a caption.

```
<html>
<body>
<h4>This table has a caption, and a thick border:</h4>
<table border="6">
<caption>My Caption</caption>
<tr>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>400</td>
<td>500</td>
<td>600</td>
</tr>
</table>
</body>
</html>
```

This table has a caption, and a thick border:		
My Caption		
100	200	300
400	500	600

Cells Spanning Multiple Columns

In this example, you learn how to define table cells that span more than one row or one column.

```
<html>
<body>
<h4>Cell that spans two columns:</h4>
<table border="1">
<tr>
```

Web Organization LAB

```
<th>Name</th>
<th colspan="2">Telephone</th>
</tr>
<tr>
<td>Bill Gates</td>
<td>555 77 854</td>
<td>555 77 855</td>
</tr>
</table>
<h4>Cell that spans two rows:</h4>
<table border="1">
<tr>
<th>First Name:</th>
<td>Bill Gates</td>
</tr>
<tr>
<th rowspan="2">Telephone:</th>
<td>555 77 854</td>
<td>555 77 855</td>
</tr>
<tr>
<td>555 77 854</td>
<td>555 77 855</td>
</tr>
</table>
</body>
</html>
```

Cell that spans two columns:

Name	Telephone	
Bill Gates	555 77 854	555 77 855

Cell that spans two rows:

First Name:	Bill Gates	
Telephone:	555 77 854	555 77 855
	555 77 854	555 77 855

Tags Inside a Table

This example demonstrates how to display elements inside other elements.

```
<html>
<body>
<table border="1">
<tr>
<td>
<p>This is a paragraph</p>
<p>This is another paragraph</p>
</td>
<td>This cell contains a table:
<table border="1">
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
</tr>
</table>
</td>
</tr>
</table>
```

Web Organization LAB

```
</tr>
<tr>
  <td>This cell contains a list
    <ul>
      <li>apples</li>
      <li>bananas</li>
      <li>pineapples</li>
    </ul>
  </td>
  <td>HELLO</td>
</tr>
</table>
</body>
</html>
```

<p>This is a paragraph</p> <p>This is another paragraph</p>	<p>This cell contains a table:</p> <table><tr><td>A</td><td>B</td></tr><tr><td>C</td><td>D</td></tr></table>	A	B	C	D
A	B				
C	D				
<p>This cell contains a list</p> <ul style="list-style-type: none">• apples• bananas• pineapples	<p>HELLO</p>				

Cell Padding

This example demonstrates how to use cell padding to create more white space between the cell content and its borders.

```
<html>
<body>
<h4>Without cellpadding:</h4>
<table border="1">
  <tr>
    <td>First</td>
    <td>Row</td>
  </tr>
  <tr>
    <td>Second</td>
    <td>Row</td>
  </tr>
</table>
<h4>With cellpadding:</h4>
<table border="1" cellpadding="10">
  <tr>
```

Without cellpadding:	
First	Row
Second	Row
With cellpadding:	
First	Row
Second	Row

Web Organization LAB

```
<td>First</td>
<td>Row</td>
</tr>
<tr>
<td>Second</td>
<td>Row</td>
</tr>
</table>
</body>
</html>
```

Cell Spacing

This example demonstrates how to use cell spacing to increase the distance between the cells

```
<html>
<body>
<h4>Without cellpadding:</h4>
<table border="1">
  <tr>
    <td>First</td>
    <td>Row</td>
  </tr>
  <tr>
    <td>Second</td>
    <td>Row</td>
  </tr>
</table>
<h4>With cellpadding:</h4>
<table border="1" cellpadding="10">
  <tr>
    <td>First</td>
    <td>Row</td>
  </tr>
  <tr>
    <td>Second</td>
    <td>Row</td>
  </tr>
</table>
</body>
</html>
```

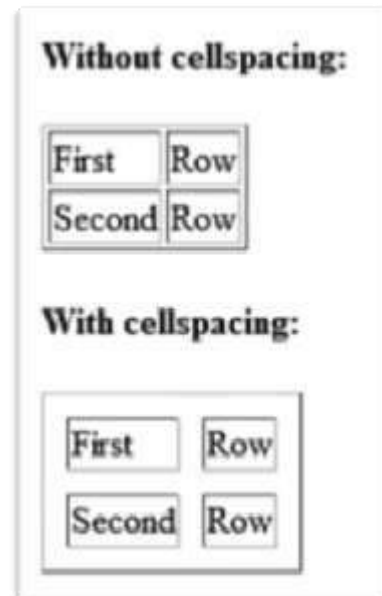


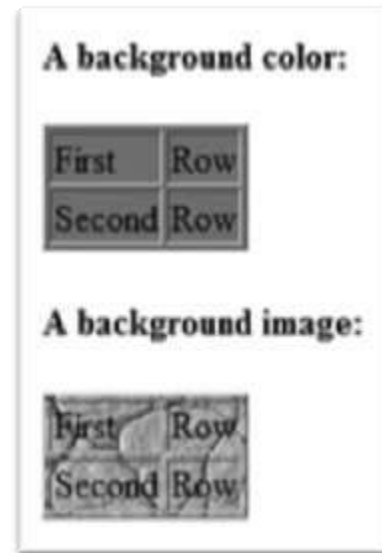
Table Background Colors and Images

This example demonstrates how to add a background to a table.

```
<html>
<body>
<h4>A background color:</h4>
<table border="1" bgcolor="gray">
  <tr>
```

Web Organization LAB

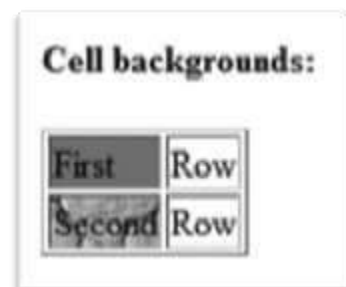
```
        <td>First</td>
        <td>Row</td>
    </tr>
    <tr>
        <td>Second</td>
        <td>Row</td>
    </tr>
</table>
<h4>A background image:</h4>
<table border="1" background="bgdesert. jpg">
    <tr>
        <td>First</td>
        <td>Row</td>
    </tr>
    <tr>
        <td>Second</td>
        <td>Row</td>
    </tr>
</table>
</body>
</html>
```



Cell Background Colors and Images

The following example demonstrates how to add a background to one or more table cells.

```
<html>
<body>
<h4>Cell backgrounds:</h4>
<table border="1">
    <tr>
        <td bgcolor="gray">First</td>
        <td>Row</td>
    </tr>
    <tr>
        <td background="bgdesert. jpg">
            Second</td>
        <td>Row</td>
    </tr>
</table>
</body>
</html>
```



Aligning Cell Content

This sample code demonstrates how to use the align attribute to align the content of cells to create a neatly organized table.

Web Organization LAB

```
<html>
<body>
<table width="400" border="1">
  <tr>
    <th align="left">Money spent on....</th>
    <th align="right">January</th>
    <th align="right">February</th>
  </tr>
  <tr>
    <td align="left">Clothes</td>
    <td align="right">$241.10</td>
    <td align="right">$50.20</td>
  </tr>
  <tr>
    <td align="left">Make-Up</td>
    <td align="right">$30.00</td>
    <td align="right">$44.45</td>
  </tr>
  <tr>
    <td align="left">Food</td>
    <td align="right">$730.40</td>
    <td align="right">$650.00</td>
  </tr>
  <tr>
    <th align="left">Sum</th>
    <th align="right">$1001.50</th>
    <th align="right">$744.65</th>
  </tr>
</table>
</body>
</html>
```

Money spent on....	January	February
Clothes	\$241.10	\$50.20
Make-Up	\$30.00	\$44.45
Food	\$730.40	\$650.00
Sum	\$1001.50	\$744.65

frame Attribute

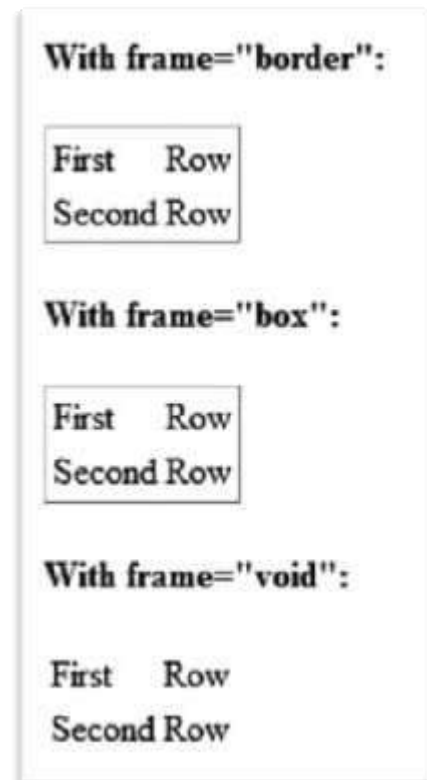
The following examples demonstrate how to use the **frame** attribute to control the borders around the table.

If you see no frames around the tables in your browser, either your browser is too old or it does not support the attribute.

