

Introduction to Website Design

Contents

1.1 Introduction.....	2
1.2 General Concepts	3
a. Web Server	3
b. Internet Protocol (IP) Address	3
c. Web Page Addresses (URLs)	4
d. Browser.....	5
e. File Transfer Protocol (FTP).....	7
f. Domain Names and Hosting	7
1.3 Web design	8
a. Design	8
1) Interaction Design (IxD).....	8
2) User Interface (UI) design	9
3) User Experience (UX) design	9
b. Development.....	9
1) Authoring	10
2) Styling	10
3) Scripting and Programming	10
c. Content Strategy	11
d. Multimedia.....	12
1.4 Website Design Technologies.....	13
a. Hyper Text Markup Language (HTML)	13
b. Cascading Style Sheets (CSS)	14
c. JavaScript and DOM scripting.....	15
d. Server-Side Programming and Database Management	16
e. JQuery	16
f. Bootstrap	16
g. Content Management System (CMS).....	17
1.5 Websites Types	18
1.6 Web Page Authoring Tools.....	19

1.1 Introduction

The **Internet** is a network of connected computers. No company owns the Internet; it is a cooperative effort governed by a system of standards and rules. The purpose of connecting computers together, of course, is to share information. There are many ways information can be passed between computers, including email, file transfer (FTP), and many more specialized modes upon which the Internet is built. These standardized methods for transferring data or documents over a network are known as **protocols**. The Web uses a protocol called **HTTP (Hyper Text Transfer Protocol)**. Through hypertext links and by using protocols, the documents link to one another thus forming a huge “web” of connected information.

There are two other famous terms (intranet and extranet). **Intranet** is a private network of connected devices related to specific company. It allows only staff of the company to access all network resources, information, and services. In the other hand, the **extranet** allows partners, suppliers, and authorized customers to access the private network (intranet). Information and data access performed through a proper account or link system.

The **Web** (originally called the **World Wide Web**, thus the “**www**” in site addresses) is just one of the ways information can be shared over the Internet. Standards of web are almost created by the **World Wide Web Consortium (W3C)** which is the main international standards organization for the World Wide Web (abbreviated WWW or W3). Founded and currently led by Tim Berners-Lee, the consortium is made up of member organizations which maintain full-time staff for the purpose of working together in the development of standards for the World Wide Web. The W3C also engages in education and outreach, develops software and serves as an open forum for discussion about the Web.

1.2 General Concepts

There are many important concepts related to web design process:

a. Web Server

Let's talk more about the computers that make up the Internet and these computers are known as **servers**. More accurately, the server is the software (not the computer itself) that allows the computer to communicate with other computers; however, it is common to use the word "**server**" to refer to the computer as well. The role of server software is to wait for a request for information, then retrieve and send that information back as quickly as possible. In order for a computer to be part of the Web, it must be running special web server software that allows it to handle Hypertext Transfer Protocol transactions. Web servers are also called "HTTP servers". There are many server software options out there, but the two most popular are **Apache** (AppServ) open source software, Nginx, XAMPP, WAMP, and **Microsoft Internet Information Services (IIS)**.

b. Internet Protocol (IP) Address

Every computer and device (modem, router, smartphone, cars, etc.) connected to the Internet is assigned a unique numeric IP address (IP stands for Internet Protocol). For example, the computer that hosts oreilly.com has the IP address 208.201.239.100. All those numbers can be dizzying, so fortunately, **the Domain Name System (DNS)** was developed to allow us to refer to that server by its domain name, "oreilly.com", as well. The numeric IP address is useful for computer software, while the domain name is more accessible to humans. Matching the text domain names to their respective numeric IP addresses is the job of a separate DNS server.

It is possible to configure your web server so that more than one domain name is mapped to a single IP address, allowing several sites to share a single server see figure (1).

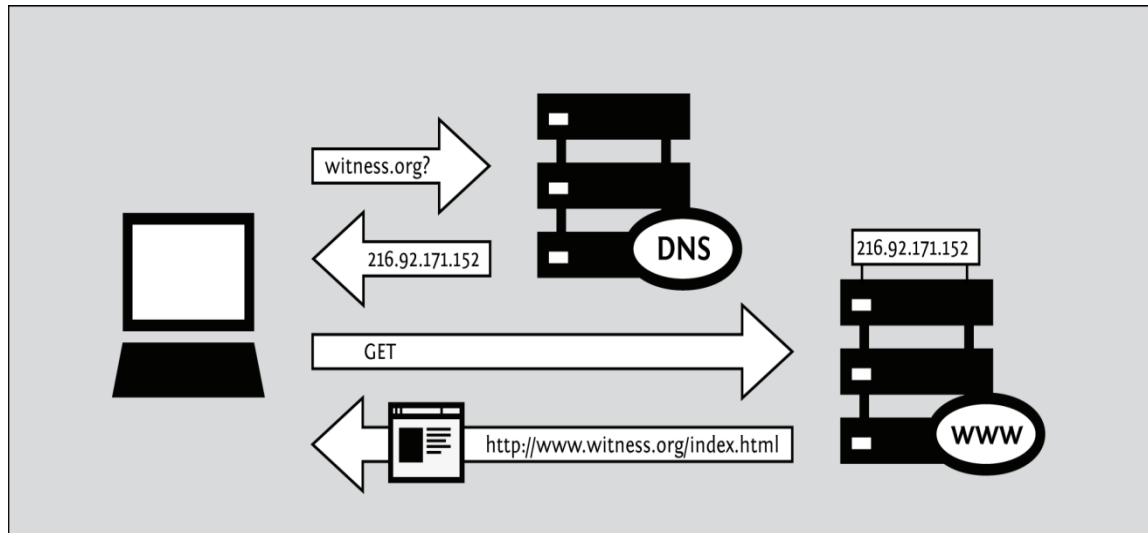


Figure 1: Simple Work of DNS

c. Web Page Addresses (URLs)

Every page and resource on the Web has its own special address called a URL, which stands for **Uniform Resource Locator**. It's nearly impossible to get through a day without seeing a URL (pronounced "U-R-L") plastered on the side of a bus, printed on a business card, or broadcast on a television commercial.



Figure 2: URL

Some URLs are short and sweet. Others may look like crazy strings of characters separated by dots (periods) and slashes, but each part has a specific purpose. Let's pick one apart. The parts of a URL are 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).

d. Browser

People use desktop browsers, mobile browsers, and other assistive technologies as clients to access documents on the Web. The server returns the documents for the browser (also referred to as the user agent in technical circles) to display. The requests and responses are handled via the HTTP protocol, mentioned earlier. Although we've been talking about "documents," HTTP can be used to transfer images, movies, audio files, data, scripts, and all the other web resources that commonly make up web sites and applications, see figure (3).

It is common to think of a **browser** as a window on a computer monitor with a web page displayed in it. These are known as graphical browsers or desktop browsers. The most popular desktop browsers as of this writing include *Internet Explorer for Windows*, *Chrome*, *Firefox*, *Safari*, and *Opera*. These days, however, more and more people are accessing the Web on the go using browsing clients built into mobile phones or tablets. Even on the desktop browsers that first introduced us to the wide world of the Web, pages may look and perform differently from browser to browser. This is due to varying support for web technologies and the users' ability to set their own browsing preferences.

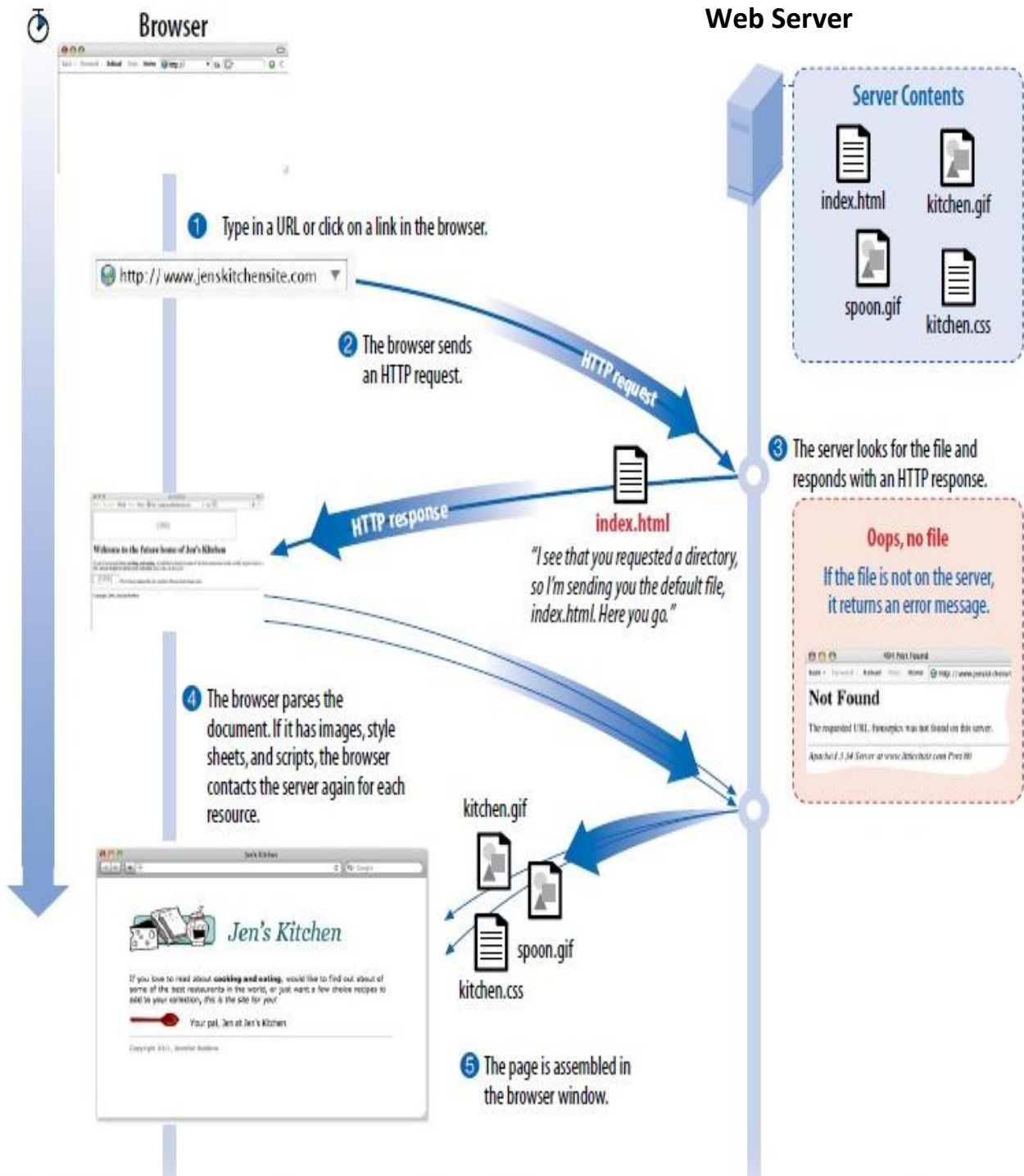


Figure 3: HTTP Request and Response

e. File Transfer Protocol (FTP)

An FTP program enables you to upload and download files between your computer and the computer that will serve your pages to the web. There are also dedicated FTP programs for Windows (WS_FTP, CuteFTP, ceFTP, and Filezilla) or for Macintosh(Transmit, Cyberduck, and Fetch).

f. Domain Names and Hosting

Domain names help users find their way around the Internet. You already know domain names because they are commonly surrounded by *www* on the front and *.com* on the end. Domains can also include various endings such as *.org*, *.edu*, and *.gov*. Domain names exist because it's not very convenient for you to use or remember IP addresses. DNS translate easy-to-understand domain names into IP addresses. A DNS converts a familiar string of letters, the "domain name," to the numbered IP address. Instead of typing the IP address 72.32.147.166 into a web browser, you can type the domain name, such as *www.digitalclassroombooks.com*. A DNS on the Internet converts your requested domain into the appropriate IP address, which routes your request to the appropriate web server.

