Web programming

1-Introduction web programming

2- HTML

3- JavaScript

4- ASP

Introduction

- Just as there is a diversity of programming languages available and suitable for conventional programming tasks, there is a diversity of languages available and suitable for Web programming.

- The Internet becomes the main method in exchanging cultures and transferring knowledge between people.

- The Web was originally designed to deliver static Web pages from a Web server connected somewhere on the Internet to a Web browser sitting on a user's desktop computer. Basically, all a user could do was click on a hot spot or hypertext link to retrieve a new page, read it, and then go on to the next page.
- The Web was not designed to support EC sites, especially B2C sites. In its original state, it was not possible to create pages that would allow consumers to easily determine what products were for sale, to select products as they moved from page to page (i.e., an electronic shopping cart), to place an order, or to verify an order. Similarly, there as no simple way to integrate a Web server with a database system containing product, pricing, The Web was originally designed to deliver

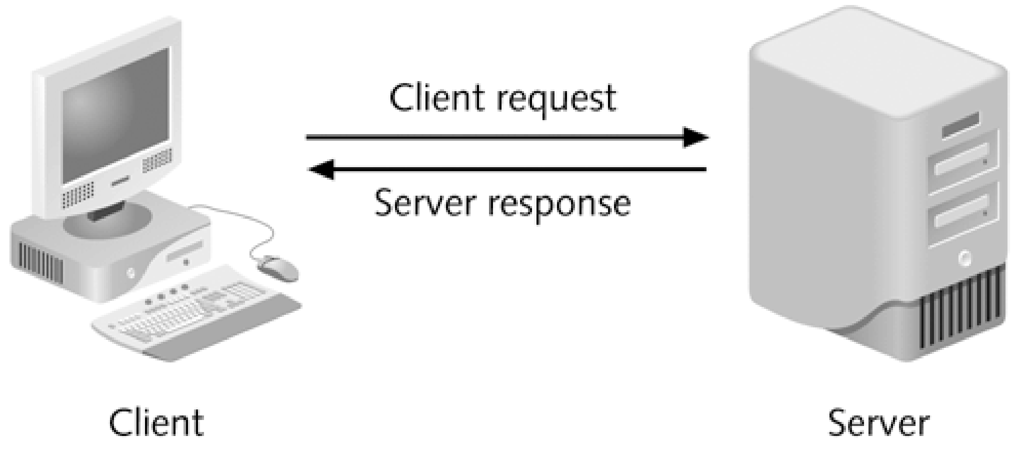
- and promotional data with transactional systems for processing orders and with payment systems for handling credit card purchases and settlements. Over time, these limitations have been addressed. First, forms were added to HTML. Forms provided a way to produce Web pages from which a consumer could select, order, and pay for products. Second, special programming and scripting languages (e.g., Java and JavaScript) were created. These newer languages allowed application developers to produce interactive Web pages whose functionality emulated the rich functionality of standard Windows-based applications. Finally, a standard application programming interface (API), called the common gateway interface (CGI), was introduced. Generally speaking, an API provides a way for one software program to communicate with another,

- whereas CGI provides a way for software developers and application programmers to integrate Web servers with various back-end programs and data sources.
- Because of CGI's inefficiencies, newer APIs and special database gateway programs were also introduced. As a result of these changes, the Web is now well suited for the dynamic world of EC.
- This appendix examines issues of end-user interactivity and dynamic data access. The first sections focus on Java and JavaScript, which are special programming languages that can be used to create Web pages with rich graphical user interfaces (GUIs). The remaining sections examine various methods—CGI programming, specialized APIs, and server-side scripting—for integrating a Web server with back-end programs, including relational databases.

Internet

- The Internet is a computer network made up of thousands of networks worldwide.
- All computers on the Internet communicate with one another using the Transmission Control Protocol/Internet Protocol suite, abbreviated to TCP/IP.
- Computers on the Internet use a client/server architecture.
- This means that the remote server machine provides files and services to the user's local client machine.
- Software can be installed on a client computer to take advantage of the latest access technology.

An Internet user has access to a wide variety of services : electronic mail, file transfer, vast information resources, interest group membership, interactive collaboration, multimedia displays, real-time broadcasting



Web server

- A Web server is a computer that runs special serving software. That software "serves" HTML pages and the files associated with those pages when requested by a client, usually a Web browser.

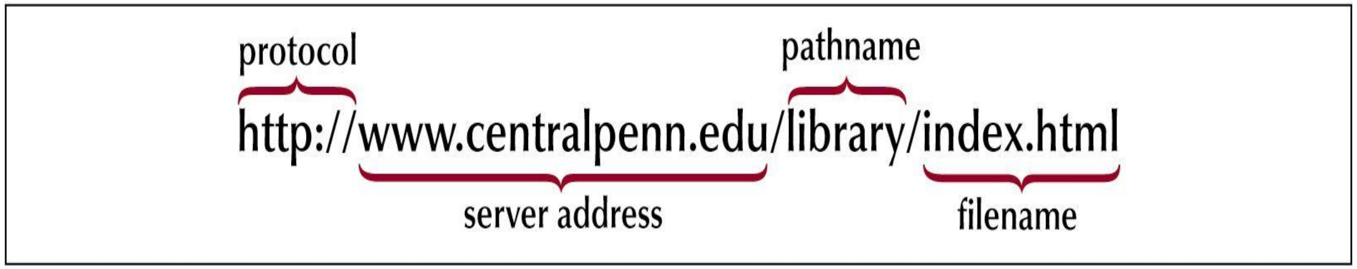
Client (“front end”) :

- Presents an interface to the user gathers information from the user, submits it to a server, then receives, formats, and presents the results returned from the server

Uniform Resource Locator **URL**

- A URL is a Web Page's address and identifies where the web page is stored on the Internet.
- It is a four-part addressing scheme.

Structure of a URL Figure 6



Internet Service Provider (ISP):

- Provides access to the Internet along with other types of services such as e-mail.