Web Organization LAB

Example 1:

```
<html>
<body>
<h4>With frame="border":</h4>
<table frame="border">
  <tr>
    <td>First</td>
    <td>Row</td>
  </tr>
  <tr>
    <td>Second</td>
    <td>Row</td>
  </tr>
</table>
<h4>With frame="box":</h4>
<table frame="box">
  <tr>
    <td>First</td>
    <td>Row</td>
  </tr>
  <tr>
    <td>Second</td>
    <td>Row</td>
  </tr>
</table>
<h4>With frame="void":</h4>
<table frame="void">
  <tr>
    <td>First</td>
    <td>Row</td>
  </tr>
  <tr>
    <td>Second</td>
    <td>Row</td>
  </tr>
</table>
<html>
<body>
```



Example 2:

```
<html>
<body>
<h4>With frame="above":</h4>
<table frame="above">
  <tr>
    <td>First</td>
    <td>Row</td>
```

Web Organization LAB

```
</tr>
<tr>
    <td>Second</td>
    <td>Row</td>
</tr>
</table>
<h4>With frame="below":</h4>
<table frame="below">
    <tr>
        <td>First</td>
        <td>Row</td>
    </tr>
    <tr>
        <td>Second</td>
        <td>Row</td>
    </tr>
</table>
<h4>With frame="hsides":</h4>
<table frame="hsides">
    <tr>
        <td>First</td>
        <td>Row</td>
    </tr>
    <tr>
        <td>Second</td>
        <td>Row</td>
    </tr>
</table>
</body>
</html>
```

Example 3:

```
<html>
<body>
<h4>With frame="vsides":</h4>
<table frame="vsides">
    <tr>
        <td>First</td>
        <td>Row</td>
    </tr>
    <tr>
        <td>Second</td>
        <td>Row</td>
    </tr>
</table>
<h4>With frame="lhs":</h4>
<table frame="lhs">
    <tr>
```

With frame="above":

First Row
Second Row

With frame="below":

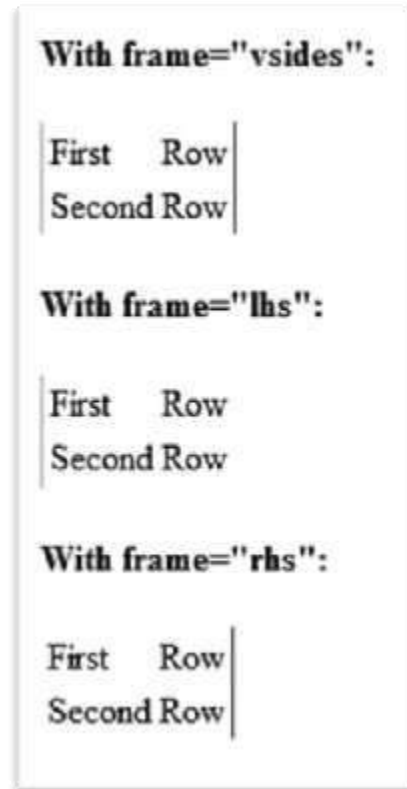
First Row
Second Row

With frame="hsides":

First Row
Second Row

Web Organization LAB

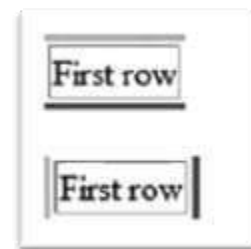
```
        <td>First</td>
        <td>Row</td>
    </tr>
    <tr>
        <td>Second</td>
        <td>Row</td>
    </tr>
</table>
<h4>With frame="rhs":</h4>
<table frame="rhs">
    <tr>
        <td>First</td>
        <td>Row</td>
    </tr>
    <tr>
        <td>Second</td>
        <td>Row</td>
    </tr>
</table>
</body>
</html>
```



Using frame and border to Control Table Borders

You can use the frame and border attributes to control the borders around the table. If you see no frames around the tables in these examples, your browser does not support the frame attribute.

```
<html>
<body>
<table frame="hsides" border="3">
    <tr>
        <td>First row</td>
    </tr>
</table>
<br />
<table frame="vsides" border="3">
    <tr>
        <td>First row</td>
    </tr>
</table>
</body>
</html>
```



Web Organization LAB

Table Tags

TAG	DESCRIPTION
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns
<col>	Defines the attribute values for one or more columns in a table
<thead>	Defines a table head
<tbody>	Defines a table body
<tfoot>	Defines a table footer



HTML Lists

Unordered Lists

An unordered list is a list of items. The list items are marked with bullets (typically small black circles). An unordered list starts with the `` tag. Each list item starts with the `` tag.

```
<html>
<body>
<h4>An Unordered List:</h4>
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
</body>
</html>
```

An Unordered List:

- Coffee
- Tea
- Milk

You can display different kinds of bullets in an unordered list by using the `type` attribute.

```
<html>
<body>
<h4>Disc bullets list:</h4>
<ul type="disc">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
</ul>
<h4>Circle bullets list:</h4>
<ul type="circle">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
</ul>
<h4>Square bullets list:</h4>
<ul type="square">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
</ul>
</body>
</html>
```

Disc bullets list:

- Apples
- Bananas
- Lemons

Circle bullets list:

- Apples
- Bananas
- Lemons

Square bullets list:

- Apples
- Bananas
- Lemons

Web Organization LAB

Ordered Lists

An ordered list is also a list of items; the list items are numbered sequentially rather than bulleted.

An ordered list starts with the `` tag. Each list item starts with the `` tag.

```
<html>
<body>
<h4>An Ordered List:</h4>
<ol>
    <li>Coffee</li>
    <li>Tea</li>
    <li>Milk</li>
</ol>
</body>
</html>
```

An Ordered List:

1. Coffee
2. Tea
3. Milk

Different Types of Ordering

You can display different kinds of ordered lists by using the type attribute.

Example 1:

```
<html>
<body>
<h4>Letters list:</h4>
<ol type="A">
    <li>Apples</li>
    <li>Bananas</li>
    <li>Lemons</li>
</ol>
<h4>Lowercase letters list:</h4>
<ol type="a">
    <li>Apples</li>
    <li>Bananas</li>
    <li>Lemons</li>
</ol>
</body>
</html>
```

Letters list:

- A. Apples
- B. Bananas
- C. Lemons

Lowercase letters list:

- a. Apples
- b. Bananas
- c. Lemons

Example 2:

```
<html>
<body>
<h4>Roman numbers list:</h4>
<ol type="I">
    <li>Apples</li>
    <li>Bananas</li>
    <li>Lemons</li>
</ol>
```

Roman numbers list:

- I. Apples
- II. Bananas
- III. Lemons

Lowercase Roman numbers list:

- i. Apples
- ii. Bananas
- iii. Lemons

Web Organization LAB

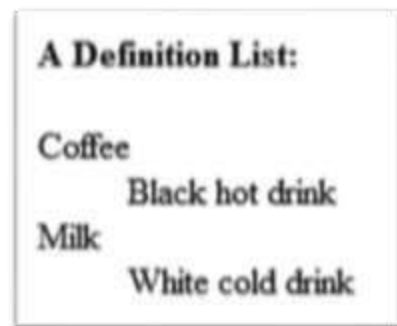
```
</ol>
<h4>Lowercase Roman numbers list:</h4>
<ol type="i">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
</ol>
</body>
</html>
```

Definition Lists

A definition list is not a list of single items. It is a list of items (terms), together with a description of each item (term).

- A definition list starts with a <dl> tag (definition list).
- Each term starts with a <dt> tag (definition term).
- Each description starts with a <dd> tag (definition description). Inside the <dd> tag you can put paragraphs, line breaks, images, links, other lists, and so on.

```
<html>
<body>
<h4>A Definition List:</h4>
<dl>
  <dt>Coffee</dt>
  <dd>Black hot drink</dd>
  <dt>Milk</dt>
  <dd>White cold drink</dd>
</dl>
</body>
</html>
```



Nested Lists

A nested list is a list within another list. Usually the second list is indented another level and the item markers will appear differently than the original list.

```
<html>
<body>
<h4>A nested List:</h4>
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```



Web Organization LAB

```
</ul>
</body>
</html>
```

Nested lists can be several levels deep.

```
<html>
<body>
<h4>A nested List:</h4>
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea
        <ul>
          <li>China</li>
          <li>Africa</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
</body>
</html>
```



List Tags

TAG	DESCRIPTION
	Defines an ordered list
	Defines an unordered list
	Defines a list item
<dl>	Defines a definition list
<dt>	Defines a term (an item) in a definition list
<dd>	Defines a description of a term in a definition list
<dir>	Deprecated. Use instead
<menu>	Deprecated. Use instead



HTML Forms & Input

Forms

HTML forms are used to collect different kinds of user input. A form is an area that can contain form elements.

Form elements are elements that allow the user to enter information in a form (like text fields, text area fields, drop-down menus, radio buttons, check boxes, and so on).

A form is defined with the `<form>` tag:

```
<form>  
    .  
    input elements  
    .  
</form>
```

A screenshot of a web form. At the top, it says "This form sends an e-mail to w3schools." Below this, there are three input fields: "Name:" with a text box containing "yourname", "Mail:" with a text box containing "yourmail", and "Comment:" with a larger text box containing "yourcomment". At the bottom, there are two buttons: "Send" and "Reset".

input Tag and Attributes

The most used form tag is the `<input>` tag. The type of input is specified with the type attribute. The following types are the most commonly used input types.

Web Organization LAB

- **Text Fields**

Text fields are used when you want the user to type letters, numbers, and so on in a form. Note that the form itself is not visible.