Web servers maintain a constant connection to the Internet, so your websites are available 24 hours a day. Because most companies want their web servers to be available all day, every day, they are often maintained by web hosting companies. These firms are paid to maintain your web server so that it is always accessible and running. If you run a small website, they may put your site on a server that is shared with other sites. For more demanding sites, or sites with sensitive information, a business will pay higher fees for a dedicated server.

Even large companies will often turn to hosting businesses to maintain their web servers, although some companies may elect to place their web servers within their own company.

1.3 Web design

The term “web design” has come to encompass a number of disciplines divided the myriad roles and responsibilities typically the term “web design” into four very broad categories: **design, development, content strategy, and multimedia**. Large-scale websites are almost always created by a team of people, numbering from a handful to hundreds. Each member of the team focuses on one facet of the site-building process.

a. Design

Designer” often is responsible for more than one (if not all) of these responsibilities: **User Experience, Interaction, and User Interface design**. On the Web, the first matter of business is designing how the site works. Before picking colors and fonts, it is important to identify the site’s goals, how it will be used, and how visitors move through it. These tasks fall under the disciplines of **Interaction Design (IxD), User Interface (UI) design, and User Experience (UX) design**.

1) Interaction Design (IxD)

The goal of the Interaction Design is to make the site as easy, efficient and delightful to use as possible.

2) User Interface (UI) design

Tends to be more narrowly focused on the functional organization of the page as well as the specific tools (buttons, links, menus, and so on) that users use to navigate content or accomplish tasks.

3) User Experience (UX) design

UX design is based on a solid understanding of users and their needs based on observations and interviews. According to Donald Norman (who coined the term), user experience design includes “all aspects of the user’s interaction with the product: how it is perceived, learned, and used.” For a website or application, that includes the visual design, the user interface, the quality and message of the content and even overall site performance. The experience must be in line with the organization’s brand and business goals in order to be successful.

b. Development

A fair amount of the web design process involves the creation and troubleshooting of the documents, style sheets, scripts, and images that make up a site. At web design firms, the team that handles the creation of the files that make up the website (or templates for pages that get assembled dynamically) is usually called the development or production department. Web developers may not design the look or structure of the site themselves, but they do need to communicate well with designers and understand the intended site goals so they may suggest solutions that meet those goals. The broad disciplines that fall under development are **authoring, styling, and scripting/programming**.

1) Authoring

Authoring is the term used for the process of preparing content for delivery on the Web, or more specifically, marking up the content with HTML tags that describe its content and function. If you want a job as a web developer, you need to have an intricate knowledge of HTML and how it functions on various browsers and devices. The HTML specification is constantly evolving, which means you'll need to keep up with the latest best practices and opportunities as well as bugs and limitations.

2) Styling

In web design, the appearance of the page in the browser is controlled by style rules written in CSS (Cascading Style Sheets). In contemporary web design, the appearance of the page is handled separately from the HTML markup of the page. Again, if you are interested in working in web development, knowing your way around CSS and how it is supported (or not supported) by browsers is guaranteed to be part of your job description.

3) Scripting and Programming

As the web has evolved into a platform of applications for getting stuff done, programming has never been more important. JavaScript is the language that makes elements on web pages do things. It adds behaviors and functionality to elements in the page and even to the browser window itself. There are other web-related programming languages as well, including PHP, Ruby, Python, and ASP.NET, that run on the server and process data and information before it is

sent to the user's browser. See the table (1) to view the differences between "Frontend versus Backend".

Table 1: Frontend versus Backend

Frontend	Backend
1. "Frontend" refers to any aspect of the design process that appears in or relates directly to the browser.	"Backend" refers to the programs and scripts that work on the server behind the scenes to make web pages dynamic and interactive. In general, backend web development falls in the hands of experienced programmers, but it is good for all web designers to be familiar with backend functionality.
2. Graphic design and image production.	Information design as it pertains to how the information is organized on the server.
3. Interface design.	Forms processing.
4. Information design as it pertains to user's experience of the site.	Database programming.
5. HTML document and style sheet development.	Content management systems.
6. JavaScript.	Other server-side web applications using PHP, JSP, Ruby, ASP.NET, Java, and other programming languages.

c. Content Strategy

Third on our list, though ideally first in the actual website creation process, is the critical matter of the site's content itself. Anyone who uses the title "web designer" needs to be aware that everything we do supports the process of

getting the content, message, or functionality to our users. Furthermore, good writing can help the user interfaces we create be more effective. Of course, someone needs to create the content and maintain it don't underestimate the resources required to do this successfully. In addition, I want to call your attention to two content-related specialists on the modern web development team: the **Content Strategist** and **Information Architect (IA)**.

When the content isn't written right, the site can't be fully effective. A Content Strategist makes sure that every bit of text on a site, from long explanatory text down to the labels on buttons, supports the brand identity and marketing goals of the company. Content strategy may also extend to data modeling and content management on a large and ongoing scale, such as planning for content reuse and update schedules.

An Information Architect (also called an Information Designer) organizes the content logically and for ease of findability. It may be responsible for search functionality, site diagrams, and how the content and data is organized on the server. Information architecture is inevitably entwined with UX and UI design, and it is not uncommon for a single person or team to perform all roles.

d. Multimedia

One of the cool things about the Web is that you can add multimedia elements to a site, including sound, video, animation, and even interactive games. You may decide to add multimedia skills, such as audio and video editing or Flash development to your web design tool belt, or you may decide to go all in and become a multimedia specialist. Web development companies usually look

for people who have mastered the standard multimedia tools, and have a good visual sensibility and an instinct for intuitive and creative multimedia design.

1.4 Website Design Technologies

The following is a list of technologies associated with web development. Which languages and technologies you learn will depend on the role you. Web-related technologies are:

- a. Hypertext Markup Language (HTML)**
- b. Cascading Style Sheets (CSS)**
- c. JavaScript and DOM scripting**
- d. Server-side programming and database management**
- e. JQuery**
- f. Bootstrap**
- g. Content Management System (CMS)**

a. Hyper Text Markup Language (HTML)

Hyper Text Markup Language (HTML) is the language used to create web page documents. There are a few versions of HTML In use today: HTML4.01 is the most firmly established and the newer, more robust HTML5 is gaining steam and browser support. Both versions have a stricter implementation called XHTML (eXtensible HTML), which is essentially the same language with much stricter syntax rules.

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. The markup indicates the document's underlying structure (you can think of it as a detailed, machine-readable outline). The best way to learn HTML is to write out some pages by hand, if you end up working in web production, you'll live and breathe HTML.

b. Cascading Style Sheets (CSS)

While HTML is used to describe the content in a web page, it is Cascading Style Sheets (CSS) that describe how that content should look. In the web design biz, the way the page looks is known as its presentation. That means fonts, colors, background images, line spacing, page layout are controlled with CSS. With the newest version (CSS3), you can even add special effects and basic animation to your page.

CSS also provides methods for controlling how documents will be presented in contexts other than the traditional desktop browser, such as in print and on devices with small screen widths. It also has rules for specifying the nonvisual presentation of documents, such as how they will sound when read by a screen reader although those are not well supported. Style sheets are also a great tool for automating production because you can change the way an element looks across all the pages in your site by editing a single style sheet document. Style sheets are supported to some degree by all modern browsers.

Although it is possible to publish web pages using HTML alone, you'll probably want to take on style sheets so you're not stuck with the browser's default styles. If you're looking into designing websites professionally, proficiency at style sheets is mandatory.

c. JavaScript and DOM scripting

JavaScript is a scripting language that is used to add interactivity and behaviors to web pages, including these:

- 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 she visits
- Building interface widgets, such as expanding menus

JavaScript is used to manipulate the elements on the web page, the styles applied to them, or even the browser itself. There are other web scripting languages, but JavaScript (also called ECMAScript) is the standard and most ubiquitous.

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. Writing JavaScript is a type of programming, so it may be time-consuming to learn if you have no prior programming experience. Many people teach themselves JavaScript by reading books and following and modifying existing examples. Most web-authoring tools come with standard scripts that you can use right out of the box for common functions.

d. Server-Side Programming and Database Management

Some simple websites are collections of static 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 web applications running on the server. There are a number of programming languages and frameworks (listed in parentheses) that are used to create web applications, including (PHP, Python, Ruby, JavaScript, Java and ASP.Net).

Developing web applications is programmer territory and is not expected of all web designers. However, that doesn't mean you can't offer such functionality to your clients. It is possible to get shopping carts, content management systems, mailing lists, and blogs.

e. JQuery

"The write less, do more", is a JavaScript library. This library characterized by its simplicity, easy to learn, fast and small JavaScript codes, and rich features. It simplifies JavaScript programming and makes using of JavaScript more flexible. It is the common framework of JavaScript language which used by many of the biggest companies, such as Google, Microsoft, IBM, and Netflix.

f. Bootstrap

It is a free front-end framework for faster and easier web development. It is the most popular framework to develop responsive mobile websites. It consists primarily of HTML, CSS, and JavaScript layouts that can be used to build pages templates. In bootstrap, the designer can ignore focusing on CSS rules and write

only HTML tags and call only the required CSS rules. Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops. There are many advantages:

- Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap.
- Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops.
- Mobile-first approach: In Bootstrap 3, mobile-first styles are part of the core framework
- Browser compatibility: Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Edge, Safari, and Opera).

g. Content Management System (CMS)

A web content management system (CMS) is server-based software that simplifies, structures, and manages the creation and delivery of web content, providing a graphic user interface that allows users to create web pages and other web information without having to learn HTML, CSS, or other kinds of web coding. Content management systems offer powerful advantages over old-style hand-coded static web pages. The key to understanding the advantages of a CMS is the separation of content from presentation code. This separation of content and form is much more flexible than static HTML web pages, where content is embedded in one fixed format of HTML markup and CSS page styling. In a CMS the content is drawn from a web database and can be presented in many different kinds of templates, in many different arrangements, and for many kinds

of display devices, including desktop computers, laptops, tablets, and other mobile devices.

Popular open-source content management systems like WordPress, Drupal, and Joomla are the most widely used CMS programs for individuals, governments, universities, and small to medium-sized businesses. Large business, government, news, and e-commerce sites typically use more complex commercial CMS programs like OpenText CMS (formerly Red Dot), Ingeniux CMS, or Ektron CMS. Commercial CMS products typically can handle much larger volumes of content, and offer sophisticated systems to support e-commerce, large-volume credit card transactions, and other business and financial functions.

1.5 Websites Types

Users tend to view Web sites, and thus Web site design, by the function of the site or by its visual appearance. It is important to be able to describe sites this way; however, there are many more ways to categorize them.

Another way we might group sites is within the following broad categories:

- **Informational sites** these sites provide information about a particular subject or organization (the “brochureware” sites). These are the most common Web sites on the Internet and often take on aspects of the other site categories over time.
- **Transactional sites:** this type of site can be used to conduct some transaction or task. E-commerce sites fall into this category. These sites provide information or transaction-related facilities, but focus on the interaction between the visitors of the site.