HyperText Transfer Protocol (HTTP)

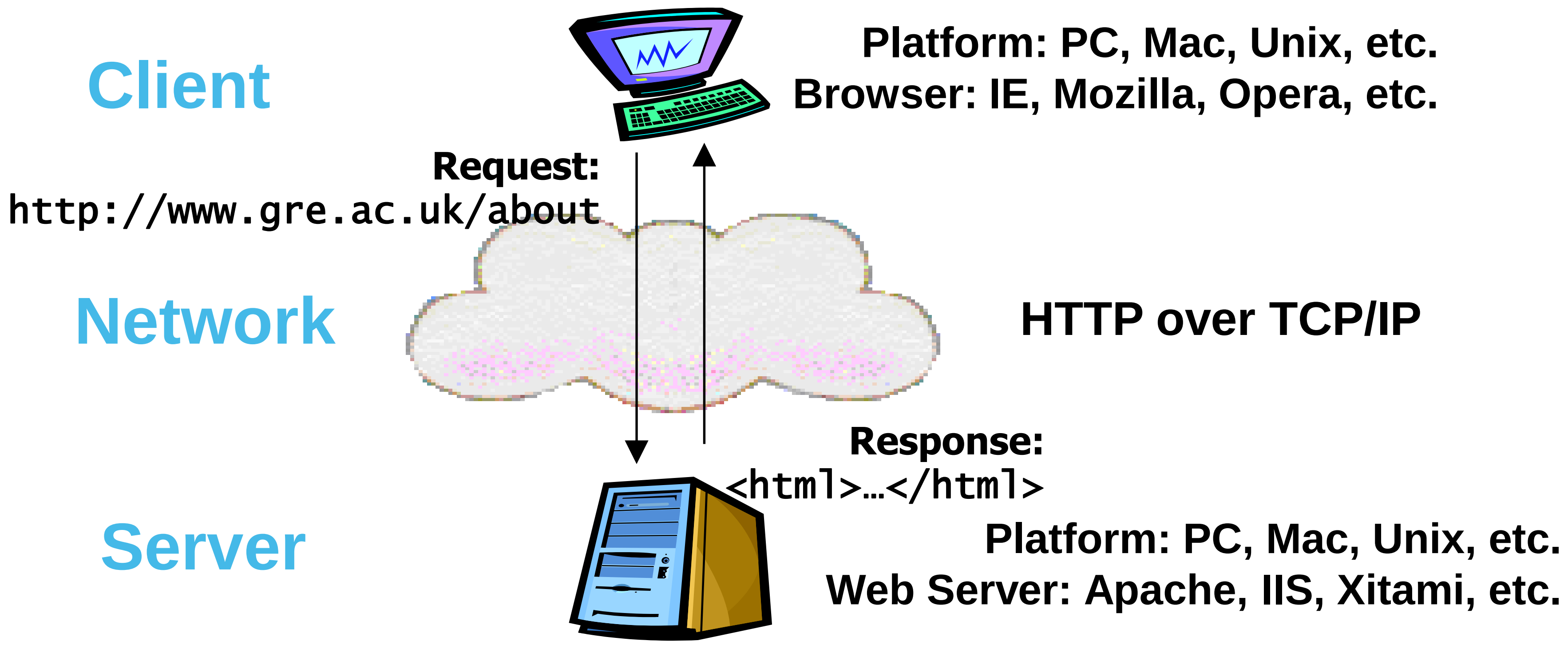
- to transmit data Protocols for other Internet applications

- Client-side:
  - HTML / XHTML (Extensible HyperText Markup Language)
  - JavaScript / VBScript (client-side scripting)
  - Applets / ActiveX controls
- Server-side:
  - JSP (Java Server Pages)
  - ASP (Active Server Pages)
  - ASP.NET (next generation of ASP)
  - PHP
  - Phython

2- Web application

- An application which is accessed via web browser over a network is called web application or web based application. All the websites are examples of web applications.
- Web application is written in a server side scripting language like ASP (active server pages).
- When user types address of website for example www.programming-web.com, browser transmits request to the web server which hosts the required site.
- On web server, web server software like email receives this request and processes it

- The output generated by web server includes only those scripts which can be rendered by web browser.
- Above process repeats when user perfumes an action which requires communication with server such as submitting entry form or viewing another page.


Client
Platform: PC, Mac, Unix, etc.
Browser: IE, Mozilla, Opera, etc.
Request:
http://www.gre.ac.uk/about
Network
HTTP over TCP/IP
Response:
<html>…</html>
Server
Platform: PC, Mac, Unix, etc.
Web Server: Apache, IIS, Xitami, etc.

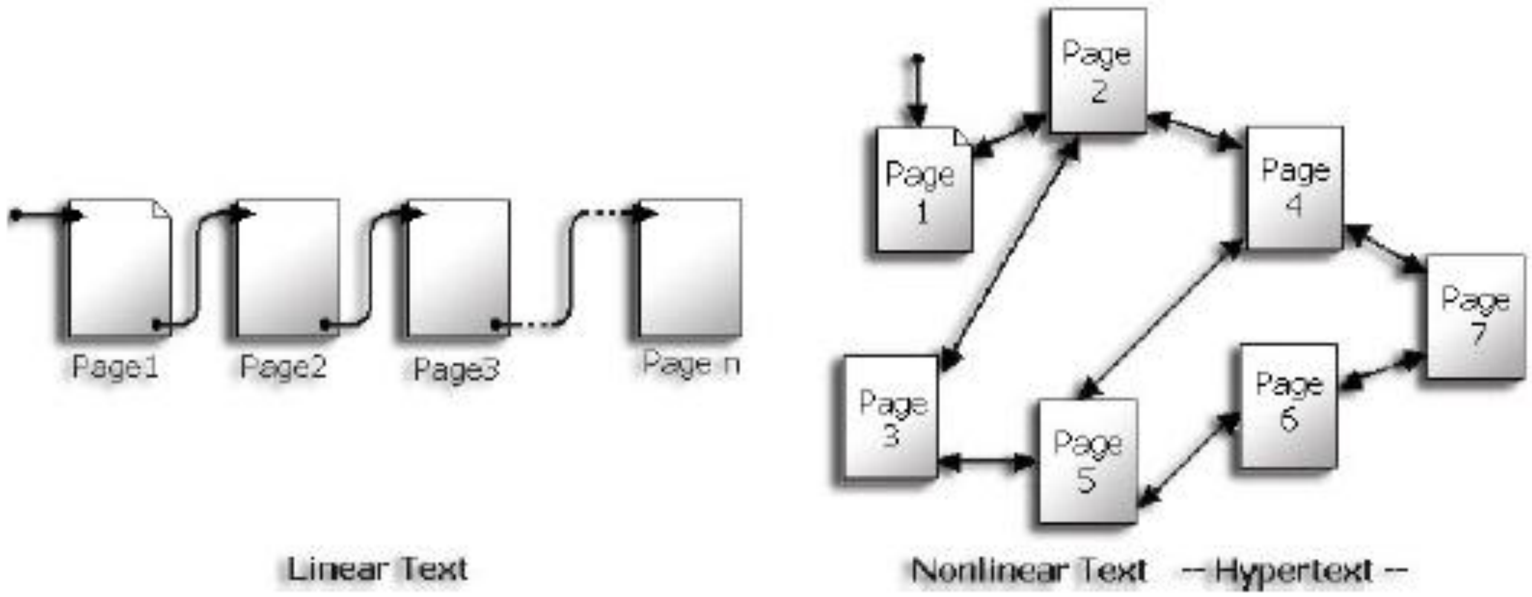
WWW Architecture

The Web Concepts

- The World-Wide Web (W3) was developed to be a pool of human knowledge, and human culture, which would allow collaborators in remote sites to share their ideas and all aspects of a common project.
- There are many Web concepts as following:

Hypertext:

- Hypertext links allow the reader to jump instantly from one electronic document to another.
- Two type : linear text and nonlinear text



three basic “rules” of hypertext:

- A large body of information is organized into numerous fragments, or in the case of the Web, into pages.
- The pages relate to each other.
- The user needs only a small fraction of the information at any given moment.

Web Page

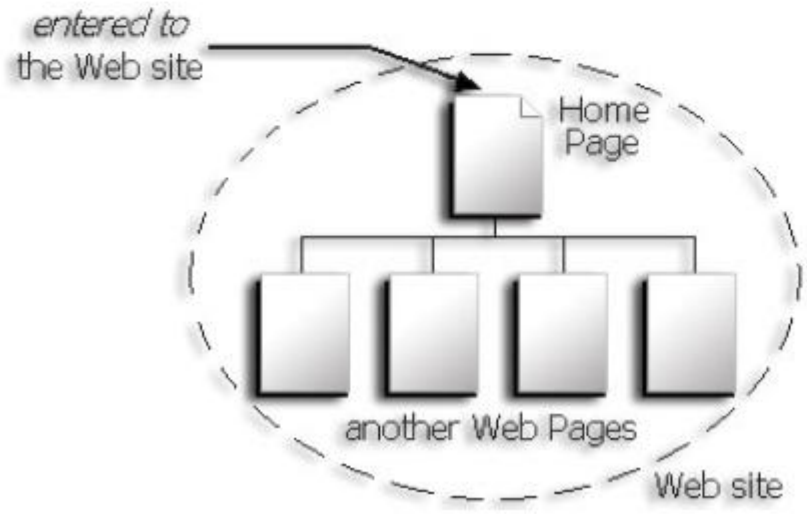
- The Web page is a space of information on the Internet, that presents information about a particular person, business, or organization or cause.
- The Web consists of files, called Web pages (documents).
- it is containing links to resources (text, images, audios, videos, and other data), throughout the Internet

Web Site

- A Web site is a group of related Web pages
- which presents information about a particular person, business, organization or cause
- A well-designed Web site is a collection of related Web documents (pages) that share a common theme, look, and feel.
- The first thing we can see when ‘enter’ the site is ‘Home Page’, that offers links to more detailed information on the different topics in the same subject covered by the site.

Web Browsing

- A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.
- An information resource is identified by a Uniform Resource Identifier (URI) and may be a web page ‘image, video, or other piece.



- CGI :(common gateway interface) refer to a specification by which program can communicate with a web server.

- http://www.icci.org/studies/ips.html .

- 1. Protocol: http.
- 2. Host computer name: www.
- 3. Second-level domain name: icci.
- 4. Top-level domain name: org.
- 5. Directory name: studies.
- 6. File name: ips.html.

Several Top-level domain are common:
com: commercial enterprise. شركات
edu: educational institution. للمؤسسات التعليمية
gov: government entity. للمؤسسات الحكومية
mil: military entity. للمواقع العسكرية
net: network access provider. للمواقع ذات النشاط الخاص
org: usually nonprofit organizations منظمة رسمية غير حكومية

Classifying the Web Sites:

- There are several classify for the early Web sites in many terms, as the follow:
- Environment:
- The General Approach
- Classify in terms of Range of Complexity:

Environment:

- There are three main types of Web sites according to this classify: Internet, Intranet, and Extranet Web Sites.

A- Internet Web Sites:

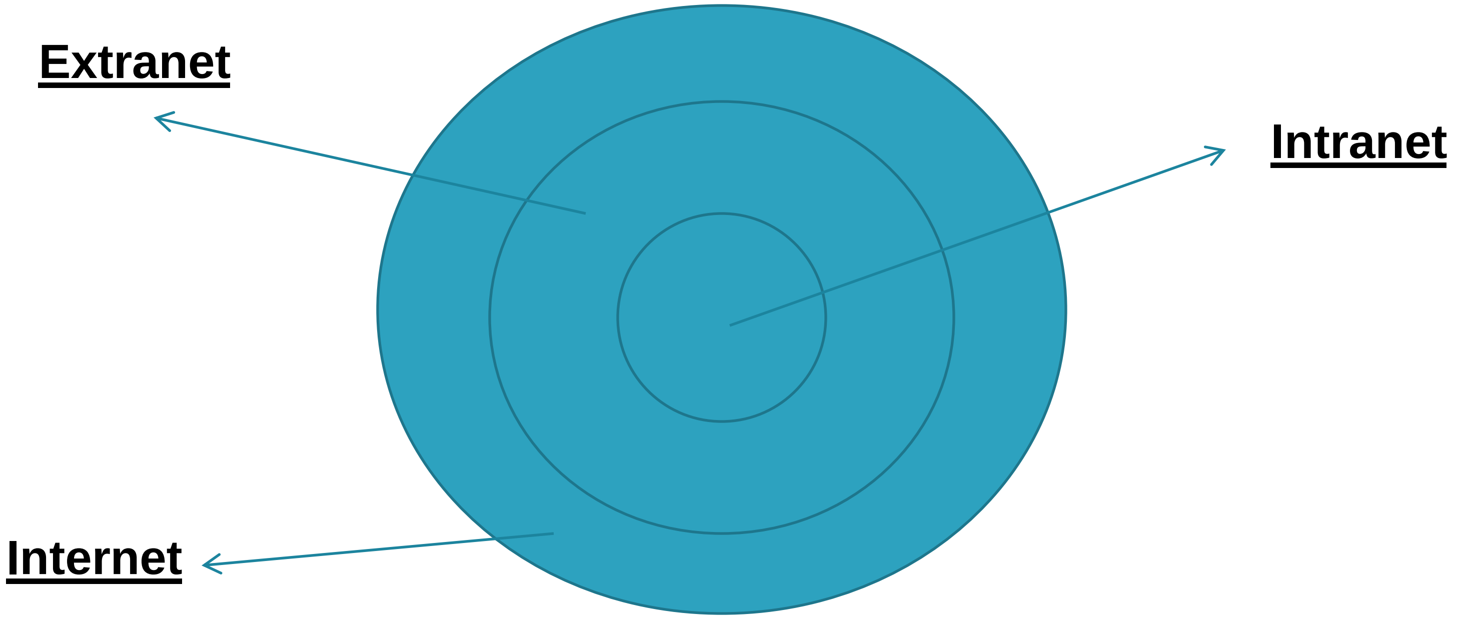
Internet Web Site is traditional Web sites that are intended for access by the general public.