```
<html>
<body>
<form action="">
  First name:
  <input type="text" name="firstname" />
  <br />
  Last name:
  <input type="text" name="lastname" />
</form>
</body>
</html>
```

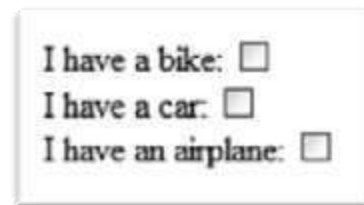


TIP : In most browsers, the width of the text field is 20 characters by default.

- **Check Boxes**

A user can select or deselect a check box.

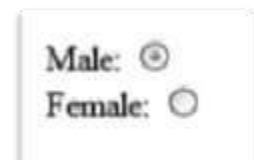
```
<html>
<body>
<form action="">
  I have a bike:
  <input type="checkbox" name="vehicle" value="Bike">
  <br />
  I have a car:
  <input type="checkbox" name="vehicle" value="Car">
  <br />
  I have an airplane:
  <input type="checkbox" name="vehicle" value="Airplane">
</form>
</body>
</html>
```



- **Radio Buttons**

When a user clicks a radio button, that button becomes selected, and all other buttons in the same group become deselected.

```
<html>
<body>
<form action="">
  Male:
  <input type="radio" checked="checked" name="Sex" value="male">
  <br>
  Female:
  <input type="radio" name="Sex" value="female">
</form>
```



Web Organization LAB

```
</form>
</body>
</html>
```

- **Drop-Down List**

The next example shows how to create a simple drop-down list on an HTML page. A drop-down list is a selectable list.

```
<html>
<body>
<form action="">
  <select name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
</form>
</body>
</html>
```



You can also display a simple drop-down list with a value preselected on the list.

```
<html>
<body>
<form action="">
  <select name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat" selected="selected">Fiat</option>
    <option value="audi">Audi</option>
  </select>
</form>
</body>
</html>
```

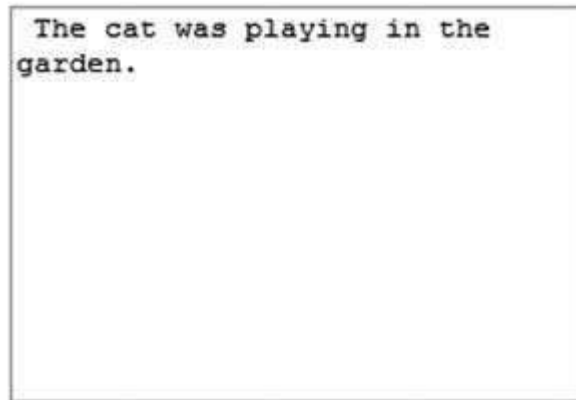


- **Text Area**

Using a textarea (a multiline text input control), you can write an unlimited number of characters. A textarea can be in a form or elsewhere on a page.

```
<html>
<body>
<textarea rows="10" cols="30"> The cat was playing in the garden.
</textarea>
</body>
</html>
```

Web Organization LAB



The cat was playing in the garden.

- **Buttons**

Buttons are common items on a form. This example demonstrates how to create a button. You can define your own text on the face of the button.

```
<html>
<body>
<form action="">
  <input type="button" value="Hello world!">
</form>
</body>
</html>
```



- **Fieldset**

A fieldset is a grouping of data fields. This example demonstrates how to draw a border with a caption around your data

```
<html>
<body>
<fieldset>
  <legend> Health information: </legend>
  <form action="">
    Height <input type="text" size="3">
    Weight <input type="text" size="3">
  </form>
</fieldset>
```

<p> If there is no border around the input form, your browser is too old.</p>

```
</body>
</html>
```



Health information:

Height Weight

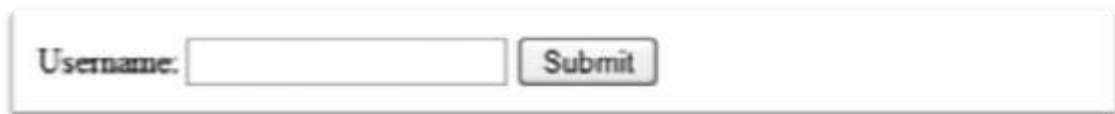
If there is no border around the input form, your browser is too old.

Web Organization LAB

action Attribute

When the user clicks the Submit button, the content of the form is sent to the server. The form's action attribute defines the name of the file to send the content to. The file defined in the action attribute usually does something with the received input.

```
<form name="input" action="html_form_submit.asp" method="get">
  Username:
  <input type="text" name="user" />
  <input type="submit" value="Submit" />
</form>
```

A screenshot of a web form. It consists of a label 'Username:' followed by a text input field. To the right of the input field is a button labeled 'Submit'.

If you type some characters in the text field and click the Submit button, the browser sends your input to a page called "html_form_submit.asp". The page will show you the received input.

Form Examples

Example 1: This example demonstrates how to add a form to a page. The form contains two input fields and a Submit button.

```
<html>
<body>
<form name="input" action="html_form_action.asp" method="get">
  Type your first name:
  <input type="text" name="FirstName" value="Mickey" size="20">
  <br>Type your last name:
  <input type="text" name="LastName" value="Mouse" size="20"><br>
  <input type="submit" value="Submit">
</form>
<p>If you click the "Submit" button, you will send your input to a new page
called html_form_action.asp.</p>
</body>
</html>
```

A screenshot of a web form. It has two text input fields. The first field is preceded by the label 'Type your first name:' and contains the text 'Mickey'. The second field is preceded by the label 'Type your last name:' and contains the text 'Mouse'. Below these fields is a button labeled 'Submit'. Below the button is a paragraph of text: 'If you click the "Submit" button, you will send your input to a new page called html_form_action.asp.'

Web Organization LAB

Example 2 – Form with Check Boxes: The following form contains three check boxes and a Submit button.

```
<html>
<body>
<form name="input" action="html_form_action.asp" method="get">
    I have a bike:
    <input type="checkbox" name="vehicle" value="Bike" checked="checked"
    />
    <br />
    I have a car:
    <input type="checkbox" name="vehicle" value="Car" />
    <br />
    I have an airplane:
    <input type="checkbox" name="vehicle" value="Airplane" />
    <br /><br />
    <input type="submit" value="Submit" />
</form>
<p>
If you click the "Submit" button, you send your input to a new page called
html_form_action.asp.
</p>
</body>
</html>
```

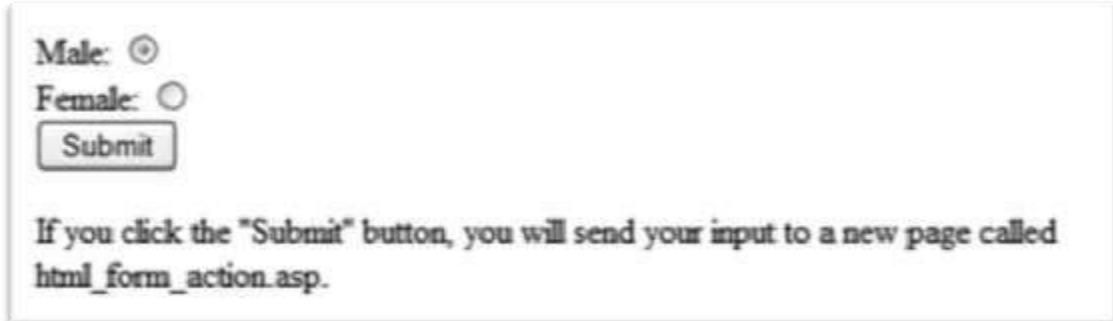
A screenshot of a web browser displaying a form. The form contains three lines of text, each followed by a checkbox: "I have a bike: [checked]", "I have a car: [unchecked]", and "I have an airplane: [unchecked]". Below these is a "Submit" button. At the bottom of the form, there is a paragraph of text: "If you click the 'Submit' button, you send your input to a new page called html_form_action.asp."

Example 3 – Form with Radio Buttons: This example displays a form with two radio buttons and a Submit button.

```
<html>
<body>
<form name="input" action="html_form_action.asp" method="get">
    Male:
    <input type="radio" name="Sex" value="Male" checked="checked"><br>
    Female:
    <input type="radio" name="Sex" value="Female"><br>
    <input type="submit" value="Submit">
</form>
```

Web Organization LAB

<p>If you click the "Submit" button, you will send your input to a new page called html_form_action.asp.</p>
</body>
</html>

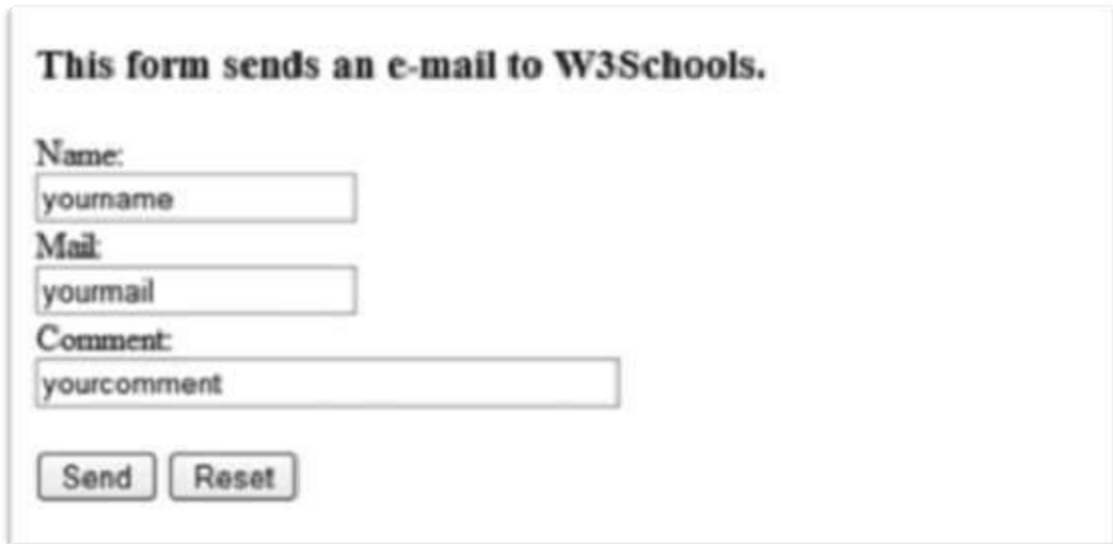


A screenshot of a web form. It contains two radio buttons: "Male:" with a selected radio button (indicated by a dot inside the circle) and "Female:" with an unselected radio button. Below the radio buttons is a "Submit" button. Underneath the button is a paragraph of text: "If you click the 'Submit' button, you will send your input to a new page called html_form_action.asp."

Example 4 -Send E-Mail from a Form: The next example demonstrates how to send e-mail from a form.