- **Community sites:** community-based sites tend to focus on a particular topic or type of person and encourage interaction between likeminded individuals.
- **Entertainment sites:** these sites are for game playing or some form of amusing interaction, which may include transactional, community, and informational elements.
- **Other sites:** included here are artistic or experimental sites, personal Web spaces such as Web logs (also called blogs), and sites that may not follow common Web conventions or have a well-defined economic purpose.

1.6 Web Page Authoring Tools

Web authoring tools are similar to desktop publishing tools, but the end product is a web page (an HTML file and its supporting files). These tools provide a visual “WYSIWYG” (What You See Is What You Get, pronounced “whizzy-wig”) interface and shortcuts that save you from typing repetitive HTML and CSS. These tools won’t excuse you from learning HTML. Even the most sophisticated tools won’t generate HTML as clean or well-considered as a professional writing by hand, but they can speed up the process once you know what you’re doing.

The following are some popular web-authoring programs:

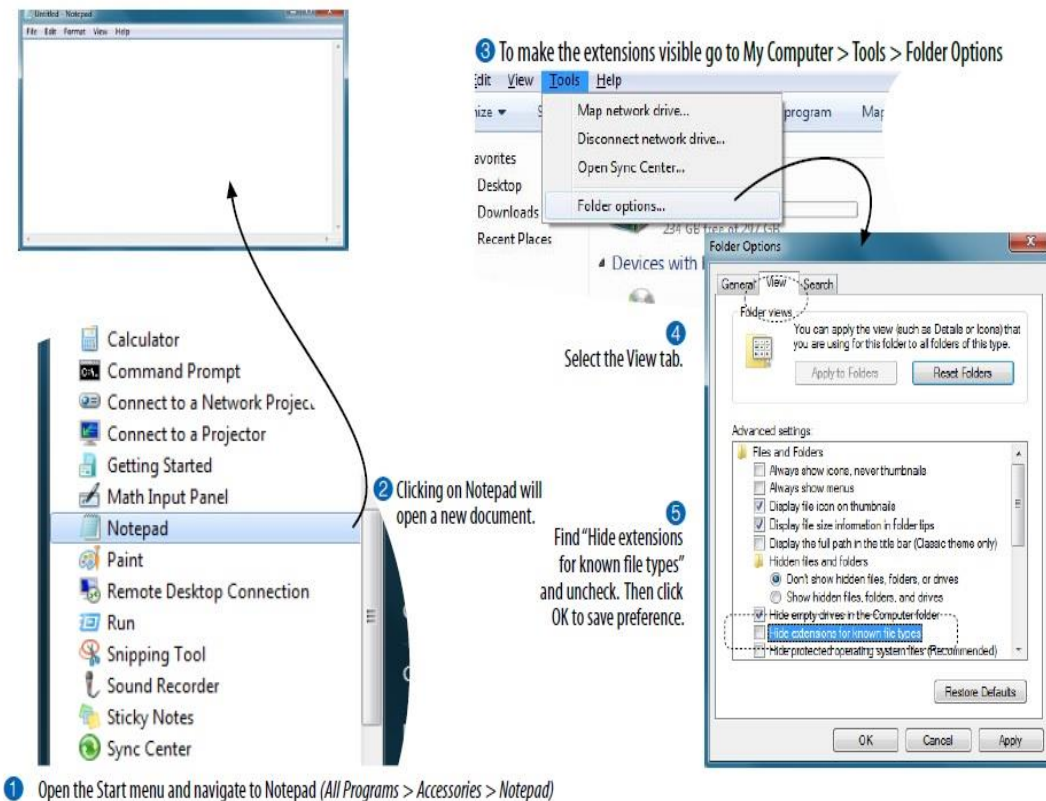
- **Adobe Dreamweaver:** This is the hands-down industry standard due to its relatively clean code and advanced features.
- **Microsoft Expression Web (Windows only):** Part of Microsoft’s suite of professional design tools, MS Expression Web boasts standards-compliant code and CSS-based layouts.

- **Nvu (Linux, Windows, and Mac OS X):** (pronounced N-view, for “new view”) is an open source tool that matches many of the features in Dreamweaver.
- **Notepad++** is a free source code editor that supports several languages. Running in the MS Windows environment, its use is governed by GPL License. Notepad++ is written in C++ and uses pure Win32 API and STL which ensures a higher execution speed and smaller program size. By optimizing as many routines as possible without losing user friendliness.
- **Atom** is (Linux and windows) code editor that offers you many features such as built-in packages, smart and auto completion tags, and file system browser. It also supports multiple panes and finds and replace function.
- **Vim** is one of best editors of UNIX. Vim is almost like Vi editor but it has more better features. Vim used to create and change any code or text effectively. It supports many plugins and old file formats and programming languages.
- **Sublime Text** is a text editor used for coding, styling, and markup. It allows for better presentation and highlighting codes. It also offers features like sorting, rearranging, and changing settings of code indentation.

Contents

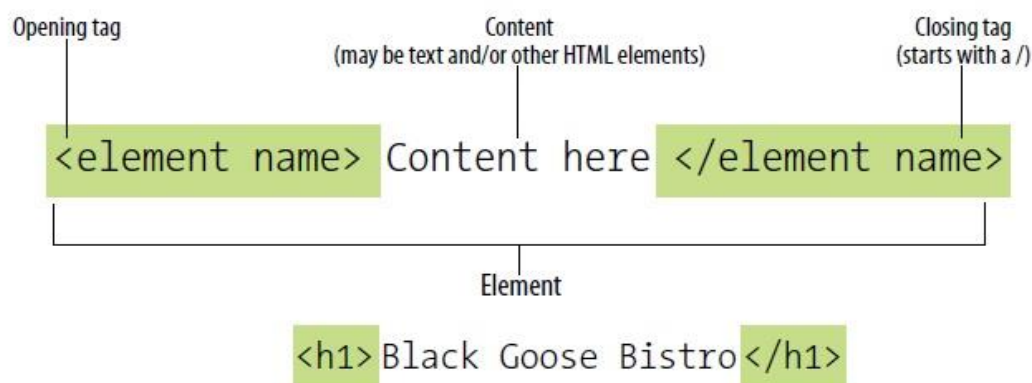
1. Creating document in notepad (Windows OS)	2
2. Document Structure.....	2
3. Basic Document Structure	4
4. Text Elements.....	4
5. Image	7

1. Creating document in notepad (Windows OS)



2. Document Structure

- HTML elements with an opening tag (<p> for a paragraph, for example) and closing tag (</p>). Before we start adding tags to our document, let's look at the anatomy of an HTML element (its syntax) and firm up some important terminology.



- To understand the structure of any web page you can view the source of the web page and see how it designed. HTML source of any web page can be viewed by right click on the web page and select view page source.
- A tag consists of the element name (usually an abbreviation of a longer descriptive name) within angle brackets (< >). The browser knows that any text within brackets is hidden and not displayed in the browser window.
- The element name appears in the opening tag (also called a start tag) and again in the closing (or end) tag preceded by a slash (/). The closing tag works something like an “off” switch for the element.
- Not all elements have content, however. Some are empty by definition, such as the **img** element used to add an image to the page.
- Capitalization. In HTML, the capitalization of element names is not important. So , , and are all the same.



3. Basic Document Structure

1. **DOCTYPE** is a document type declaration (also called DOCTYPE declaration) that identifies this document as an HTML5 document.
2. The **html** element is called the root element because it contains all the elements in the document, and it may not be contained within any other element. It is used for both HTML and XHTML documents.
3. The **head** element contains descriptive information about the document itself, such as its title, the style sheet(s) it uses, scripts, and other types of “meta” information.
4. The **meta** elements within the head element provide information about the document itself. A meta element can be used to provide all sorts of information, but in this case, it specifies the character encoding (the standardized collection of letters, numbers, and symbols) used in the document.
5. Also in the head is the mandatory **title** element. According to the HTML specification, every document must contain a descriptive title.
6. Finally, the **body** element contains everything that we want to show up in the browser window.

4. Text Elements

- **Semantic Markup** based on choosing the HTML element that provides the most meaningful description of the content.
- The markup gives the document structure, the way elements follow each other or nest within one another creates relationships between the

elements (its technical name is the **DOM**, for **Document Object Model**).

- Title element required for every document, it is quite useful as well. The title is what is displayed in a **user's Bookmarks or Favorites list** and on tabs in desktop browsers.
- `<h1></h1>` to `<h6></h6>` are heading element used to make main head and sub head in the page content.
- `<p></p>` to write paragraph in the page content.
- `<!-- -->` to add comments in page content.
- `` or `<i></i>` to emphasize the text (italic).
- `<u></u>` underlined the text.
- `
` to add line break.
- *Exercise1: create html page using the above text elements.*

1. Type Design

1. Serif Typefaces

+ text paragraph

1. Baskerville

1. Description

+ text paragraph

2. History

+ text paragraph

2. Georgia

+ text paragraph

2. Sans-Serif Typefaces

+ text paragraph

- *Example: Markup an HTML page to produce the following:*

HTML Text Element

Semantic Markup

based on choosing the HTML element that provides the most meaningful description of the content. The markup gives the document structure, the way elements follow each other or nest within one another creates relationships between the elements (its technical name is the **DOM**, for **Document Object Model**).

```
<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>HTML Elements </title>

<head/>

<body>

<h1>HTML text Element</h1>

<h2>Semantic Markup</h2>
```

`<p>`based on choosing the HTML element that provides the most meaningful description of the content. The markup gives the document structure,`
` the way elements follow each other or nest within one another creates relationships between the elements`
`)its technical name is the `DOM` for `Document Object Model`).

```
<body/>

</html>
```

- *Example: create html page using heading elements to show the following:*

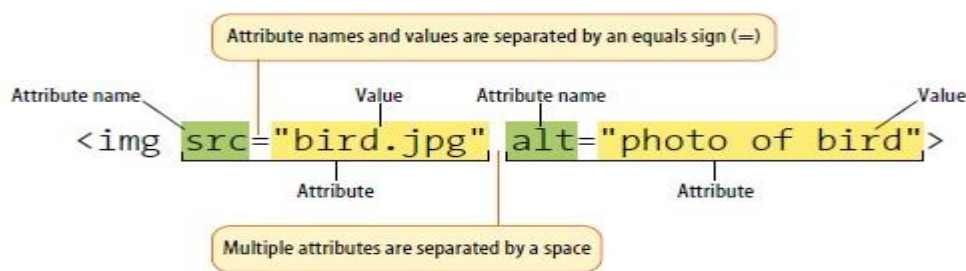
Hint:

```
h1 — Type Design
h2 — Serif Typefaces
    Serif typefaces have small slabs at the ends of letter strokes. In
    general, serif fonts can make large amounts of text easier to read.
h3 — Baskerville
h4 — Description
    Description of the Baskerville typeface.
h4 — History
    The history of the Baskerville typeface.
h3 — Georgia
    Description and history of the Georgia typeface.
h2 — Sans-serif Typefaces
    Sans-serif typefaces do not have slabs at the ends of strokes.
```

- Default font size is 16px.
- Other Text Elements (Exercise)

Element	Description
<code><mark></mark></code>	Defines marked/highlighted text
<code></code>	Defines bold text
<code></code>	Define important strong text
<code><hr></code>	Horizontal rule
<code><big></big></code>	Defines big text
<code><small></small></code>	Defines small text
<code><sub></sub></code>	Defines subscripted text
<code><sup></sup></code>	Defines superscripted text
<code><pre></pre></code>	Defines preformatted text
<code><ins></ins></code>	Defines inserted text
<code></code>	Defines deleted text

5. Image



- Images must be in the GIF, JPEG, or PNG file and image files need to be named with the proper suffixes .gif, .jpg (or .jpeg) and .png , respectively.
- You can insert other image format after converting it to proper format.