B- Intranet Web Sites:

Intranet Web Site is intended only for internal (intra-organizational) use.

C- Extranet Web Sites:

Extranet Web Site is a combination of these. They are typically private and secured areas for the use of an organization and it is designated partners.



The General Approach:

- There are two main types of Web sites according to this classify: Static and Dynamic Web sites.

A- Static Web Sites:

B- Dynamic Web Sites:

A- Static Web Sites:

- Static Web Site implies that the Web site will be a flat-file system of HTML files.
- all pages reside on the server and have fixed content that will be served "as is" to the user.

B- Dynamic Web Sites:

- Dynamically site requires that the content be stored in a database, not all sites require complete database functionality
- part of a site may be dynamic while others are static.

Classify in terms of Range of Complexity

There are five main types of Web sites according to this classify:

- Static Web Sites.
- Static with Form-Based Interactivity Web Sites.
- Static with Dynamic Data Access Web Sites.
- Dynamically Generated Web Sites.
- Web-Based Software Applications Web Sites.

Introduction to HTML

Definitions

- W W W – World Wide Web.
- HTML – **HyperText Markup Language** – The Language of Web Pages on the World Wide Web.

  HTML is a text formatting language.
- Browser – A software program which is used to show web pages.

- “Normal text” surrounded by bracketed *tags* that tell browsers how to display web pages
- Pages end with “.htm” or “.html”
- HTML Editor – A word processor that has been specialized to make the writing of HTML documents more effortless.

Tags

- Codes enclosed in brackets
- Usually paired

  <TITLE>My Web Page</TITLE>
- ***Not*** case sensitive

  <TITLE> = <title> = <TITLE>

Choosing Text Editor

- There are many different programs that you can use to create web documents.
- HTML Editors enable users to create documents quickly and easily by pushing a few buttons. Instead of entering all of the HTML codes by hand.
- These programs will generate the HTML Source Code for you.

Choosing Text Editor

- HTML Editors are excellent tools for experienced web developers; however; it is important that you learn and understand the HTML language so that you can edit code and fix “bugs” in your pages.
- For this Course, we will focus on using the standard Microsoft Windows text editors, NotePad. We may use also textpad.

Starting NotePad

NotePad is the standard text editor that comes with the microsoft windows operating system. To start NotePad in follow the steps bellow:

- Click on the “Start” button located on your Windows task bar.
- Click on “Programs” and then click on the directory menu labeled “Accessories”.
- Locate the shortcut “NotePad” and click the shortcut once.

Creating a Basic Starting Document

```
<HTML>
<HEAD>
    <TITLE>Al al-Bayt University</TITLE>
</HEAD>
<BODY>
  This is what is displayed.
</BODY>
</HTML>
```

Previewing Your Work

- Once you have created your basic starting document and set your document properties it is a good idea to save your file.
- To save a file, in NotePad, follow these steps:

1. Locate and click on the menu called “File”.
2. Select the option under File Menu labeled “Save As”.
3. In the “File Name” text box, type in the entire name of your file (including the extension name .html).

Creating a Basic Starting Document

- The HEAD of your document point to above window part. The TITLE of your document appears in the very top line of the user’s browser. If the user chooses to “Bookmark” your page or save as a “Favorite”; it is the TITLE that is added to the list.
- The text in your TITLE should be as descriptive as possible because this is what many search engines, on the internet, use for indexing your site.