```
<html>
<body>
<form   action="MAILTO:someone@w3schools.com"   method="post"   enctype=
"text/plain">

</form>
</body>
</html>
```



A screenshot of a web form titled "This form sends an e-mail to W3Schools." The form contains three input fields: "Name:" with the placeholder text "yourname", "Mail:" with the placeholder text "yourmail", and "Comment:" with the placeholder text "yourcomment". Below the input fields are two buttons: "Send" and "Reset".

Web Organization LAB

Form Tags

TAG	DESCRIPTION
<form>	Defines a form for user input
<input>	Defines an input field
<textarea>	Defines a textarea (a multiline text input control)
<label>	Defines a label to a control
<fieldset>	Defines a fieldset
<legend>	Defines a caption for a fieldset
<select>	Defines a selectable list (a drop-down box)
<optgroup>	Defines an option group
<option>	Defines an option in the drop-down box
<button>	Defines a pushbutton
<isindex>	Deprecated. Use <input> instead



HTML Layout Using Tables

<p>A part of this page is formatted with two columns, like a newspaper page.</p> <p>As you can see on this page, there is a left column and a right column.</p> <p>This text is displayed in the left column.</p>	<p>An HTML <code><table></code> is used to divide a part of this Web page into two columns.</p> <p>The trick is to use a table without borders, and maybe a little extra cell padding.</p> <p>No matter how much text you add to this table, it will stay inside its column borders.</p>
---	--

Dividing a part of an HTML page into table columns is very easy to do. Just set it up like the following example.

```
<html>
<body>
<table border="0" width="100%" cellpadding="10">
  <tr>
    <td width="50%" valign="top"> This is some text. This is some
      text. This is some text. This is some text. This is some text.
    </td>
    <td width="50%" valign="top"> Another text. Another text.
      Another text. Another text. Another text. Another text.
      Another text.
    </td>
  </tr>
</table>
</body>
</html>
```

This is some text. This is some text.	Another text. Another text.
This is some text. This is some text.	Another text. Another text.
This is some text.	Another text. Another text.
	Another text.

Web Organization LAB

HTML Frames

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

- The web developer must keep track of more HTML documents.
- It is difficult to print the entire page.
- Users often dislike them.
- It presents linking challenges.
- People often use frames to wrap their own ads and branding around other people's content

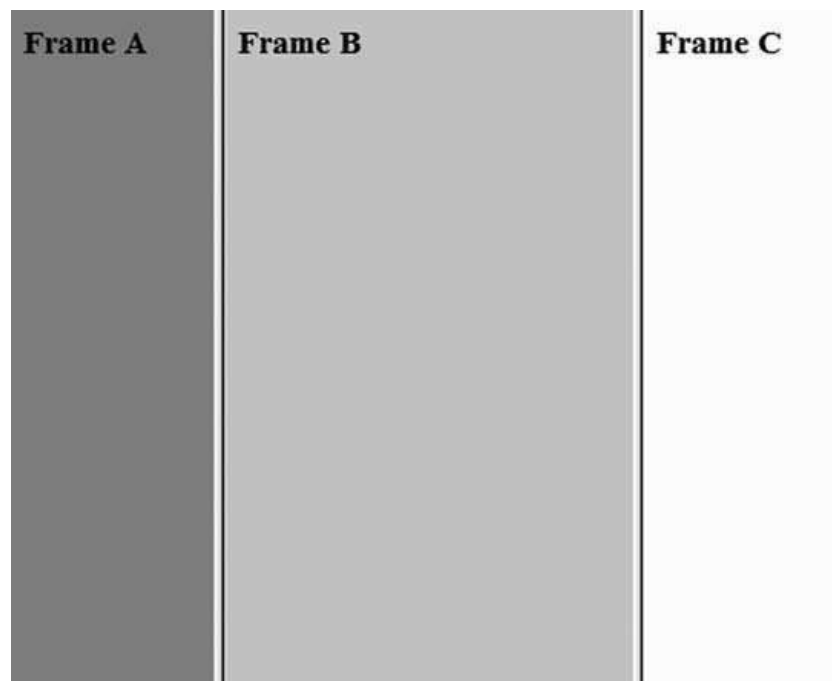
frameset Tag

The `<frameset>` tag defines how to divide the window into frames. Each frameset defines a set of rows or columns. The values of the rows/columns indicate the amount of screen area each row/column will occupy.

Vertical Frameset

The following example demonstrates how to make a vertical frameset with three different documents.

```
<html>
<frameset cols="25%, 50%, 25%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
  <frame src="frame_c.htm">
</frameset>
</html>
```



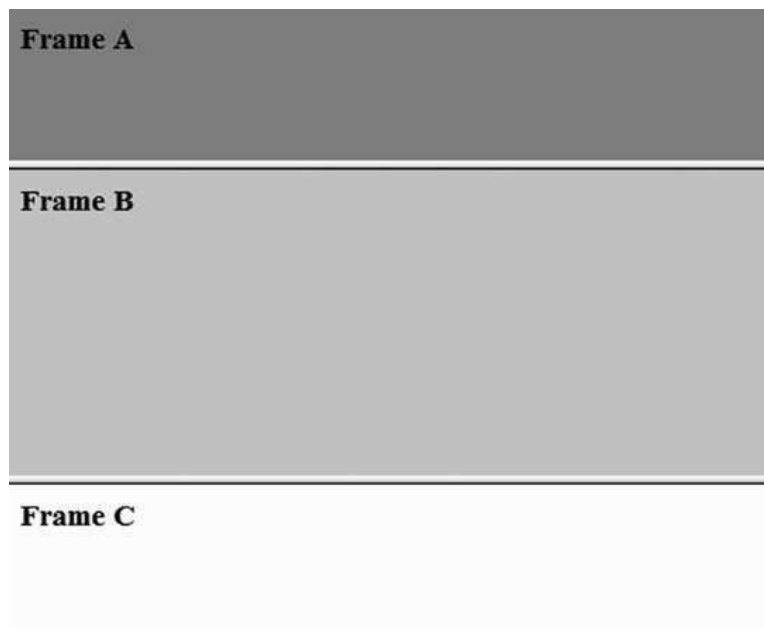
Web Organization LAB

NOTE: Note that the code does not use the `<body>` tag when a `<frameset>` tag is in use.

Horizontal Frameset

The following example demonstrates how to make a horizontal frameset with three different documents.

```
<html>
<frameset rows="25%, 50%, 25%">
    <frame src="frame_a.htm">
    <frame src="frame_b.htm">
    <frame src="frame_c.htm">
</frameset>
</html>
```



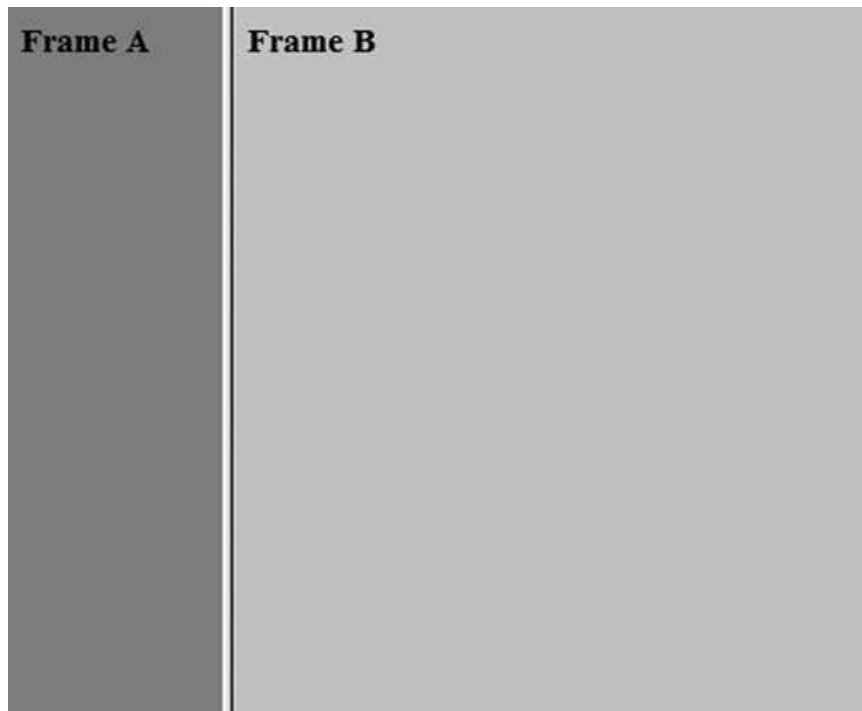
frame Tag

The `<frame>` tag defines which HTML document will initially open in each frame.

In the following example, we have a frameset with two columns. The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The HTML document `frame_a.htm` is put into the first column, and the HTML document `frame_b.htm` is put into the second column.