- Helper applications can be used by browser to open any unsupported format.
- **Attributes** go after the element name, separated by a space. In non-empty elements, attributes go in the opening tag only:
`<element attributename="value">`
`<element attributename="value">Content</element>`
- You can also put more than one attribute in an element in any order. Just keep them separated with spaces.
`<element attribute1="value" attribute2="value">`
- Value might be a number, a word, a string of text, a URL, or a measurement, depending on the purpose of the attribute. Note that quotation marks in HTML files need to be straight (") not curly (").
- **Attributes in image element:**
 - **Alt** provides text that should be displayed if the image is not available.
 - **Src** provides the name of the image file that should be inserted.
 - **Width="number"** indicated the width of the image on the web page.
 - **Height="number"** indicated the height of the image on the web page.
 - **Number** indicates value in pixel (width="100px", width="100") or percentage (width="50%").
 - *Example: Call image and make the width and height 300px in view:*

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>HTML Image</title>
</head>
<body>
  <h1> Image</h1>
  
  <body>
</html>
```

Preview

Image



- *Example 2: show the same image with 50% in height and width:*

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>HTML Image</title>
```

```
<head/>

<body>



<body>

</html>
```

Preview:

Image



Contents

2.1 Anchor	2
2.2 Addressing Modes.....	3
a. Absolute URLs	3
b. Relative URLs	4
2.3 Linking to Document Fragment	6
a. Link to Specific Point in the Same Page	6
b. Link to Specific location in another Page.....	7
2.4 Lists.....	8
a. Unordered Lists.....	8
b. Ordered Lists	9
c. Description Lists.....	10

2.1 Anchor

- Anchor is location in your site or someone else's site.

` Link text or Element `

- `<a>` **linked text or element**`` used to **make linking** in web page.
- **href** attribute refers to **hypertext reference**, it provides the address of the page or resource (its URL) to the browser.
- **href attribute** used to **add URL to target destination** (web page or location inside page).
- **Example 1:** ` click to visit google `
- **Example 2:** ` <h1> click to visit google </h1>`
- You can make image as link in web page:
Example 3: ``
- All graphical browsers display linked text as **blue** and **underlined** by default.
- Some older browsers put a **blue** border around **linked images**, but most current ones do not.
- **Visited** links generally display in **purple**. **Users can change these colors** in their browser preferences, and, of course, you can change the appearance of links for your sites using style sheets.
- **Target** attribute with `<a>` tag can be used to redirect page to new tab.
- **Target=_blank** opens the linked document in a new window or tab.
- **Target=_self** opens the linked document in the same frame as it was clicked (this is default).
- **Target=_parent** opens the linked document in the parent frame.
- **Target=_top** opens the linked document in the full body of the window.

```
<!DOCTYPE html>
<html>
  <head>
    <title> Anchor</title>
  </head>
  <body>
    <a href=http://www.google.com><h1>click to visit
google</h1></a>
  </body>
</html>
```

[click to visit google](http://www.google.com)

```
<!DOCTYPE html>
<html>
  <head>
    <title> Anchor</title>
  </head>
  <body>
    <a href=http://www.google.com> </a>
  </body>
</html>
```



2.2 Addressing Modes

- Addressing modes are used to specify location of the pages in the web.
- There are two types of URLs:
 - a. Absolute URLs.
 - b. Relative URLs.

a. Absolute URLs

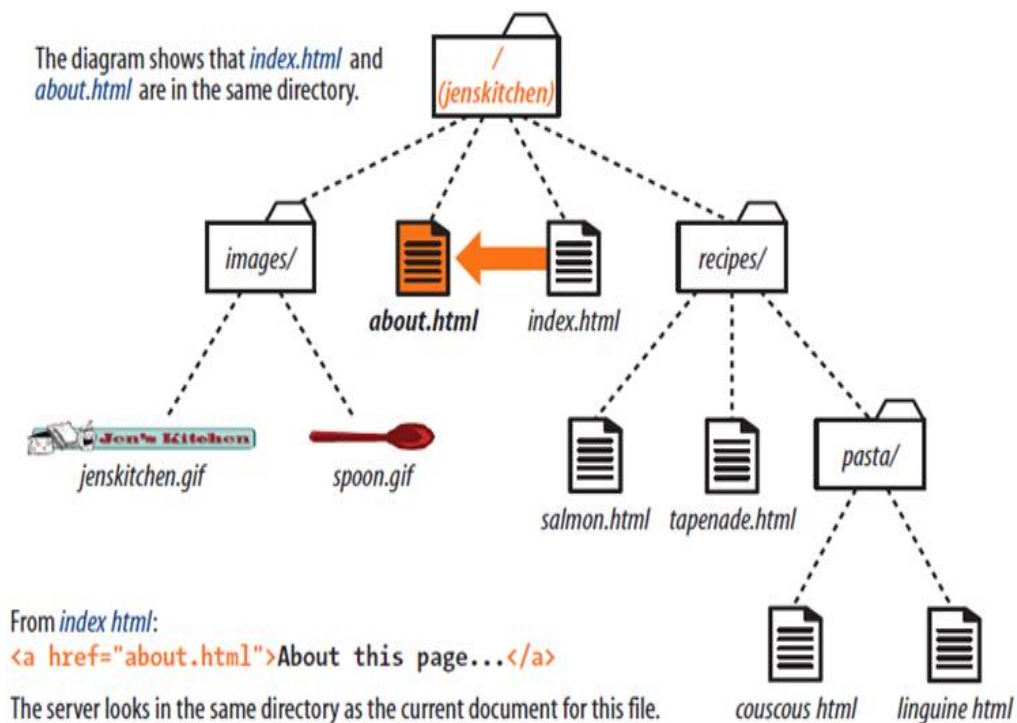
- Provide the full URL for the document, including the **protocol (http://)**, the **domain name**, and the **pathname as necessary**.

- You need to use an absolute URL when **pointing to a document out on the Web** (i.e., not on your own server), example: href="http://www.google.com".

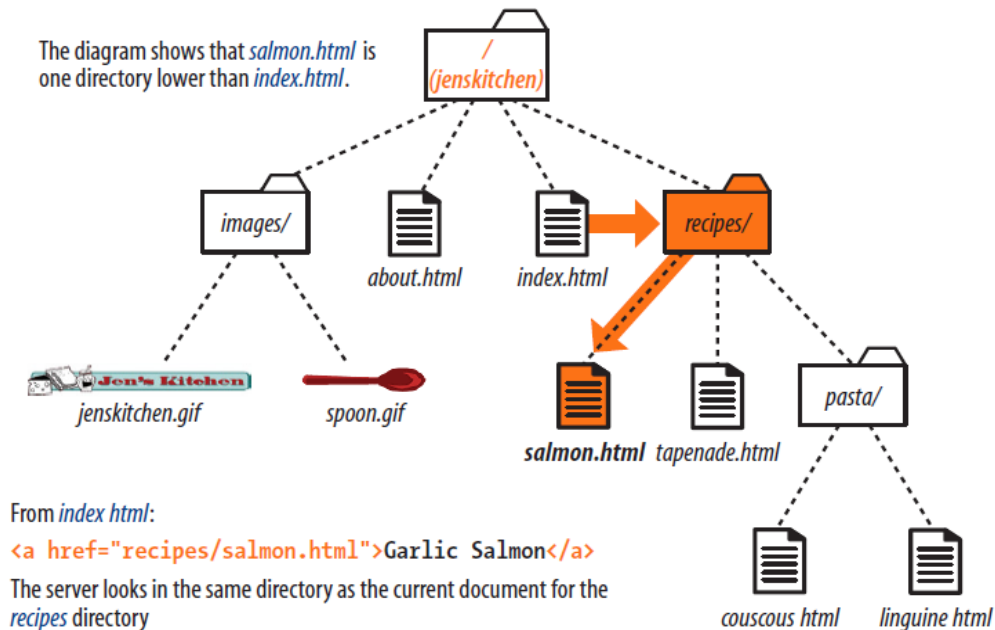
b. Relative URLs

- Describe the **pathname** to a file relative to the current document.
- Relative URLs can be used when you are **linking to another document on your own site** (i.e., on the same server).
- It **doesn't require the protocol or domain name**, just the **pathname**.
- **Example: href="recipes/index.html"**
- There are many ways to link pages using relative URLs:
 1. Linking to page in the same directory.
 2. Linking to lower directory.
 3. Linking to higher directory.
 4. Site root relative pathname.

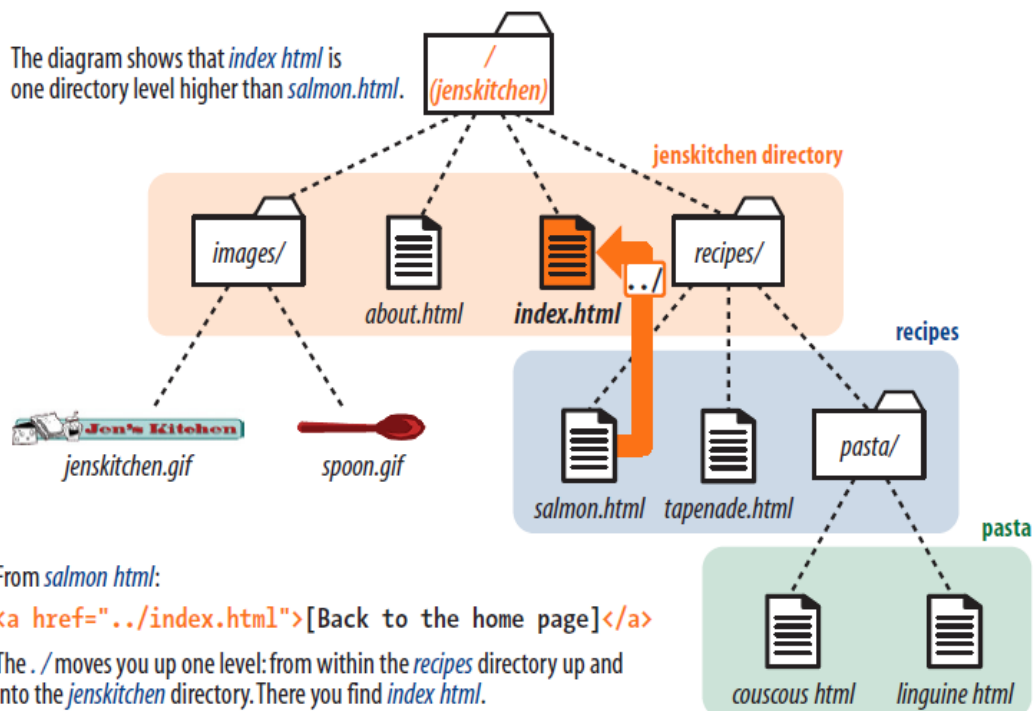
1. Linking to page in the same directory.