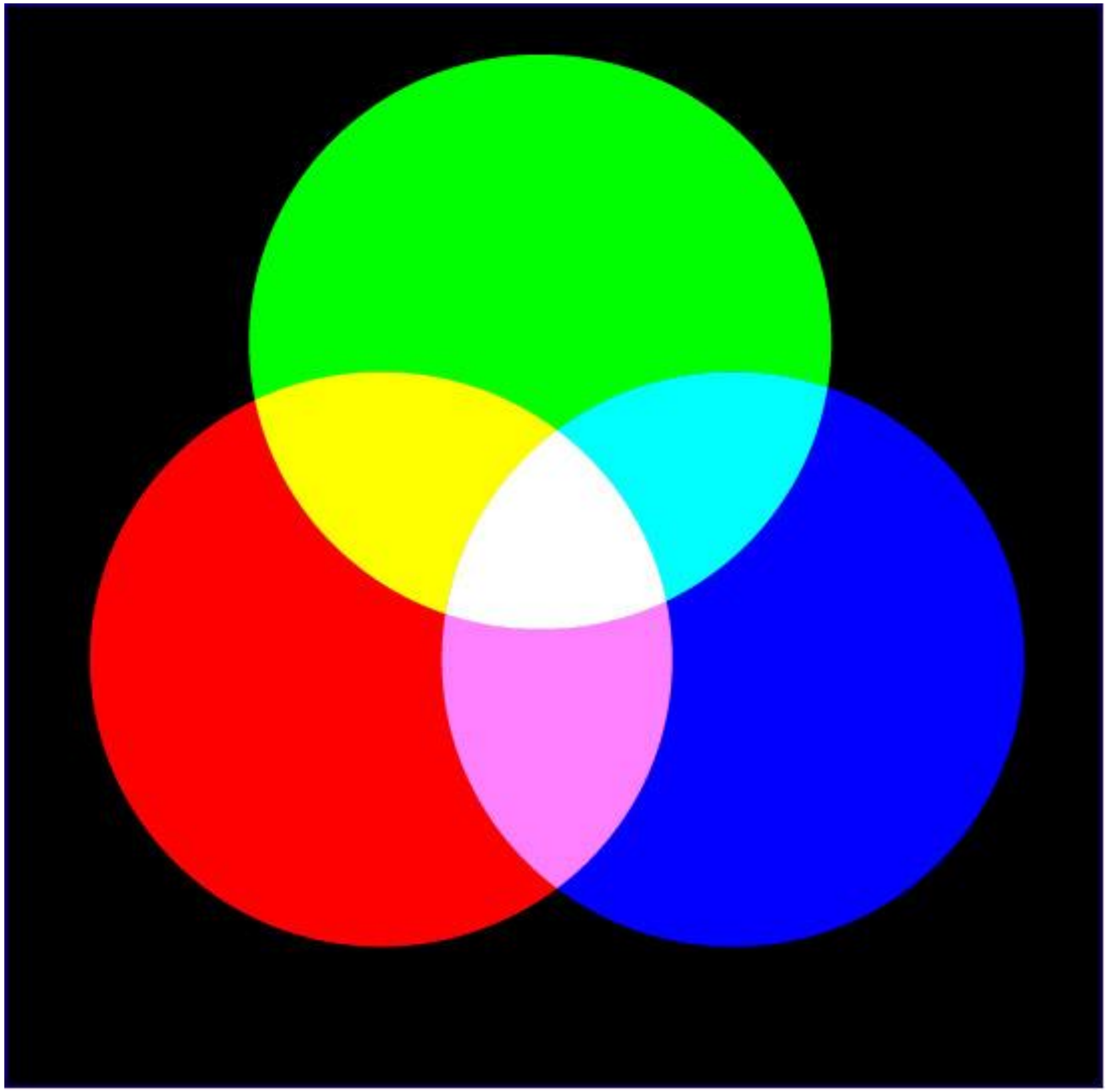
Setting Document Properties

- Document properties are controlled by attributes of the BODY element. For example, there are color settings for the background color of the page, the document's text and different states of links.

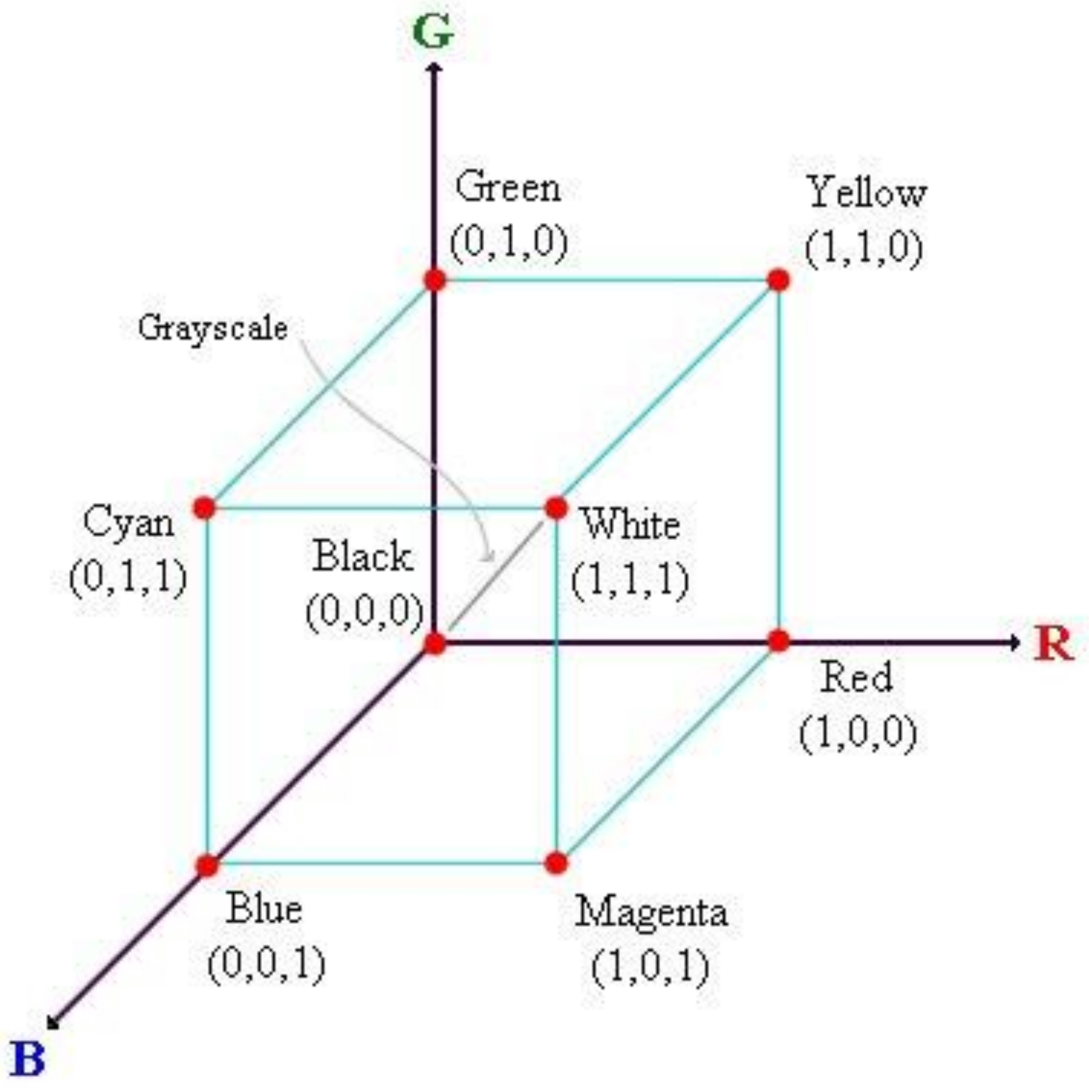
Color Codes

- Colors are set using “RGB” color codes, which are, represented as hexadecimal values. Each 2-digit section of the code represents the amount, in sequence, of red, green or blue that forms the color. For example, a RGB value with 00 as the first two digits has no red in the color.