```
<frameset cols="25%, 75%">
    <frame src="frame_a.htm">
    <frame src="frame_b.htm">
</frameset>
```

Web Organization LAB



N O T E: The frameset column size value can also be set in pixels (`cols="200,500"`), and one of the columns can be set to use the remaining space (`cols="25%,*"`).

Designing with Frames

If a frame has visible borders, the user can resize it by dragging the border. To prevent a user from doing this, you can add `noresize="noresize"` to the `<frame>` tag.

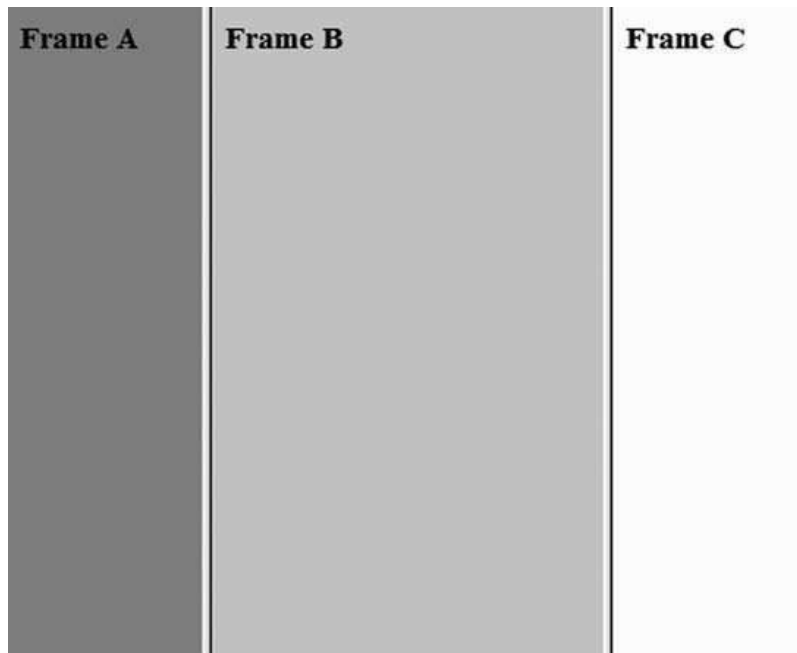
noframes Tag

Although they are less common these days, it's a good idea to add the `<noframes>` tag for older or text-based browsers that do not support frames. You cannot use the `<body>` tags together with the `<frameset>` tags. However, if you add a `<noframes>` tag containing some text for browsers that do not support frames, you will have to enclose the text in `<body>` tags. See how it is done in the following example.

```
<html>
<frameset cols="25%, 50%, 25%">
  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
  <frame src="frame_c.htm">
  <noframes>
    <body>Your browser does not handle frames!</body>
  </noframes>
</frameset>
</html>
```

(This browser supports frames, so the `noframes` text remains invisible.)

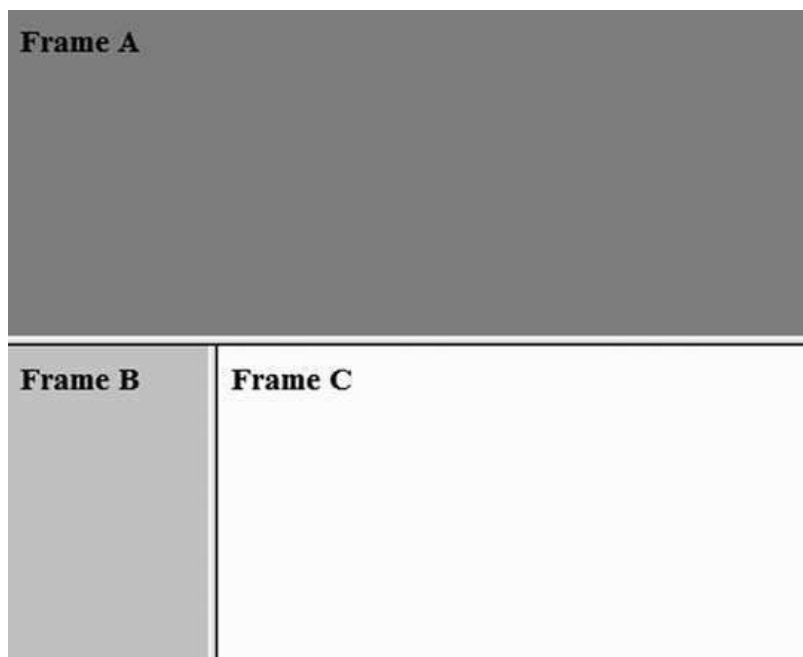
Web Organization LAB



Mixed Frameset

The following example demonstrates how to make a frameset with three documents and how to mix them in rows and columns.

```
<html>
<frameset rows="50%, 50%">
  <frame src="frame_a.htm">
  <frameset cols="25%, 75%">
    <frame src="frame_b.htm">
    <frame src="frame_c.htm">
  </frameset>
</frameset>
</html>
```



Web Organization LAB

noresize Attribute

This example demonstrates the noresize attribute. The frames are not resizable.

Unlike other frames, if you move the mouse over the borders between the frames, you will notice that you cannot drag or move the frame borders.

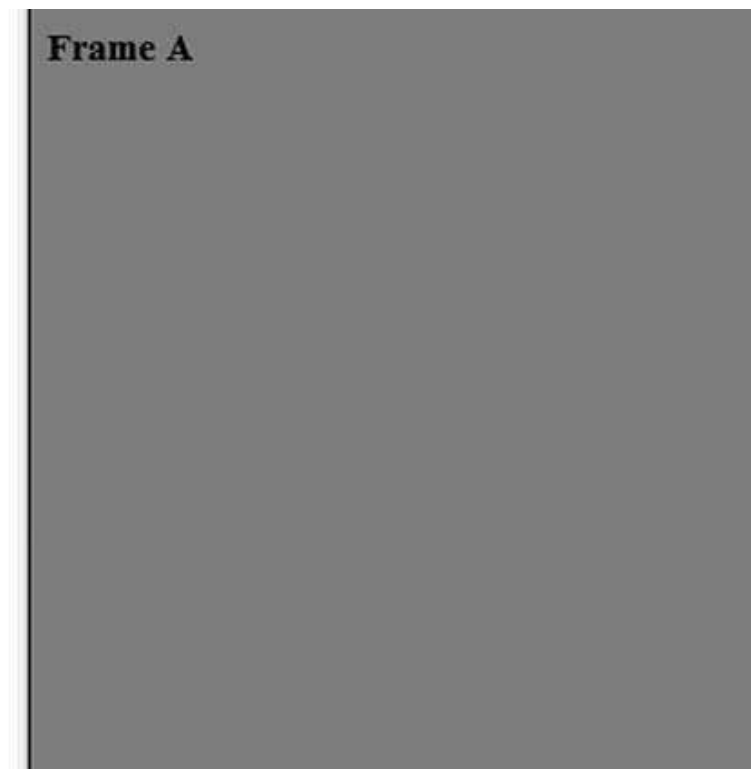
```
<html>
<frameset rows="50%, 50%">
  <frame noresize="noresize" src="frame_a.htm">
  <frameset cols="25%, 75%">
    <frame noresize="noresize" src="frame_b.htm">
    <frame noresize="noresize" src="frame_c.htm">
  </frameset>
</frameset>
</html>
```

Navigation Frame

This example demonstrates how to make a navigation frame. A navigation frame contains a list of links with the second frame as the target. The second frame will show the linked document.

```
<html>
<frameset cols="120, *">
  <frame src="tryhtml_contents.htm">
  <frame src="frame_a.htm" name="showframe">
</frameset>
</html>
```

[Frame a](#)
[Frame b](#)
[Frame c](#)



Web Organization LAB

In the first column, the file called tryhtml_contents.htm contains links to three documents on the w3schools.com Web site. The source code for the links:

```
<a href = "frame_a.htm" target = "showframe">Frame a</a><br>
<a href = "frame_b.htm" target = "showframe">Frame b</a><br>
<a href = "frame_c.htm" target = "showframe">Frame c</a>
```

Inline Frame

Frames can also be used within a single HTML page. These are known as inline frames. The following example demonstrates how to create an inline frame like the one that appears in Figure below.