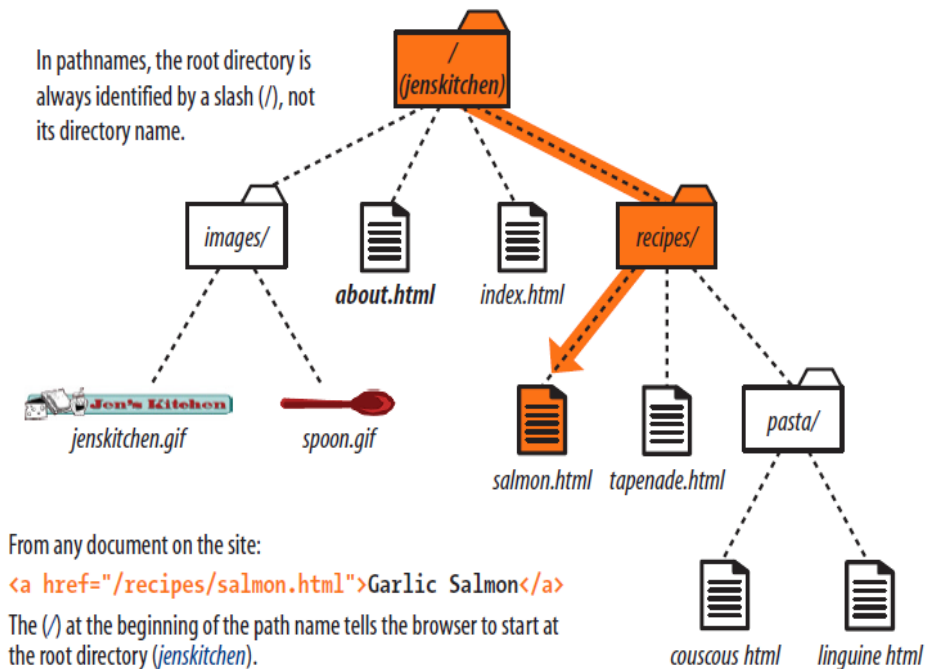
2. Linking to lower directory.



3. Linking to higher directory.



4. Site root relative pathname.



2.3 Linking to Document Fragment

- Linking to specific point in a page is useful for providing **shortcuts** to **information** at the bottom of a long, scrolling page or for **getting back** to the **top** of a page with just one click or tap.
- Linking to a specific point in the page is also **referred** to as linking to a **document fragment**.
- To link a page to another point in the same page or another page:
 - a. Link to Specific Point in the Same Page
 - b. Link to Specific location in another Page.

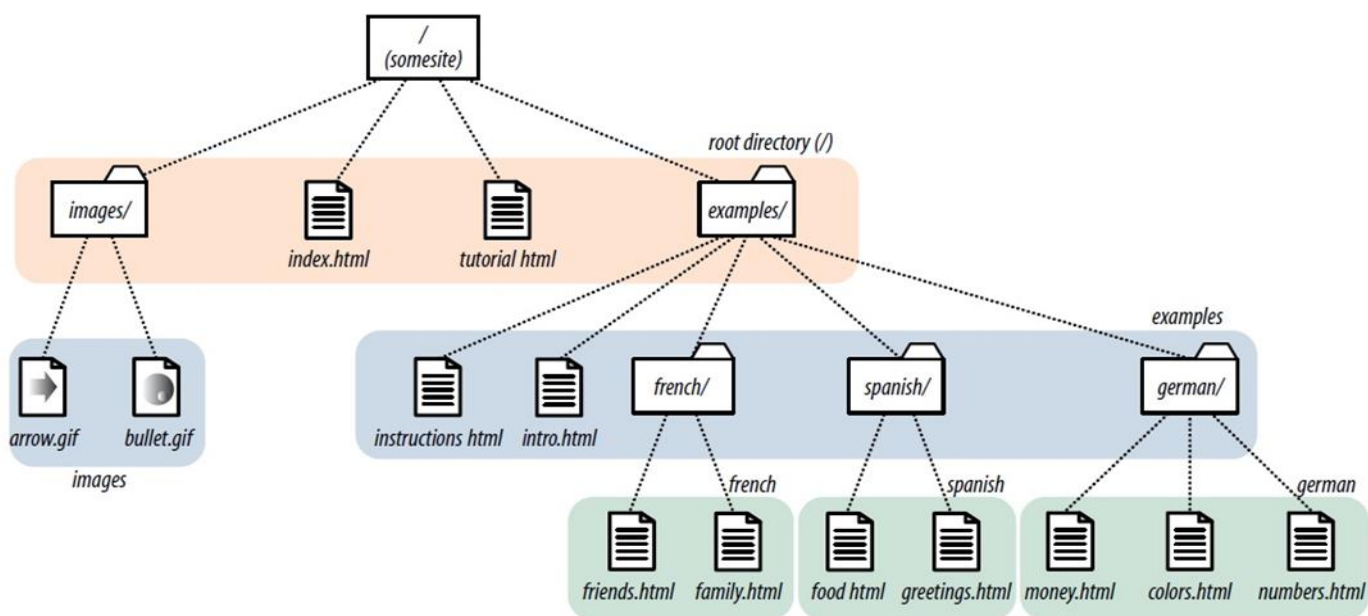
a. Link to Specific Point in the Same Page

- Linking to a particular spot within a page is a **two-part** process.
- First, **you identify the destination**, and then you **make a link to it**.

- *Identifying the destination:* create a destination, use the **id** attribute to give the target element in the document a **unique name**.
- *Linking to the destination:* use the **a** element with the **href** attribute to provide the location of the link with # to **id** in the first step.
- **Exercise:** use link in the same page.

b. Link to Specific location in another Page.

- You can link to a fragment in another document by **adding the fragment name** to the end of the URL (**absolute or relative**).
- **Example:** `Go to middle of exam`.
- If the fragments in external documents move or go away, the page will still load; the browser will just go to the top of the page as it does for regular links.
- **Exercise:** if you have the web site files as shown below, create a link back to the home page (index.html) from the page instructions.html.
- In the file intro.html, create a link to the website for this book (www.learningwebdesign.com/4e/materials).
- Create a link to instructions.html from the page greetings.html.
- Create a link back to the home page (index.html) from money.html.



2.4 Lists

- HTML provides elements for marking up three types of lists:

a. Unordered lists

b. Ordered lists

c. Description lists

a. Unordered Lists

- To identify an unordered list, mark it up as a **ul** element, the opening **** tag goes before the first list item, and the closing tag **** goes after the last item.
- Each item in the list gets marked up as a **list item ** by enclosing it in opening and closing **** tags.
- By default, unordered lists display with a **bullet** before each list item, but you can change that with a **style sheet**.
- **Attribute type** can be used to change the shape of marked items types are (**circle, square, disc and none**).
- Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Unordered List</title>
  </head>
  <body>
    <h1> list of fonts:</h1>
    <ul>
      <li>Arial</li>
      <li>Calibri</li>
      <li>Times New Roman</li>
      <li>Andalus</li>
```

```
</ul>
</body>
</html>
```

List of fonts:

- Arial
- Calibri
- Times New Roman
- Andalus

b. Ordered Lists

- Ordered lists are for items that occur in a **particular order**, such as **step by step** instructions or **driving directions**.
- They work just like the unordered lists described earlier, except they are defined with the **ol** element (for **ordered list**, of course).
- Instead of **bullets**, the browser automatically inserts **numbers** before ordered list items, so you don't need to number them in the source document.
- To identify an ordered list, mark it up as a **ol** element, the opening **** tag goes before the first list item, and the closing tag **** goes after the last item.
- Each item in the list gets marked up as a list item **** by enclosing it in opening and closing **** tags.
- Default type of ordering in attribute type is **numbers**.
- Types are ("A", "a", "1", "I" and "i"). Exercise
- **Example:** write a complete HTML page that produces the following list:

Important key points to success :

1. Forget the past and live in present
2. There is no short cut to success
3. Learn new things
4. Strive to achieve the goal in certain time
5. Formulate your plans

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ordered List</title>
  </head>
  <body>
    <h1> Important key points to success :</ h1>
    <ol>
      <li>Forget the past and live in present</li>
      <li>There is no shortcut to success</li>
      <li>Learn new things</li>
      <li>Strive to achieve the goal in certain time</li>
      <li> Formulate your plan</li>
    </ol>
  </body>
</html>
```

c. Description Lists

- Description lists are used for any type of **name/value pairs**, such as **terms** and their definitions, questions and answers, or other types of terms and their associated information.
- The whole **description list** is marked up as a **dl** element; the content of a **dl** is some number of **dt** elements indicating the names (**description term**) and **dd** elements for their respective values (**description definition**).

- **Example:**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Description List</title>
  </head>
<body>
  <h1> Important commands in CMD</h1>
  <dl>
    <dt> recimg</dt>
    <dd>Create custom recovery images </dd>
    <dt> sfc /scannow</dt>
    <dd> Scan System files for problems</dd>
    <dt>telnet </dt>
    <dd>connect to telnet server </dd>
    <dt> cipher</dt>
    <dd>Permanently delete and overwrite a directory</dd>
  </dl>
</body>
</html>
```

Important commands in CMD

recimg	Create custom recovery images
sfc /scannow	Scan System files for problems
telnet	Connect to telnet server
cipher	Permanently delete and overwrite a directory

Contents

3.1 Tables	2
3.2 Table Structure.....	2
3.3 Spanning cells.....	3
3.3.1 Column spanning	3
3.3.2 Row spans	4
3.4 Table Caption.....	5
3.5 Table border	6
3.6 Cell Spacing and Padding	7
3.7 Background Color and Background	7
3.8 Table Height and Width.....	8
3.9 Table Align.....	9

3.1 Tables

- The HTML `<table>` element represents tabular data that is, information presented in a two-dimensional table comprised of rows and columns of cells containing data.
- There are many attributes related to table tag such as (align, border, width, height, bgcolor, bordercolor, and background).

3.2 Table Structure

- Tables were created for instances when you need to add tabular material (data arranged into rows and columns) to a web page.
- Tables may be used to organize calendars, schedules, statistics, or other types of information, a table cell may contain any sort of information, including numbers, text elements, and even images and multimedia objects.
- Tables are defined with the `<table>` tag, a table is divided into rows (with the `<tr>` tag), and each row is divided into data cells (with the `<td>` tag), td stands for "**table data**," and holds the content of a data cell. A `<td>` tag can contain text, links, images, lists, forms, other tables, etc.

Menu item	Calories	Fat (g)
Chicken Noodle Soup	120	2
Caesar Salad	400	26

- **Example:**

```
<table border=1>
<tr><th>Seq</th><th>Student Name</th><th>ID</th></tr>
<tr><td>1</td><td>Ali</td><td>0990909</td></tr>
<tr><td>2</td><td>Mohammed</td><td>0990901</td></tr>
</table>
```

Seq	Student Name	ID
1	Ali	0990909
2	Mohammed	0990939

- **Border in the previous table is added (will explain later).**
- `<th>` for "table headers" inside table row `<tr>`.
- The default view of table in the browser will be without borders.
- Table headers are important because they provide information or context about the cells in the row or column they precede.
- The **th** element may be handled differently than **tds** by alternative browsing devices.
- Table header has default rendering (bold and centered).

3.3 Spanning cells

- Cell spanning is the stretching of a cell to cover several rows or columns.
- Spanning cells allows you to create complex table structures, but it has the side effect of making the markup a little more difficult to keep track of.

3.3.1 Column spanning

- Column spans, created with the **colspan** attribute in the **td** or **th** element, stretch a cell to the right to span over the subsequent columns.
- **Example:**

```
<table border=1>
<tr><th colspan=2>College of CS&IT </th></tr>
<tr><td>CS Department</td><td>IS Department</td></tr>
</table>
```

College of CS&IT	
CS Department	IS Department

- Notice in the first row (**tr**) that there is only one **th** element, while the second row has two **td** elements.
- The **th** for the column that was spanned over is no longer in the source; the cell with the **colspan** stands in for it.
- Every row should have the same number of cells or equivalent **colspan** values. For example, there are two **td** elements and the **colspan** value is 2, so the implied number of columns in each row is equal.

3.3.2 Row spans

- Row spans, created with the **rowspan** attribute, work just like column spans, but they cause the cell to span downward over several rows.
- **Example:**