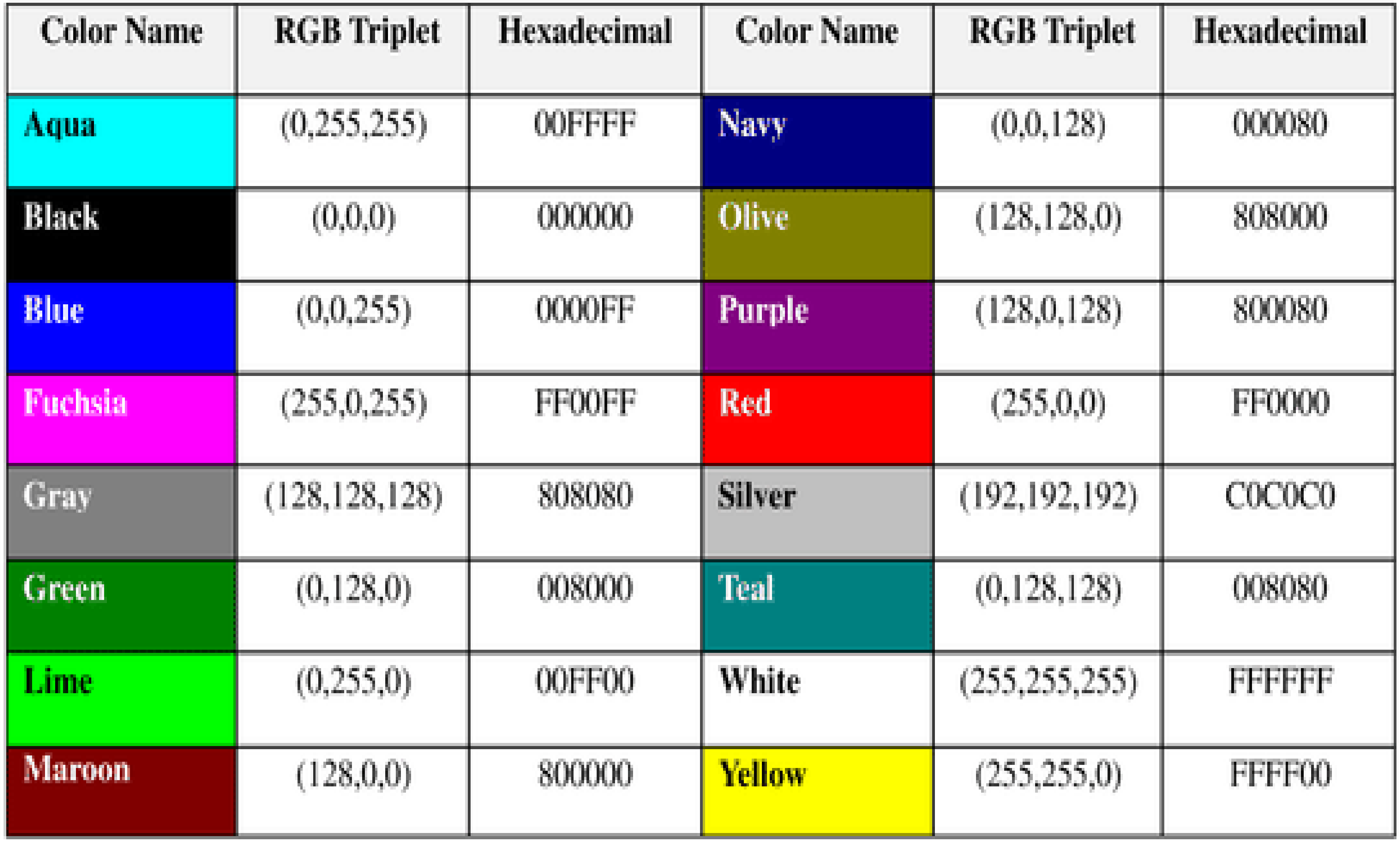
Main Colours



RGB Colour Model



16 Basic Colors



Color Name	RGB Triplet	Hexadecimal	Color Name	RGB Triplet	Hexadecimal
Aqua	(0,255,255)	00FFFF	Navy	(0,0,128)	000080
Black	(0,0,0)	000000	Olive	(128,128,0)	808000
Blue	(0,0,255)	0000FF	Purple	(128,0,128)	800080
Fuchsia	(255,0,255)	FF00FF	Red	(255,0,0)	FF0000
Gray	(128,128,128)	808080	Silver	(192,192,192)	C0C0C0
Green	(0,128,0)	008000	Teal	(0,128,128)	008080
Lime	(0,255,0)	00FF00	White	(255,255,255)	FFFFFF
Maroon	(128,0,0)	800000	Yellow	(255,255,0)	FFFF00

Color Codes

1. WHITE	1. #FFFFFF
2. BLACK	2. #000000
3. RED	3. #FF0000
4. GREEN	4. #00FF00
5. BLUE	5. #0000FF
6. MAGENTA	6. #FF00FF
7. CYAN	7. #00FFFF
8. YELLOW	8. #FFFF00
9. AQUAMARINE	9. #70DB93
10. BAKER'S CHOCOLATE	10. #5C3317
11. VIOLET	11. #9F5F9F
12. BRASS	12. #B5A642
13. COPPER	13. #B87333
14. PINK	14. #FF6EC7
15. ORANGE	15. #FF7F00

The Body Element

- The BODY element of a web page is an important element in regards to the page’s appearance. Here are the attributes of the **BODY** tag to control all the levels:

  TEXT="#RRGGBB" to change the color of **all the text** on the page (**full page text color.**)

- This element contains information about the page’s background color, the background image, as well as the text and link colors.

Background Color

- It is very common to see web pages with their background color set to white or some other colors.
- To set your document's background color, you need to edit the <BODY> element by adding the BGCOLOR attribute. The following example will display a document with a white background color:

<BODY BGCOLOR="#FFFFFF"></BODY>

TEXT Color

- The TEXT attribute is used to control the color of all the normal text in the document. The default color for text is black. The TEXT attribute would be added as follows:

  <BODY BGCOLOR=“#FFFFFF” TEXT=“#FF0000”></BODY>

In this example the document's page color is white and the text would be red.

LINK, VLINK, and ALINK

These attributes control the colors of the different link states:

1. LINK – initial appearance – default = Blue.
2. VLINK – visited link – default = Purple.
3. ALINK –active link being clicked–default= Yellow.

The Format for setting these attributes is:

```
<BODY BGCOLOR=“#FFFFFF” TEXT=“#FF0000”
     LINK=“#0000FF”
     VLINK=“#FF00FF”
    ALINK=“FFFF00”> </BODY>
```

Using Image Background

- The BODY element also gives you ability of setting an image as the document’s background.
- An example of a background image’s HTML code is as follows:

```
<BODY BACKGROUND=“hi.gif”
  BGCOLOR=“#FFFFFF”></BODY>
```

Headings, <Hx> </Hx>

- Inside the **BODY** element, heading elements **H1** through **H6** are generally used for major divisions of the document. Headings are permitted to appear in any order, but you will obtain the best results when your documents are displayed in a browser if you follow these guidelines:

1. **H1**: should be used as the highest level of heading, **H2** as the next highest, and so forth.
2. You should not skip heading levels: e.g., an **H3** should not appear after an **H1**, unless there is an **H2** between them.