```
<html>
<body>
<iframe src="default.asp"></iframe>
<p>Some older browsers don't support iframes.</p>
<p>If they don't, the iframe will not be visible.</p>
</body>
</html>
```



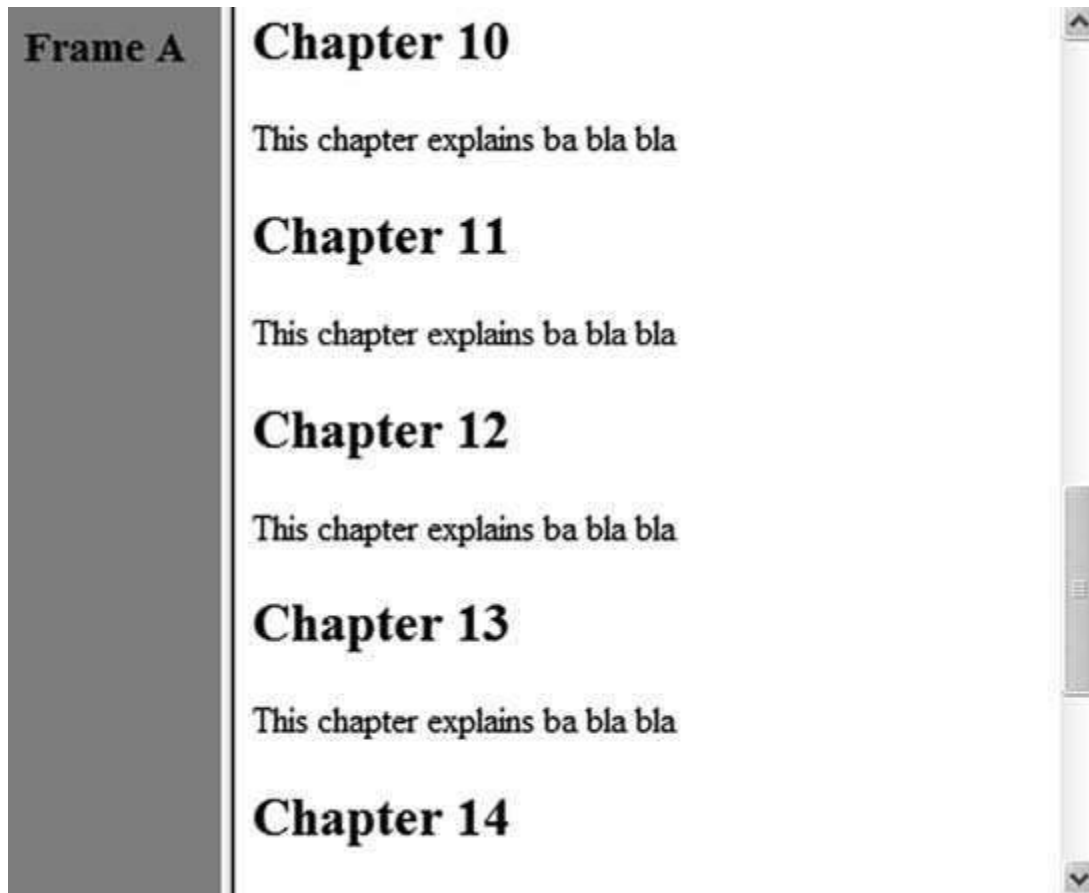
Some older browsers don't support iframes.

If they don't, the iframe will not be visible.

Jump to

This example demonstrates two frames. One of the frames has a source to a specified section in a file. The specified section is made with in the link.htm file.

```
<html>
<frameset cols="20%, 80%">
  <frame src="frame_a.htm">
  <frame src="link.htm#C10">
</frameset>
</html>
```



Jump to a Specified Section

This example demonstrates two frames. The navigation frame (content.htm) to the left contains a list of links with the second frame (link.htm) as a target on the right.

The second frame shows the linked document.

```
<html>
<frameset cols="180,*">
  <frame src="content.htm">
  <frame src="link.htm" name="showframe">
</frameset>
</html>
```


Web Organization LAB

[Link without Anchor](#)
[Link with Anchor](#)

Chapter 1

This chapter explains ba bla bla

Chapter 2

This chapter explains ba bla bla

Chapter 3

This chapter explains ba bla bla

Chapter 4

This chapter explains ba bla bla

Chapter 5

One of the links in the navigation frame is linked to a specified section in the target file. The HTML code in the file content.htm looks like this:

```
<a href = "link.htm" target = "showframe">Link without Anchor</a><br><a href = "link.htm#C10" target = "showframe">Link with Anchor</a>.
```

Frame Tags

TAG	DESCRIPTION
<frameset>	Defines a set of frames
<frame>	Defines a sub window (a frame)
<noframes>	Defines a noframe section for browsers that do not handle frames
<iframe>	Defines an inline sub window (frame)



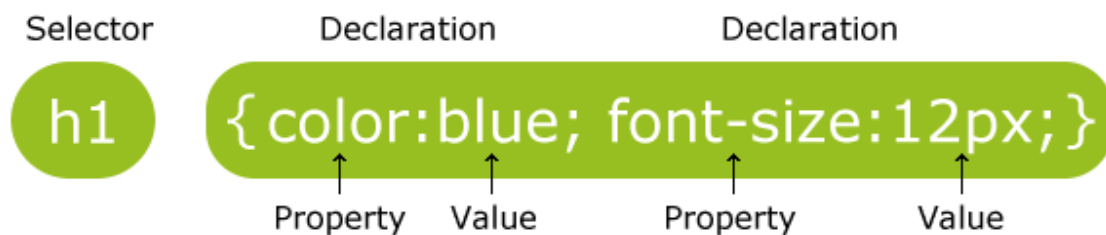
CSS Basic

1. What is CSS?

- CSS stands for Cascading Style Sheets
- Styles define how to display HTML elements
- Styles were added to HTML 4.0 to solve a problem
- External Style Sheets can save a lot of work
- External Style Sheets are stored in CSS files

2. CSS Syntax

A CSS rule has two main parts: a selector, and one or more declarations:



- The selector is normally the HTML element you want to style.
- Each declaration consists of a property and a value.
- The property is the style attribute you want to change. Each property has a value.

3. CSS Example

CSS declarations always ends with a semicolon, and declaration groups are surrounded by curly brackets:

```
p {color:red;text-align:center;}
```

To make the CSS more readable, you can put one declaration on each line, like this:

```
p {  
  color:red;  
  text-align:center;  
}
```

Web Organization LAB

4. CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers. A CSS comment begins with "/*", and ends with "*/", like this:

```
/*This is a comment*/
P
{
  text-align:center;
  /*This is another comment*/
  color:black;
  font-family:arial;
}
```

5. The id and class Selectors

In addition to setting a style for a HTML element, CSS allows you to specify your own selectors called "id" and "class".

5.1 The id Selector

The id selector is used to specify a style for a **single**, unique element.

The id selector uses the id attribute of the HTML element, and is defined with a "#".

The style rule below will be applied to the element with id="para1":

```
#para1
{
  text-align:center;
  color:red;
}
```

NOTE : Do NOT start an ID name with a number! It will not work in Mozilla/Firefox.

5.2 The class Selector

The class selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements.

This allows you to set a particular style for any HTML elements with the same class.

The class selector uses the HTML class attribute, and is defined with a "."

In the example below, all HTML elements with class="center" will be center-aligned:

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

You can also specify that only specific HTML elements should be affected by a class.

In the example below, all p elements with class="center" will be center-aligned:

Web Organization LAB

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

NOTE: Do NOT start a class name with a number! This is only supported in Internet Explorer

6. Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

6.1 External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

```
<head>
    <link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a .css extension. An example of a style sheet file is shown below:

```
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
```

Do not leave spaces between the property value and the units! "margin-left:20 px" (instead of "margin-left:20px") will work in IE, but not in Firefox or Opera.

6.2 Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, by using the <style> tag, like this:

```
<head>
    <style type="text/css">
        hr {color:sienna;}
        p {margin-left:20px;}
        body {background-image:url("images/back40.gif");}
    </style>
</head>
```

Web Organization LAB

6.3 Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly!

To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

6.4 Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

For example, an external style sheet has these properties for the h3 selector:

```
h3
{
  color:red;
  text-align:left;
  font-size:8pt;
}
```

And an internal style sheet has these properties for the h3 selector:

```
h3
{
  text-align:right;
  font-size:20pt;
}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

```
color:red;
text-align:right;
font-size:20pt;
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

TIP: Even multiple external style sheets can be referenced inside a single HTML document.

6.5 Cascading order

What style will be used when there is more than one style specified for an HTML element?

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

Web Organization LAB

1. Browser default
2. External style sheet
3. Internal style sheet (in the head section)
4. Inline style (inside an HTML element)

So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or in a browser (a default value).

NOTE : If the link to the external style sheet is placed after the internal style sheet in HTML <head>, the external style sheet will override the internal style sheet!

7. CSS Background

CSS background properties are used to define the background effects of an element.

CSS properties used for background effects:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

7.1 Background Color

The background-color property specifies the background color of an element. The background color of a page is defined in the body selector:

```
body {background-color:#b0c4de;}
```

The background color can be specified by:

- name – a color name, like "red"
- RGB – an RGB value, like "rgb(255,0,0)"
- Hex – a hex value, like "#ff0000"

In the example below, the h1, p, and div elements have different background colors:

```
h1 {background-color:#6495ed;}  
p {background-color:#e0ffff;}  
div {background-color:#b0c4de;}
```

7.2 Background Image

The background-image property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element. The background image for a page can be set like this:

```
body {background-image:url('paper.gif');}
```

Web Organization LAB

7.3 Background Image – Repeat Horizontally or Vertically

By default, the `background-image` property repeats an image both horizontally and vertically.

Some images should be repeated only horizontally or vertically, or they will look strange, like this:

```
body
{
  background-image:url('gradient2.png');
}
```

If the image is repeated only horizontally (`repeat-x`), the background will look better:

```
body
{
  background-image:url('gradient2.png');
  background-repeat:repeat-x;
}
```

7.4 Background Image – Set position and no-repeat

When using a background image, use an image that does not disturb the text.