```
<table border=1>
<tr><th rowspan=4>Seasons</th><td>Winter</td></tr>
<tr><td>Autumn</td></tr>
<tr><td>Summer</td></tr>
<tr><td>Spring</td></tr>
</table>
```

Seasons	Winter
	Autumn
	Summer
	Spring

- Notice that the **td** elements for the cells that were spanned over (the first cells in the remaining rows) do not appear in the source.
- The **rowspan="4"** implies cells for the subsequent two rows, so no **td** elements are needed.
- **Exercise:** write HTML doc to show your weekly schedule.
- **Exercise:** write HTML doc to show the table below:

3.4 Table Caption

- Use the **caption** element to give a table a title or brief description that displays next to the table, you can use it to describe the table's contents or provide hints on how it is structured.
- The caption is displayed above the table by default.
- **Example:**

```

<table border=1>
<caption> Four Seasons</caption>
<tr><th rowspan=4>Seasons</th><td>Winter</td></tr>
<tr><td>Autumn</td></tr>
<tr><td>Summer</td></tr>
<tr><td>Spring</td></tr>
</table>

```

Four Seasons

Seasons	Winter
	Autumn
	Summer
	Spring

3.5 Table border

- If you do not specify a border for the table, it will be displayed without borders (border=0).
- A border can be added using the **border** attribute.
- This integer attribute defines, in pixels, the size of the frame surrounding the table.
- **Bordercolor** attribute used to set color for table border.
- **Example:**

```
< table border=3 bordercolor=red>
<caption> Four Seasons</caption>
<tr><th rowspan=4>Seasons</th><td>Winter</td></tr>
<tr><td>Autumn</td></tr>
<tr><td>Summer</td></tr>
<tr><td>Spring</td></tr>
</table>
```

Four Seasons

Seasons	Winter
	Autumn
	Summer
	Spring

3.6 Cell Spacing and Padding

- There are two attributes called *cellpadding* and *cellspacing* which you will use to adjust the white space in your table cells.
- The *cellspacing* attribute defines the width of the border.
- *Cellpadding* represents the distance between cell borders and the content within a cell.
- This pixel-sized space will be applied on all four sides; if it is a percentage length, the content will be centered and the total vertical space (top and bottom) will represent this percentage.

3.7 Background Color and Background

- **Bgcolor** attribute defines the background color of the table as well as the whole body of html page and its content.
- It is one of the 6-digit hexadecimal code as defined in RGB, prefixed by a '#'.
- One of the sixteen predefined color strings may be used:

"green = "#008000		"black = "#000000	
"lime = "#00FF00		"silver = "#C0C0C0	
"olive = "#808000		"gray = "#808080	
"yellow = "#FFFF00		"white = "#FFFFFF	
"navy = "#000080		"maroon = "#800000	
"blue = "#0000FF		"red = "#FF0000	
"teal = "#008080		"purple = "#800080	
"aqua = "#00FFFF		"fuchsia = "#FF00FF	

- **Example:**

```
<table border=3 bordercolor=red bgcolor=lightblue cellspacing=5  
cellpadding=5>  
<Caption> Four Seasons</caption>  
<tr><th rowspan=4>Seasons</th><td>Winter</td></tr>  
<tr><td>Autumn</td></tr>  
<tr><td>Summer</td></tr>  
<tr><td>Spring</td></tr>  
</table>
```

Four Seasons

Seasons	Winter
	Autumn
	Summer
	Spring

- Background attribute can set background image for whole table or just for one cell.
- Background attribute can also applied to whole body of the html page.
- Background attribute takes the path of the image.

3.8 Table Height and Width

- You can set a table width and height using width and height attributes.
- You can specify table width or height in terms of pixels or in terms of percentage of available screen area.

3.9 Table Align

- This enumerated attribute indicates how the table must be aligned inside the containing document. It may have the following values:
 - Left: the table is displayed on the left side of the document.
 - Center: the table is displayed in the center of the document.
 - Right: the table is displayed on the right side of the document.
- **Example:**

```
<table border=3 height=200 width=200 align =center background=1.JPG>  
<tr><th rowspan=4>Seasons</th><td>Winter</td></tr>  
<tr><td>Autumn</td></tr>  
<tr><td>Summer</td></tr>  
<tr><td>Spring</td></tr>  
</table>
```



Contents

4.1 Forms.....	2
4.2 Form Works	2
4.3. Form's Work Procedure	2
4.3.1 The Action Attribute.....	4
4.3.2 The Method Attribute.....	5
4.3.3 Target Attribute	6
4.3.4 The Name Attribute.....	6
4.4. Form Elements	6
4.4.1 Text Entry Controls.....	7
4.4.2 Password Text	8
4.4.3 Reset and Submit Buttons	9
4.4.4. Radio and checkbox buttons.....	10
4.4.5 Menus	11
4.4.6 File selection control	13
4.4.7 Hidden Controls	14
4.5 Form Accessibility Features	14
4.6 Fieldset and legend.....	16

4.1 Forms

It didn't take long for the web to shift from a network of pages to read to a place where you went to get things done making purchases, booking plane tickets, signing petitions, searching a site, posting a tweet... etcetera. All of these interactions are handled by forms. Forms are used to collect user input.

<Form>

-

Form Elements

-

</form>

4.2 Form Works

There are two parts to a working form. The first part is the *form* that you see on the page itself that is created using HTML markup. Forms are made up of buttons, input fields, and drop-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.

4.3. Form's Work Procedure

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 many forms.

1. 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 renders them with the appropriate form controls on the page, including two text entry fields and a submit button shown in Figure (4.1).

2. 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 hitting the “Submit” button.
3. The browser collects the information she entered, encodes it and sends it to the web application on the server.
4. The web application accepts the information and processes it. In this example, the name and email address are added to a database.
5. 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 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.
6. 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.

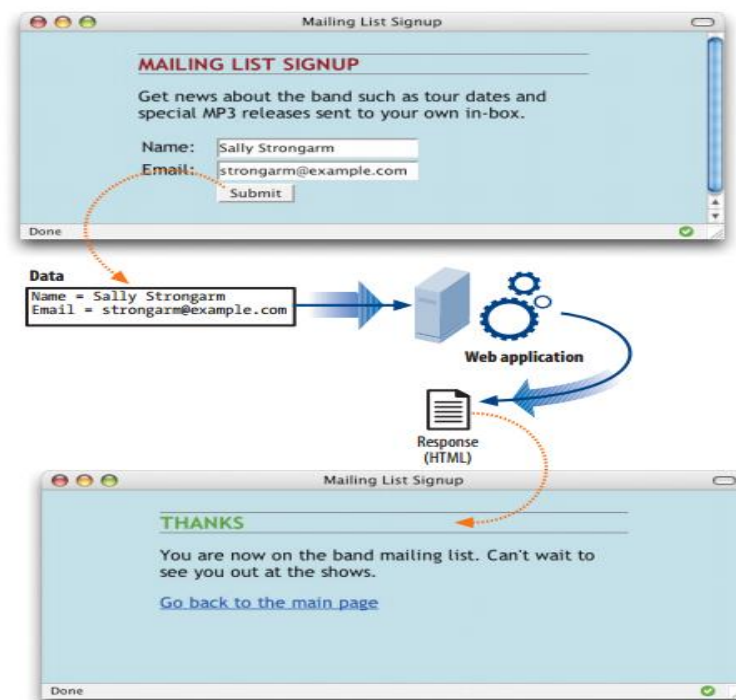


Figure (4.1): Work of Form

4.3.1 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.php*

```
<form action="/mailinglist.php" method="post">...</form>
```

The *.php* suffix indicates that this form is processed by a script written in the PHP scripting language, but web forms may be processed using one of the following technologies:

- PHP (*.php*) is an open source scripting language most commonly used with the Apache web server.
- Microsoft's ASP.NET (Active Server Pages) (*.asp*) is a programming environment for the Microsoft Internet Information Server (IIS).
- Ruby is the programming language that is used with the Rails platform. Many popular web applications are built with it (*.rb* extension).
- JavaServer Pages (*.jsp*) is a Java-based technology similar to ASP.
- Python is a popular scripting language for web and server applications (*.py* extension).

Sometimes there is form-processing code such as PHP embedded right in the HTML file. In that case, leave the action empty and the form will post to the page itself.

4.3.2 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 figure (4.1)

username = Sally Strongarm

email = strongarm@example.com

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

username=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. The method is optional and will default to GET if omitted. Our example uses the POST method, as shown here:

<form action="/cgi-bin/maillinglist.pl" method="POST">...</form>

A. 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.

B. 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, GET may not be used when the form is used to upload a file. The length of a URL is limited (about 3000 characters).

4.3.3 Target Attribute

The target attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window. The default value is "_self" which means the form will be submitted in the current window.

To make the form result open in a new browser tab, use the value "_blank":

<form action="/action_page.php" target="_blank">

4.3.4 The Name Attribute

The name attributes provides the variable name for the control. You can't just name controls willy-nilly. The web application that processes the data is programmed to look for specific variable names. Each variable must be named uniquely, that is, the same name may not be used for two variables. You should also avoid putting character spaces in variable names; use an underscore or hyphen instead.

4.4. Form Elements

There are many form elements such as:

1. Text entry controls and Label
2. Submit and reset buttons
3. Radio and checkbox buttons

4. Pull-down and scrolling menus
5. File selection and upload control
6. Hidden controls
7. Dates and times (HTML5)
8. Numerical controls (HTML5)
9. Color picker control (HTML5)
10. Output, option, button, and optgroup.

4.4.1 Text Entry Controls

One of the most common tasks in a web form is to enter text information. Which element you use depends on whether users are asked to enter a single line of text (input) or multiple lines (textarea).

A. Single-line Text Field

One of the most straightforward form input types is the text entry field used for entering a single word or line of text. In fact, it is the default input type, which means it is what you'll get if you forget to include the type attribute or include an unrecognized value. Add a text input field to a form with the input element with its type attributes set to text.

```
<label>City</label><input name="city" value="Your Hometown" maxlength=50  
type="text">
```

City

There are a few attributes in there I'd like to point out.

- *name*

The name attribute is required for indicating the variable name.

- *value*

The value attribute specifies default text that appears in the field when the form is loaded. When you reset a form, it returns to this value.

- *maxlength*

By default, users can type an unlimited number of characters in a text field regardless of its size (the display scrolls to the right if the text exceeds the character width of the box). You can set a maximum character limit using the `maxlength` attribute if the forms processing program you are using requires it.

There are other input attributes such as (`min`, `max`, `size`, `disabled`, `pattern`, `placeholder`, `required`, `step`, `autofocus`, `list`, and `autocomplete`).

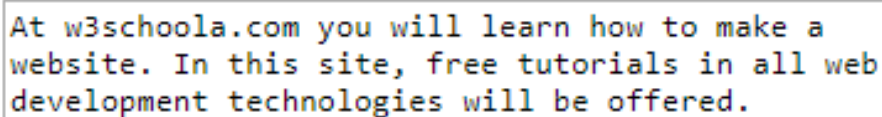
B. Multi-line text field

The `<textarea>` tag defines a multi-line text input control. A text area can hold an unlimited number of characters, and the text renders in a fixed-width font (usually Courier). The size of a text area can be specified by the `cols` and `rows` attributes, or even better; through CSS' height and width properties.

```
<textarea rows=4 cols=50>
```

At w3schoola.com you will learn how to make a website. In this site, free tutorials in all web development technologies will be offered.