Headings, <Hx> </Hx>

```
<HTML>
<HEAD>
<TITLE> Example Page</TITLE>
</HEAD>
<BODY>
<H1> Heading 1 </H1>
<H2> Heading 2 </H2>
<H3> Heading 3 </H3>
<H4> Heading 4 </H4>
<H5> Heading 5 </H5>
<H6> Heading 6 </H6>
</BODY>
</HTML>
```

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Paragraphs, <P> </P>

- Paragraphs allow you to add text to a document in such a way that it will automatically adjust the end of line to suite the window size of the browser in which it is being displayed. Each line of text will stretch the entire length of the window.

Paragraphs, <P> </P>

```
<HTML><HEAD>
<TITLE> Example Page</TITLE>
</HEAD>
<BODY></H1> Heading 1 </H1>
<P> Paragraph 1, ….</P>
<H2> Heading 2 </H2>
<P> Paragraph 2, ….</P>
<H3> Heading 3 </H3>
<P> Paragraph 3, ….</P>
<H4> Heading 4 </H4>
<P> Paragraph 4, ….</P>
<H5> Heading 5 </H5>
<P> Paragraph 5, ….</P>
<H6> Heading 6</H6>
<P> Paragraph 6, ….</P>
</BODY></HTML>
```

Heading 1

Paragraph 1,….

Heading 2

Paragraph 2,….

Heading 3

Paragraph 3,….

Heading 4

Paragraph 4,….

Heading 5

Paragraph 5,….

Heading 6

Paragraph 6,….

Break, <BR>

- Line breaks allow you to decide where the text will break on a line or continue to the end of the window.
- A <BR> is an empty Element, meaning that it may contain attributes but it does not contain content.
- The <BR> element does not have a closing tag.

Break, <BR>

```
<HTML>
<HEAD>
<TITLE> Example Page</TITLE>
</HEAD>
<BODY>
<H1> Heading 1 </H1>
<P>Paragraph 1, <BR>
Line 2 <BR> Line 3 <BR>….
</P>
</BODY>
</HTML>
```

Heading 1

Paragraph 1,….
Line 2
Line 3
….

Horizontal Rule, <HR>

- The <HR> element causes the browser to display a horizontal line (rule) in your document.
- <HR> does not use a closing tag, </HR>.

Horizontal Rule, <HR>

```
<HTML>
<HEAD>
<TITLE> Example Page</TITLE>
</HEAD>
<BODY>
<H1> Heading 1 </H1>
<P>Paragraph 1, <BR>
Line 2 <BR>
<HR>Line 3 <BR>
</P>
</BODY>
</HTML>
```

Heading 1

Paragraph 1,….

Line 2

Line 3

Bold, Italic and other Character Formatting Elements

- **<FONT SIZE=“+2”>** Two sizes bigger**</FONT>**
- The size attribute can be set as an absolute value from 1 to 7 or as a relative value using the “+” or “-” sign. Normal text size is 3 (from -2 to +4).
- **<B> Bold </B>**
- **<I> *Italic* </I>**
- **<U> <u>Underline</u> </U>**
- Color = “#RRGGBB” The COLOR attribute of the FONT element. E.g., **<FONT COLOR=“#RRGGBB”>this text has color</FONT>**

Bold, Italic and other Character Formatting Elements

```
<P> <FONT SIZE=“+1”> One
    Size Larger </FONT> - Normal
    –
<FONT SIZE=“-1”> One Size
    Smaller </FONT> <BR>
<B> Bold</B> - <I> italics</I> -
    <U> Underlined </U> -
<FONT COLOR=“#FF0000”>
    Colored </FONT> <BR>
```

One Size Larger - Normal – One Size Smaller
Bold - *italics* - <u>Underlined </u>- Colored

Alignment

- Some elements have attributes for alignment (ALIGN) e.g. Headings, Paragraphs and Horizontal Rules.
- The Three alignment values are : LEFT, RIGHT, CENTER.
- <CENTER></CENTER> Will center elements.

Special Characters & Symbols

Special Character	Entity Name	Special Character	Entity Name
Ampersand	&amp; &	Greater-than sign	&gt; >
Asterisk	&lowast; **	Less-than sign	&lt; <
Cent sign	&cent; ¢	Non-breaking space	&nbsp;
Copyright	&copy; ©	Quotation mark	&quot; "
Fraction one qtr	&frac14; ¼	Registration mark	&reg; ®
Fraction one half	&frac12; ½	Trademark sign	&trade; TM

Lists

In this chapter you will learn how to create a variety of lists.

Objectives

Upon completing this section, you should be able to

1. Create an unordered list.
2. Create an ordered list.
3. Create a defined list.
4. Nest Lists.

List Elements

- HTML supplies several list elements. Most list elements are composed of one or more <LI> (List Item) elements.
- UL : Unordered List. Items in this list start with a list mark such as a bullet. Browsers will usually change the list mark in nested lists.

<UL>

<LI> List item **…</LI>**

<LI> List item **…</LI>**

</UL>

- List item …
- List item …

List Elements

- You have the choice of three bullet types: **disc(default), circle, square.**
- These are controlled in Netscape Navigator by the “TYPE” attribute for the <UL> element.

<UL TYPE=“square”>

<LI> List item …</LI>

<LI> List item …</LI>

<LI> List item …</LI>

</UL>

- List item …
- List item …
- List item …

List Elements

- OL: Ordered List. Items in this list are numbered automatically by the browser.

```
<OL>
<LI> List item …</LI>
<LI> List item …</LI>
<LI> List item …</LI>
</OL>
```

1. **List item …**
2. **List item …**
3. **List item**

- You have the choice of setting the TYPE Attribute to one of five numbering styles.

List Elements

TYPE	Numbering Styles	
1	**Arabic numbers**	**1,2,3, ……**
a	**Lower alpha**	**a, b, c, ……**
A	**Upper alpha**	**A, B, C, ……**
i	**Lower roman**	**i, ii, iii, ……**
I	**Upper roman**	**I, II, III, ……**

List Elements

- You can specify a starting number for an ordered list.