Showing the image only once is specified by the `background-repeat` property:

```
body
{
  background-image:url('img_tree.png');
  background-repeat:no-repeat;
}
```

In the example above, the background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much.

The position of the image is specified by the `background-position` property:

```
body
{
  background-image:url('img_tree.png');
  background-repeat:no-repeat;
  background-position:right top;
}
```

7.5 Background – Shorthand property

As you can see from the examples above, there are many properties to consider when dealing with backgrounds.

To shorten the code, it is also possible to specify all the properties in one single property. This is called a shorthand property.

Web Organization LAB

The shorthand property for background is simply "background":

```
body {background:#ffffff url('img_tree.png') no-repeat right top;} »
```

When using the shorthand property the order of the property values are:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

It does not matter if one of the property values are missing, as long as the ones that are present are in this order.

8. CSS Text

8.1 Text Color

The color property is used to set the color of the text. The color can be specified by:

- name – a color name, like "red"
- RGB – an RGB value, like "rgb(255,0,0)"
- Hex – a hex value, like "#ff0000"

The default color for a page is defined in the body selector.

```
body {color:blue;}
h1 {color:#00ff00;}
h2 {color:rgb(255,0,0);}
```

For W3C compliant CSS: If you define the color property, you must also define the background-color property.

8.2 Text Alignment

The text-align property is used to set the horizontal alignment of a text.

Text can be centered, or aligned to the left or right, or justified.

When text-align is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

```
h1 {text-align:center;}
p.date {text-align:right;}
p.main {text-align:justify;}
```

8.3 Text Decoration

The text-decoration property is used to set or remove decorations from text.

The text-decoration property is mostly used to remove underlines from links for design purposes:

```
a {text-decoration:none;}
```


Web Organization LAB

It can also be used to decorate text:

```
h1 {text-decoration:overline;}
h2 {text-decoration:line-through;}
h3 {text-decoration:underline;}
h4 {text-decoration:blink;}
```

It is not recommended to underline text that is not a link, as this often confuses users.

8.4 Text Transformation

The `text-transform` property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

```
p.uppercase {text-transform:uppercase;}
p.lowercase {text-transform:lowercase;}
p.capitalize {text-transform:capitalize;}
```

8.5 Text Indentation

The `text-indent` property is used to specify the indentation of the first line of a text.

```
p {text-indent:50px;}
```



9. CSS Font

CSS font properties define the font family, boldness, size, and the style of a text.

Difference between Serif and Sans-serif Fonts

On computer screens, sans-serif fonts are considered easier to read than serif fonts.



9.1 CSS Font Families

In CSS, there are two types of font family names:

- **generic family** – a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** – a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

9.2 Font Family

The font family of a text is set with the font-family property.

The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Web Organization LAB

Note: If the name of a font family is more than one word, it must be in quotation marks, like font-family: "Times New Roman".
More than one font family is specified in a comma-separated list:

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

9.3 Font Style

The font-style property is mostly used to specify italic text. This property has three values:

- **normal** – The text is shown normally
- **italic** – The text is shown in italics
- **oblique** – The text is "leaning" (oblique is very similar to italic, but less supported)

```
p.normal {font-style:normal;}  
p.italic {font-style:italic;}  
p.oblique {font-style:oblique;}
```

9.4 Font Size

The font-size property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Always use the proper HTML tags, like <h1> – <h6> for headings and <p> for paragraphs.

The font-size value can be an absolute or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

9.4.1 Set Font Size With Pixels

Setting the text size with pixels, gives you full control over the text size:

```
h1 {font-size:40px;}  
h2 {font-size:30px;}  
p {font-size:14px;}
```

The example above allows Firefox, Chrome, and Safari to resize the text, **but not Internet Explorer**.

The text can be resized in all browsers using the zoom tool (however, this resizes the entire page, not just the text).

Web Organization LAB

9.4.2 Set Font Size With Em

To avoid the resizing problem with Internet Explorer, many developers use em instead of pixels. The em size unit is recommended by the W3C.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: $pixels/16=em$

```
h1 {font-size:2.5em;} /* 40px/16=2.5em */  
h2 {font-size:1.875em;} /* 30px/16=1.875em */  
p {font-size:0.875em;} /* 14px/16=0.875em */
```

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

Unfortunately, there is still a problem with IE. When resizing the text, it becomes larger than it should when made larger, and smaller than it should when made smaller.

9.4.3 Use a Combination of Percent and Em

The solution that works in all browsers is to set a default font-size in percent for the body element:

```
body {font-size:100%;}  
h1 {font-size:2.5em;}  
h2 {font-size:1.875em;}  
p {font-size:0.875em;}
```

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

10.CSS Links

Links can be styled in different ways.

10.1 Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background-color).

Special for links are that they can be styled differently depending on what state they are in.

The four links states are:

- a:link – a normal, unvisited link
- a:visited – a link the user has visited
- a:hover – a link when the user mouse over it
- a:active – a link the moment it is clicked

```
a:link {color:#FF0000;} /* unvisited link */  
a:visited {color:#00FF00;} /* visited link */  
a:hover {color:#FF00FF;} /* mouse over link */  
a:active {color:#0000FF;} /* selected link */
```

Web Organization LAB

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

10.2 Common Link Styles

In the example above the link changes color depending on what state it is in. Let's go through some of the other common ways to style links:

10.2.1 Text Decoration

The text-decoration property is mostly used to remove underlines from links:

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}
```

10.2.2 Background Color

The background-color property specifies the background color for links:

```
a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

11.CSS Lists

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker

11.1 List

In HTML, there are two types of lists:

- unordered lists – the list items are marked with bullets
- ordered lists – the list items are marked with numbers or letters

With CSS, lists can be styled further, and images can be used as the list item marker.

11.2 Different List Item Markers

The type of list item marker is specified with the list-style-type property:

```
ul.a {list-style-type: circle;}
ul.b {list-style-type: square;}
ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
```

Web Organization LAB

Some of the property values are for unordered lists, and some for ordered lists.

Values for Unordered Lists	Value Description
none	No marker
disc	Default. The marker is a filled circle
circle	The marker is a circle
square	The marker is a square

Values for Ordered Lists	Value Description
armenian	The marker is traditional Armenian numbering
decimal	The marker is a number
decimal-leading-zero	The marker is a number padded by initial zeros (01, 02, 03, etc.)
georgian	The marker is traditional Georgian numbering (an, ban, gan, etc.)
lower-alpha	The marker is lower-alpha (a, b, c, d, e, etc.)
lower-greek	The marker is lower-greek (alpha, beta, gamma, etc.)
lower-latin	The marker is lower-latin (a, b, c, d, e, etc.)
lower-roman	The marker is lower-roman (i, ii, iii, iv, v, etc.)
upper-alpha	The marker is upper-alpha (A, B, C, D, E, etc.)
upper-latin	The marker is upper-latin (A, B, C, D, E, etc.)
upper-roman	The marker is upper-roman (I, II, III, IV, V, etc.)

Note: No versions of Internet Explorer (including IE8) support the property values "decimal-leading-zero", "lower-greek", "lowerlatin", "upper-latin", "armenian", or "georgian".

11.3 An Image as The List Item Marker

To specify an image as the list item marker, use the `list-style-image` property:

```
ul
{
list-style-image: url('sqpurple.gif');
}
```

The example above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari. If you want the image-marker to be placed equally in all browsers, a crossbrowser solution is explained below.

Web Organization LAB

11.4 Crossbrowser Solution

The following example displays the image-marker equally in all browsers:

```
ul
{
  list-style-type: none;
  padding: 0px;
  margin: 0px;
}
li
{
  background-image: url(sqpurple.gif);
  background-repeat: no-repeat;
  background-position: 0px 5px;
  padding-left: 14px;}
```

Example explained:

- For ul:
 - o Set the list-style-type to none to remove the list item marker
 - o Set both padding and margin to 0px (for cross-browser compatibility)
- For li:
 - o Set the URL of the image, and show it only once (no-repeat)
 - o Position the image where you want it (left 0px and down 5px)
 - o Position the text in the list with padding-left

11.5 List – Shorthand property

It is also possible to specify all the list properties in one, single property. This is called a shorthand property.

The shorthand property used for lists, is the list-style property:

```
ul
{
  list-style: square url("sqpurple.gif");
}
```

When using the shorthand property, the order of the values are:

- list-style-type
- list-style-position
- list-style-image

It does not matter if one of the values above are missing, as long as the rest are in the specified order.

Web Organization LAB

12.CSS Tables

The look of an HTML table can be greatly improved with CSS:

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Königlich Essen	Philip Cramer	Germany

12.1 Table Borders

To specify table borders in CSS, use the border property.

The example below specifies a black border for table, th, and td elements:

```
table, th, td
{
    border: 1px solid black;}
```

Notice that the table in the example above has double borders. This is because both the table, th, and td elements have separate borders.

To display a single border for the table, use the border-collapse property.

12.2 Collapse Borders

The border-collapse property sets whether the table borders are collapsed into a single border or separated:

```
table
{
    border-collapse: collapse;
}
table, th, td
{
    border: 1px solid black;
}
```

12.3 Table Width and Height

Width and height of a table is defined by the width and height properties.