```
</textarea>
```



4.4.2 Password Text

Input type can be set into password to allow user to input field password. When input password used, it make the text shape set into stars, for example:

```
<form>
```

```
Username: <br> <input type="text" name="username"><br>
```

Password:
 <input type="Password" name="password"></form>

Username:

Password:

4.4.3 Reset and Submit Buttons

In this section, reset and submit buttons will be explains.

A. Reset button

Reset button used to reset the entries in the form into the default value. It is used in most times to clear the fields in the form controls. Example:

```
<form action="demo.php">
```

```
Email:<input name="email" type="text"><br>
```

```
Pin:<input name="pin" type="password" maxlength=4><br>
```

```
<input type="reset" value="reset">
```

```
</form>
```

Email:

Pin:

B. Submit Button

Defines a button for submitting a form to a form-handler. The form-handler is typically a server page with a script for processing input data. The form-handler is specified in the form's **action** attribute, for example:

```
<form action="demo.php">
```

```
First name:<br>
```

```
<input name="first" type="text" value="Mickey"><br>
```

Last name:


```
<input name="pin" type="text" value="Mouse"><br><br>
```

```
<input type="submit" name="Submit">
```

```
</form>
```

First name:

Mickey

Last name:

Mouse

Submit

4.4.4. Radio and checkbox buttons

Both checkbox and radio buttons make it simple for your visitors to choose from a number of provided options. They are similar in that they function like little on/off switches. A form control made up of a collection of radio buttons is appropriate when only one option from the group is permitted in other words, when the selections are mutually exclusive (such as Yes or No, or Male or Female). When one radio button is “on,” all of the others must be off. When checkboxes are grouped together, however, it is possible to select as many or as few from the group as desired. This makes them the right choice for lists in which more than one selection is okay, for example:

```
<form action="demo.php">
```

```
<input type="checkbox" name="vehicle1" value="Bike">I have a bike<br>
```

```
<input type="checkbox" name="vehicle2" value="Car" checked>I have a car<br>
```

```
<input type="submit" value="Submit">
```

```
</form>
```

☐ I have a bike
☒ I have a car

Radio buttons are added to a form using the input element with the type attribute set to radio. Here is the syntax for a minimal radio button:

```
<input type="radio" name="variable" value="value">
```

The name attribute is required and plays an important role in binding multiple radio inputs into set. When you give a number of radio button inputs the same name value (age in the following example), they create a group of mutually exclusive options, for example:

```
<form>  
<input type="radio" name="gender" value="male" checked>Male<br>  
<input type="radio" name="gender" value="female" >Female<br>  
</form>
```

☒ Male
☐ Female

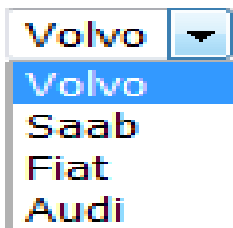
4.4.5 Menus

Another way to provide a list of choices is to put them in a drop-down or scrolling menu. Menus tend to be more compact than groups of buttons and checkboxes. You add both drop down and scrolling menus to a form with the select element. Whether the menu pulls down or scrolls is the result of how you specify its size and whether you allow more than one option to be selected.

4.5.1 Drop-down menus

The select element displays as a drop-down menu (also called a pull-down menu) by default when no size is specified or if the size attribute is set to 1. In pull-down menus, only one item may be selected. Here's an example:

```
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select>
```



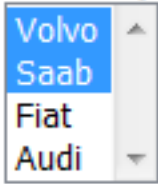
4.5.2 Scrolling menus

To make the menu display as a scrolling list, simply specify the number of lines you'd like to be visible using the size attribute. This example menu has the same options as the previous one, except it has been set to display as a scrolling list that is six lines.

```
<p> What is your favorite  car></p>
<select name="cars" size=4 multiple="multiple">
<option value="volvo" selected>Volvo</option>
<option value="saab" selected>Saab</option>
<option value="fiat" >Fiat</option>
```

```
<option value="audi" >Audi</option>
</select>
```

What is your favorite car?



The multiple attribute allows users to make more than one selection from the scrolling list. Note that pull-down menus do not allow multiple selections; when the browser detects the multiple attribute; it displays a small scrolling menu automatically by default. Use the selected attribute in an option element to make it the default value for the menu control. Selected options are highlighted when the form loads. The selected attribute can be used with pull-down menus as well.

4.4.6 File Selection Control

Web forms can collect more than just data. They can also be used to transmit external documents from a user's hard drive. For example, a printing company could use a web form to upload artwork for a business card order. A magazine could use a form on their site to collect digital photos for a photo contest. The file selection control makes it possible for users to select a document from the hard drive to be submitted with the form data. It is added to the form using our old friend the input element with its type set to file.

```
<form action="process.php" method="post" enctype="multipart/form-data">
<label>Send a photo to be used as your online icon<em>(optional)</em></label>
<input name="photo" size=28 type="file">
</form>
```

Send a photo to be used as your online icon (*optional*)

It is important to note that when a form contains a file selection input element, you must specify the encoding type (enctype) of the form as multipart/form-data and use the POST method. The size attribute in this example sets the character width of the text field (although it could also be controlled with a CSS rule) if the browser displays one.

4.4.7 Hidden Controls

There may be times when you need to send information to the form processing application that does not come from the user. In these instances, you can use a hidden form control that sends data when the form is submitted, but is not visible when the form is displayed in a browser. Hidden controls are added using the input element with the type set to hidden. Its sole purpose is to pass a name/value pair to the server when the form is submitted. In this example, a hidden form element is used to provide the location of the appropriate thank-you document to display when the transaction is complete.

```
<input name="secret" type="hidden" value=http://www.secretdata.com>
```

- There are other input types such as (button, color, week, month, time, date, datetime-local, email, image, number, range, search, tel, and URL).

4.5 Form Accessibility Features

It is essential to consider how users without the benefit of visual browsers will be able to understand and navigate through your web forms. The label, fieldset, and legend form elements improve accessibility by making the semantic connections between the components of a form clear. The resulting markup is not only more semantically rich, but there are also more elements available to act as “hooks” for style sheet rules.

Although we may see the label “Address” right next to a text field for entering an address in a visual browser, in the source, the label and field input may be separated.

```
<input name="sport1" type="checkbox" value="football">
<label>Football</label><br>
<input name="sport2" type="checkbox" value="volleyball">
<label>Volleyball</label><br>
<input name="sport3" type="checkbox" value="basketball">
<label>Basketball</label>
```

The label element associates descriptive text with its respective form field. This provides important context for users with speech based browsers. Each label element is associated with exactly one form control. There are two ways to use it. One method, called implicit association, nests the control and its description within a label element. In the following example, labels are assigned to individual checkboxes and their related text descriptions. (By the way, this is the way to label radio buttons and checkboxes. You can’t assign a label to the entire group).

☐ Football
☐ Volleyball
☐ Basketball

The other method, called explicit association, matches the label with the control’s id reference. For attribute says which control the label is for. This approach is useful when the control is not directly next to its descriptive text in the source. It also offers the potential advantage of keeping the label and the control as two distinct elements, which may come in handy when aligning them with style sheets.

```
<label for="login-user">Login account</label>
<input name="user" id="login-user" type="text">
<label for="login-password">Password</label>
```

```
<input name="password" id="login-password" type="password">
```

Login account Password

Another advantage to using labels is that users can click or tap anywhere on them to select the form element. Users with touch devices will appreciate the larger tap target.

4.6 Fieldset and legend

The fieldset element indicates a logical group of form controls. A fieldset may also include a legend element that provides a caption for the enclosed fields.

```
<fieldset>
  <legend>Information:</legend>
  <label>Name:</label><input type="text" name="name"><br>
  <label>Email:</label><input type="text" name="email"><br>
  <label>Date of birth:</label><input type="text" name="date">
</fieldset>
```

Information:

Name:

Email:

Date of birth:

Contents

5.1 Cascading Style Sheet (CSS).....	2
5.2 CSS Benefits	2
5.3 How Style Sheets Work	2
5.4 Writing the Rules	2
5.4.1 Selectors.....	3
5.4.2 Declarations	3
5.5 Attaching the Styles to the Document	4
5.6 CSS Selectors	7
5.6.1 The Element Selector.....	7
5.6.2 The id Selector	8
5.6.3 The class Selector	9
5.6.4 Grouping Selectors	10
5.7 Cascading Order.....	10
5.8 CSS Colors	11
5.9 CSS Text.....	12
5.9.1 Text Color.....	12
5.9.2 Text Alignment	13
5.9.3 Text Decoration.....	13
5.9.4 Text Transformation	14
5.9.6 Letter Spacing.....	14
5.9.7 Line Height	15
5.9.8 Text Direction	15
5.9.9 Word Spacing	15
5.10 CSS Borders.....	16
5.10.1 Border Style	16
5.10.2 Border Width	16
5.10.3 Border Color	17
5.10.4 Border - Individual Sides	17

5.1 Cascading Style Sheet (CSS)

CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External stylesheets are stored in CSS files. CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

5.2 CSS Benefits

- **Precise type and layout controls.** You can achieve print-like precision using CSS. There is even a set of properties aimed specifically at the printed page.
- **Less work.** You can change the appearance of an entire site by editing one style sheet.
- **More accessible sites.** When all matters of presentation are handled by CSS, you can mark up your content meaningfully, making it more accessible for non-visual or mobile devices.
- **Reliable browser support.** Every browser in current use supports CSS Level 2 and many cool parts of CSS Level 3.

5.3 How Style Sheets Work

- a. Markup document with HTML tags.
- b. Write style rules for how you'd like certain elements to look.
- c. Attach the style rules to the document. When the browser displays the document, it follows your rules for rendering elements (unless the user has applied some mandatory styles).

5.4 Writing the Rules

A style sheet is made up of one or more style instructions (called rules or rule sets) 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're 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; }
```

In CSS terminology, the two main sections of a rule are the selector that identifies the element or elements to be affected, and the declaration that provides the rendering instructions. The declaration, in turn, is made up of a property (such as color) and its value (green), separated by a colon and a space. One or more declarations are placed inside curly brackets, as shown in (Figure 1.1).

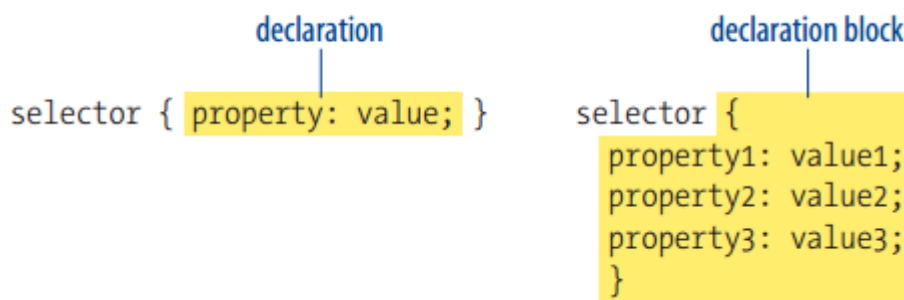


Figure (5.1): CSS Rule

5.4.1 Selectors

In the previous small style sheet example, the h1 and p elements are used as selectors. This is called an element type selector, and it is the most basic type of selector. The properties defined for each rule will apply to every h1 and p element in the document, respectively.

5.4.2 Declarations

The declaration is made up of a property/value pair. There can be more than one declaration in a single rule; for example, the rule for the p element shown earlier in

the code example has both the font-size and font-family properties. Each declaration must end with a semicolon to keep it separate from the following declaration (see note). If you omit the semicolon, the declaration and the one following it will be ignored. The curly brackets and the declarations they contain are often referred to as the declaration block. Because CSS ignores whitespace and line returns within the declaration block, authors typically write each declaration in the block on its own line, as shown in the following example. This makes it easier to find the properties applied to the selector and to tell when the style rule ends.

```
p { font-size: small;

    font-family: sans-serif;

}
```

Values are dependent on the property. Some properties take length measurements, some take color values, and others have a predefined list of keywords. When using a property, it is important to know which values it accepts; however, in many cases, simple common sense will serve you well.

5.5 Attaching the Styles to the Document

There are three ways that style information can be applied to an HTML document:

a. 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 HTML documents. In this way, all the files in a website may share the same style sheet. This is the most powerful and preferred method for attaching style sheets to content.

Example:

```
<!DOCTYPE html>

<html>

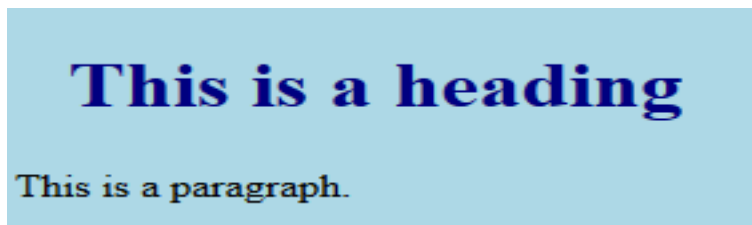
<head>

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

</head>

<title>External CSS </title>
```

```
</head>
<body>
<h1> This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
mystyle.css
body { background-color: lightblue;}
h1{color: navy;
margin-left: 20px;}
```



b. *Embedded style sheets.* This is the type of style sheet we worked with in the exercise. 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:

```
<head><title>Required document title here</title>
<style>
/* style rules go here */
</style>
</head>
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Embedded CSS </title>
<style>
body { background-color:linen;}
```

```
h1 { color: maroon;
margin-left: 40px;}

</style>

</head>

<body>

<h1> This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>
```

This is a heading

This is a paragraph.

- c. *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. Inline styles are problematic in that they intersperse presentation information into the structural markup. They also make it more difficult to make changes because every style attribute must be hunted down in the source.

Example:

```
<!DOCTYPE html>

<html>

<head><title>Inline CSS </title> </head>

<body>

<h1 style="color:blue; margin-left:30px;"> This is a heading.</h1>

<p>This is a paragraph.</p>
```

</body>

</html>

This is a heading.

This is a paragraph.

5.6 CSS Selectors

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

5.6.1 The Element Selector

The element selector selects elements based on the element name. You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color).

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Element Selector </title>
```

```
<style>
```

```
p { color: red;
```

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

```
</style>
```

```
</head>
```

```
<body>
```

```
<p>This is a paragraph.</p>
```

```
<p> Hello</p>
```

```
</body>
```

```
</html>
```

This is a paragraph.

Hello

5.6.2 The id Selector

The id selector uses the id attribute of an HTML element to select a specific element. The id of an element should be unique within a page, so the id selector is used to select one unique element. To select an element with a specific id, write a hash (#) character, followed by the id of the element. The style rule below will be applied to the HTML element with id "para1" and "para2".

```
<!DOCTYPE html>
<html>
<head>
<title>ID Selector </title>
<style>
#para1 { color: red;
text-align:center;}
#para2 { color: green;
text-align:center;}
</style>
</head>
<body>
<p id="para1">Hello World!</p>
<p id="para2"> This is new paragraph.</p>
</body>
</html>
```

Hello World!

This is new paragraph.

5.6.3 The class Selector

The class selector selects elements with a specific class attribute. To select elements with a specific class, write a period (.) character, followed by the name of the class. In the example below, all HTML elements with class="center" will be red and center-aligned:

```
<!DOCTYPE html>
<html>
<head>
<title>Class Selector </title>
<style>
.center { color: red;
text-align:center;}
</style>
</head>
<body>
<h1 class="center">Red and center-aligned heading</h1>
<p class="center"> Red and center-aligned paragraph. </p>
</body>
</html>
```

Red and center-aligned heading

Red and center-aligned paragraph.

5.6.4 Grouping Selectors

If you have elements with the same style definitions, like this:

```
h1 {  
    text-align: center;  
    color: red;  
}  
  
h2 {  
    text-align: center;  
    color: red;  
}  
  
p {  
    text-align: center;  
    color: red;  
}
```

It will be better to group the selectors, to minimize the code. To group selectors, separate each selector with a comma. In the example below we have grouped the selectors from the code above:

```
h1, h2, p {  
    text-align: center;  
    color: red;  
}
```

5.7 Cascading Order

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

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style (inside a specific 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 a browser default value.

Example: Assume that an external style sheet has the following style for the <h1> element: **h1 { color: Red;}** then, assume that an internal style sheet also has the following style for the <h1> element: **h1 { color: orange; }**, if the internal style is defined after the link to the external style sheet, the <h1> elements will be "orange":

```
<!DOCTYPE html>
<html>
<head>
<title>Cascading Order</title>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style> h1 { color: orange;} </style>
</head>
<body>
<h1 > This is heading</h1>
</body></html>
```

5.8 CSS Colors

Colors are displayed combining RED, GREEN, and BLUE. Colors in CSS are most often specified by:

- a valid color name - like "red"
- an RGB value - like "rgb(255, 0, 0)"
- a HEX value - like "#ff0000"

Colors set by using color names (red, green, blue, yellow, orange .. etc). Color names are case-insensitive: "Red" is the same as "red" or "RED". HTML and CSS supports 140 standard color names.

RGB color values can be specified using this formula: `rgb(red, green, blue)`. Each parameter (red, green, blue) defines the intensity of the color between 0 and 255. For example, `rgb(255,0,0)` is displayed as red, because red is set to its highest value (255) and the others are set to 0.

RGB values can also be specified using **hexadecimal** color values in the form: `#RRGGBB`, where RR (red), GG (green) and BB (blue) are hexadecimal values between 00 and FF (same as decimal 0-255). For example, `#FF0000` is displayed as red, because red is set to its highest value (FF) and the others are set to the lowest value (00). **Note:** HEX values are case-insensitive: `"#ff0000"` is the same as `"FF0000"`.

Example:

```
<h2 style="background-color:red">Red background-color</h2>
```

```
<h2 style="background-color:#FF0000">Red background-color</h2>
```

```
<h2 style="background-color:rgb(255, 0, 0)">Background-color set by using rgb(255, 0, 0)
</h2>
```

5.9 CSS Text

There are many properties that can be used to form text such as:

5.9.1 Text Color

The color property is used to set the color of the text. With CSS, a color is most often specified by:

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

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

Example:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Text Color</title>
<style>
body {color:blue;}
h1 {color:green;}
</style>
</head>
<body>
<h1 >This is heading</h1>
<p > This is an ordinary paragraph. </p>
</body>
</html>
```

This is heading 1

[This is an ordinary paragraph.](#)

5.9.2 Text Alignment

The **text-align** property is used to set the horizontal alignment of a text. A text can be **left or right aligned, centered, or justified**. Left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left. When the text-align property 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).

5.9.3 Text Decoration

The text-decoration property is used to set or remove decorations from text. The value **text-decoration: none;** is often used to remove underlines from links. The other text-decoration values are used to decorate text (**overline, underline and line-through**).

Example:

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Class Selector </title>
<style>
body { color: blue;
text-align:right;}
h1 { color: magenta;
text-align:center;
text-decoration:underline;}
</style>
</head>
<body>
<h1>This is heading</h1>
<p>This is an ordinary paragraph. </p>
</body>
</html>
```

This is heading 1

This is an ordinary paragraph.

5.9.4 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.

5.9.5 Text Indentation

The **text-indent** property is used to specify the indentation of the first line of a text. Text-indent takes a value in pixel.

5.9.6 Letter Spacing

The **letter-spacing** property is used to specify the space between the characters in a text. Letter-spacing takes a value in pixel.

5.9.7 Line Height

The **line-height** property is used to specify the space between lines. Line-height takes a value in pixel.

5.9.8 Text Direction

The **direction** property is used to change the text direction of an element. Direction property takes value (rtl right to left and ltr left to right).

5.9.9 Word Spacing

The **word-spacing** property is used to specify the space between the words in a text. Word-spacing takes a value in pixel.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Class Selector </title>
<style>
p{ color:magenta;
text-indent: 10px;
letter-spacing:3px;
line-height:15px;
direction:ltr;
word-spacing:5px;
}
h1 { text-transform: capitalize;
text-decoration: overline;}
</style>
</head>
<body>
<h1>This is heading</h1>
<p>This is an ordinary paragraph. </p>
```

</body>

</html>

This Is Heading 1

This is an ordinary paragraph.

5.10 CSS Borders

There are many properties that affect the borders such as:

5.10.1 Border Style

The border-style property specifies what kind of border to display. The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- 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
- none - Defines no border
- hidden - Defines a hidden border

The border-style property can have from one to four values (for the **top border, right border, bottom border, and the left border**).

5.10.2 Border Width

The **border-width** property specifies the width of the four borders. The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: **thin, medium, or thick**. The border-width property can have from one to four values (for the **top border, right border, bottom border, and the left border**).

5.10.3 Border Color

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

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

The border-color property can have from one to four values (**for the top border, right border, bottom border, and the left border**).

5.10.4 Border - Individual Sides

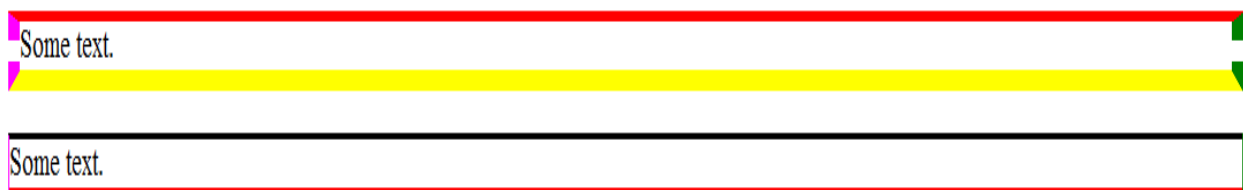
It is possible to specify a different border for each side. In CSS, there are also properties for specifying each of the borders (**top, right, bottom, and left**). You can set the top style border as dotted by set (border-top-style:dotted). You can set the right color border to red color by setting (border-right-color: red).

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Class Selector </title>
<style>
p.one { border-style: solid dotted;
border-width:5px 10px 10px;
border-color:red green yellow magenta;}
p.two { border-style: solid;
border-width: medium thin;
border-color:black green red magenta;}
</style>
</head>
```

```
<body>
<p>This property specifies the width of the four borders: </p>
<p class="one">Some text.</p>
<p class="two">Some text.</p>
</body>
</html>
```

This property specifies the width of the four borders:

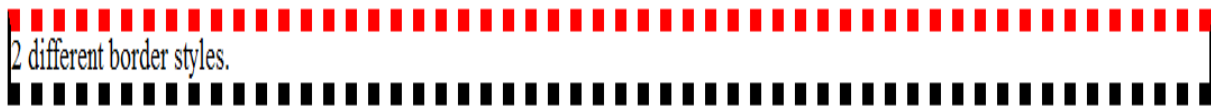


Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Class Selector </title>
<style>
p {border-top-color:red;
border-top-style:dotted;
border-right-style:solid;
border-bottom-style:dotted;
border-left-style:solid;
border-top-width:10px;
border-bottom-width:10px;}
</style>
</head>
<body>
<p class="one">2 different border styles.</p>
```

```
</body>
```

```
</html>
```



Example:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Class Selector </title>
```

```
<style>
```

```
p {border: red 10px solid;}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<p class="one">2 different border styles.</p>
```

```
</body>
```

```
</html>
```