The example below sets the width of the table to 100%, and the height of the th elements to 50px:

```
table
{
    width: 100%;
}
th
{
    height: 50px;
}
```


Web Organization LAB

```
height:50px;
}
```

12.4 Table Text Alignment

The text in a table is aligned with the `text-align` and `vertical-align` properties. The `text-align` property sets the horizontal alignment, like left, right, or center:

```
td
{
text-align:right;
}
```

Try it yourself »

The `vertical-align` property sets the vertical alignment, like top, bottom, or middle:

```
td
{
height:50px;
vertical-align:bottom;
}
```

12.5 Table Padding

To control the space between the border and content in a table, use the `padding` property on `td` and `th` elements:

```
td
{
padding:15px;
}
```

12.6 Table Color

The example below specifies the color of the borders, and the text and background color of `th` elements:

```
table, td, th
{
border:1px solid green;
}
th
{
background-color:green;
color:white;
}
```



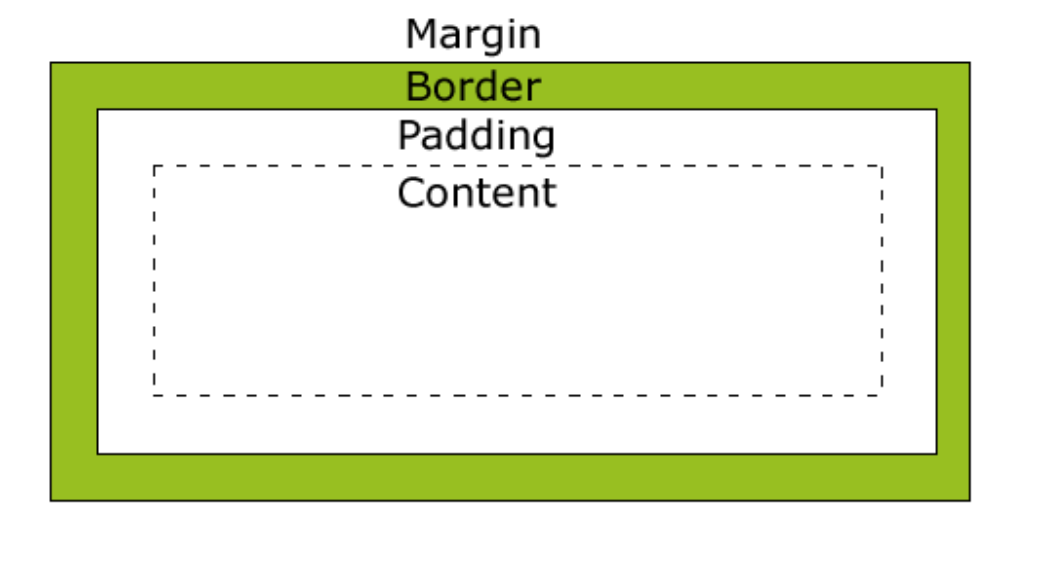
13. CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content.

The box model allows us to place a border around elements and space elements in relation to other elements.

The image below illustrates the box model:



Explanation of the different parts:

- **Margin** – Clears an area around the border. The margin does not have a background color, it is completely transparent
- **Border** – A border that goes around the padding and content. The border is affected by the background color of the box
- **Padding** – Clears an area around the content. The padding is affected by the background color of the box
- **Content** – The content of the box, where text and images appear

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

13.1 Width and Height of an Element

Important: When you specify the width and height properties of an element with CSS, you are just setting the width and height of the content area. To

Web Organization LAB

know the full size of the element, you must also add the padding, border and margin.

The total width of the element in the example below is 300px:

```
width:250px;
padding:10px;
border:5px solid gray;
margin:10px;
```

Let's do the math:

```
250px (width)
+ 20px (left and right padding)
+ 10px (left and right border)
+ 20px (left and right margin)
= 300px
```

Imagine that you only had 250px of space. Let's make an element with a total width of 250px:

```
width:220px;
padding:10px;
border:5px solid gray;
margin:0px;
```

The total width of an element should always be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should always be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

13.2 Browsers Compatibility Issue

If you tested the previous example in Internet Explorer, you saw that the total width was not exactly 250px.

IE includes padding and border in the width, when the width property is set, **unless a DOCTYPE is declared.**

To fix this problem, just add a DOCTYPE to the code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
  <style type="text/css">
    div.ex {
      width:220px;
```

Web Organization LAB

```
padding:10px;
border:5px solid gray;
margin:0px;}
</style>
</head>
```

14.CSS Border

The CSS border properties allow you to specify the style and color of an element's border.

14.1 Border Style

The border-style property specifies what kind of border to display.

None of the border properties will have ANY effect unless the **border-style** property is set!

border-style values:

none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

groove: Defines a 3D grooved border. The effect depends on the border-color value

ridge: Defines a 3D ridged border. The effect depends on the border-color value

inset: Defines a 3D inset border. The effect depends on the border-color value

outset: Defines a 3D outset border. The effect depends on the border-color value

14.2 Border Width

The border-width property is used to set the width of the border.

The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

```
p. one
{
border-style:solid;
border-width:5px;
}
p. two
{
border-style:solid;
```

Web Organization LAB

```
border-width:medium;
}
```

14.3 Border Color

The `border-color` property is used to set the color of the border. The color can be set by:

- **name** – specify a color name, like "red"
- **RGB** – specify a RGB value, like "rgb(255,0,0)"
- **Hex** – specify a hex value, like "#ff0000"

You can also set the border color to "transparent".

Note: The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

```
p.one
{
border-style:solid;
border-color:red;
}
p.two
{
border-style:solid;
border-color:#98bf21;
}
```

14.4 Border – Individual sides

In CSS it is possible to specify different borders for different sides:

```
p
{
border-top-style:dotted;
border-right-style:solid;
border-bottom-style:dotted;
border-left-style:solid;
}
```

The example above can also be set with a single property:

```
border-style:dotted solid;
```

The `border-style` property can have from one to four values.

- **border-style:dotted solid double dashed;**
 - o top border is dotted
 - o right border is solid
 - o bottom border is double
 - o left border is dashed
- **border-style:dotted solid double;**
 - o top border is dotted
 - o right and left borders are solid

Web Organization LAB

- o bottom border is double
- **border-style:dotted solid;**
 - o top and bottom borders are dotted
 - o right and left borders are solid
- **border-style:dotted;**
 - o all four borders are dotted

The border-style property is used in the example above. However, it also works with border-width and border-color.

14.5 Border – Shorthand property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the border properties in one property. This is called a shorthand property.

The shorthand property for the border properties is "border":

```
border:5px solid red;
```

When using the border property, the order of the values are:

- border-width
- border-style
- border-color

It does not matter if one of the values above are missing (although, border-style is required), as long as the rest are in the specified order.

15.CSS Margin

The CSS margin properties define the space around elements.

15.1 Margin

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent.

The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Possible Values

Value	Description
auto	The browser sets the margin. The result of this is dependent of the browser
length	Defines a fixed margin (in pixels, pt, em, etc.)
%	Defines a margin in % of the containing element

It is possible to use negative values, to overlap content.

15.2 Margin – Individual sides

In CSS, it is possible to specify different margins for different sides:

```
margin-top:100px;  
margin-bottom:100px;  
margin-right:50px;  
margin-left:50px;
```

Web Organization LAB

15.3 Margin – Shorthand property

To shorten the code, it is possible to specify all the margin properties in one property. This is called a shorthand property.

The shorthand property for all the margin properties is "margin":

```
margin:100px 50px;
```

The margin property can have from one to four values.

- **margin:25px 50px 75px 100px;**
 - o top margin is 25px
 - o right margin is 50px
 - o bottom margin is 75px
 - o left margin is 100px
- **margin:25px 50px 75px;**
 - o top margin is 25px
 - o right and left margins are 50px
 - o bottom margin is 75px
- **margin:25px 50px;**
 - o top and bottom margins are 25px
 - o right and left margins are 50px
- **margin:25px;**
 - o all four margins are 25px

16. CSS Padding

The CSS padding properties define the space between the element border and the element content.

16.1 Padding

The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element.

The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all paddings at once.

Possible Values

Value	Description
Length	Defines a fixed margin (in pixels, pt, em, etc.)
%	Defines a margin in % of the containing element

16.2 Padding – Individual sides

In CSS, it is possible to specify different padding for different sides:

```
padding-top:25px;  
padding-bottom:25px;  
padding-right:50px;  
padding-left:50px;
```

16.3 Padding – Shorthand property

To shorten the code, it is possible to specify all the padding properties in one property. This is called a shorthand property.

Web Organization LAB

The shorthand property for all the padding properties is "padding":

```
padding:25px 50px;
```

The padding property can have from one to four values.

- **padding:25px 50px 75px 100px;**
 - o top padding is 25px
 - o right padding is 50px
 - o bottom padding is 75px
 - o left padding is 100px
- **padding:25px 50px 75px;**
 - o top padding is 25px
 - o right and left paddings are 50px
 - o bottom padding is 75px
- **padding:25px 50px;**
 - o top and bottom paddings are 25px
 - o right and left paddings are 50px
- **padding:25px;**
 - o all four paddings are 25px

17. Grouping Selectors

In style sheets there are often elements with the same style.

```
h1
{
color:green;
}
h2
{
color:green;
}
p
{
color:green;
}
```

To minimize the code, you can group selectors. Separate each selector with a comma.

In the example below we have grouped the selectors from the code above:

```
h1, h2, p
{
color:green;
}
```