<OL TYPE =“i”>

<LI> List item …</LI>

<LI> List item …</LI>

</OL>

<P> text ….</P>

<OL TYPE=“i” START=“3”>

<LI> List item …</LI>

</OL>

List Elements

i. List item …
ii. List item …

Text ….

iii. List item …

List Elements

- **DL: Definition List**. This kind of list is different from the others. Each item in a DL consists of one or more **Definition Terms (DT elements),** followed by one or more **Definition Description (DD elements).**

```
<DL>
<DT> HTML </DT>
<DD> Hyper Text Markup Language </DD>
<DT> DOG </DT>
<DD> A human’s best friend!</DD>
</DL>
```

HTML

Hyper Text Markup Language

DOG

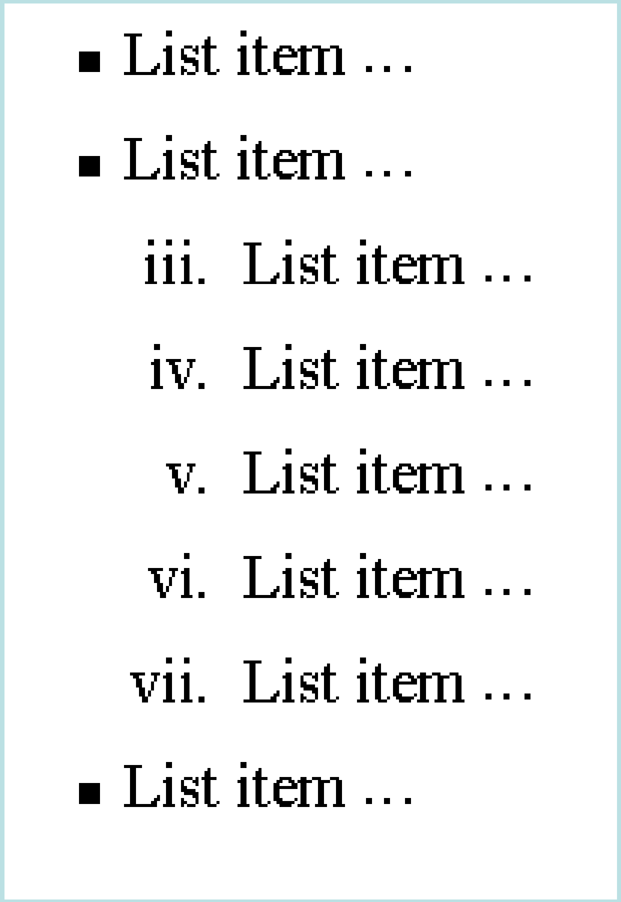
A human’s best friend!

Nesting Lists

- You can nest lists by inserting a UL, OL, etc., inside a list item (LI).

EXample

```
<UL TYPE = “square”>
<LI> List item …</LI>
<LI> List item …
<OL TYPE=“i” START=“3”>
<LI> List item …</LI>
<LI> List item …</LI>
<LI> List item …</LI>
<LI> List item …</LI>
<LI> List item …</LI>
</OL>
</LI>
<LI> List item …</LI>
</UL>
```



```
■ List item …
■ List item …
    iii. List item …
     iv. List item …
      v. List item …
     vi. List item …
    vii. List item …
■ List item …
```

What will be the output?

```
<H1 ALIGN="CENTER">SAFETY TIPS FOR CANOEISTS</H1>
<OL TYPE=“a” START=“2”>
<LI>Be able to swim </LI>
<LI>Wear a life jacket at all times </LI>
<LI>Don't stand up or move around. If canoe tips,
    <UL>
    <LI>Hang on to the canoe </LI>
    <LI>Use the canoe for support and </LI>
     <LI>Swim to shore
    </UL> </LI>
<LI>Don't overexert yourself </LI>
<LI>Use a bow light at night </LI>
</OL>
```

Images

In this chapter you will learn about images and how to place images in your pages.

Objectives

Upon completing this section, you should be able to

1. Add images to your pages.

Images

- **<IMG>**This element defines a graphic image on the page.
- **Image File (SRC:source):** This value will be a URL (location of the image) E.g. http://www.domain.com/dir/file.ext or /dir/file.txt.
- **Alternate Text (ALT):** This is a text field that describes an image or acts as a label. It is displayed when they position the cursor over a graphic image.
- **Alignment (ALIGN):** This allows you to align the image on your page.

Images

- **Width (WIDTH):** is the width of the image in pixels.
- **Height (HEIGHT):** is the height of the image in pixels.
- **Border (BORDER):** is for a border around the image, specified in pixels.

Some Examples on images

1) <IMG SRC=“jordan.gif“ border=4>

2) <IMG SRC=" jordan.gif" width="60" height="60">

3) <IMG SRC=“jordan.gif" ALT="This is a text that goes with the image">

4) <IMG SRC=" jordan.gif “ Hspace="30" Vspace="10" border=20>

5) < IMG SRC =" jordan.gif“ align="left">

blast blast blast blast blast

Anchors, URLs and Image Maps

In this chapter you will learn about Uniform Resource Locator, and how to add them as Anchor or Links inside your web pages.

Objectives

Upon completing this section, you should be able to

1. Insert links into documents.
2. Define Link Types.
3. Define URL.
4. List some commonly used URLs.
5. Plan an Image Map.

HOW TO MAKE A LINK

1) The tags used to produce links are the <A> and </A>. The <A> tells where the link should start and the </A> indicates where the link ends. Everything between these two will work as a link.

2) The example below shows how to make the word Here work as a link to yahoo.

Click <A HREF="http://www.yahoo.com">here</A> to go to yahoo.

More on LINKs

<body LINK="#C0C0C0" VLINK="#808080" ALINK="#FF0000">

- LINK - standard link - to a page the visitor hasn't been to yet. (standard color is blue - #0000FF).
  VLINK - visited link - to a page the visitor has been to before. (standard color is purple - #800080).
  ALINK - active link - the color of the link when the mouse is on it. (standard color is red - #FF0000).

If the programmer what to change the color

- Click <a href="http://www.yahoo.com"><font color="FF00CC">here</font></a> to go to yahoo.

E-Mail (Electronic Mail)

E.g. mailto:kmf@yahoo.com

- The type of service is identified as the mail client program. This type of link will launch the users mail client.
- The recipient of the message is kmf@yahoo.com

<A HREF="mailto:kmf@yahoo.com">Send me More Information </A>

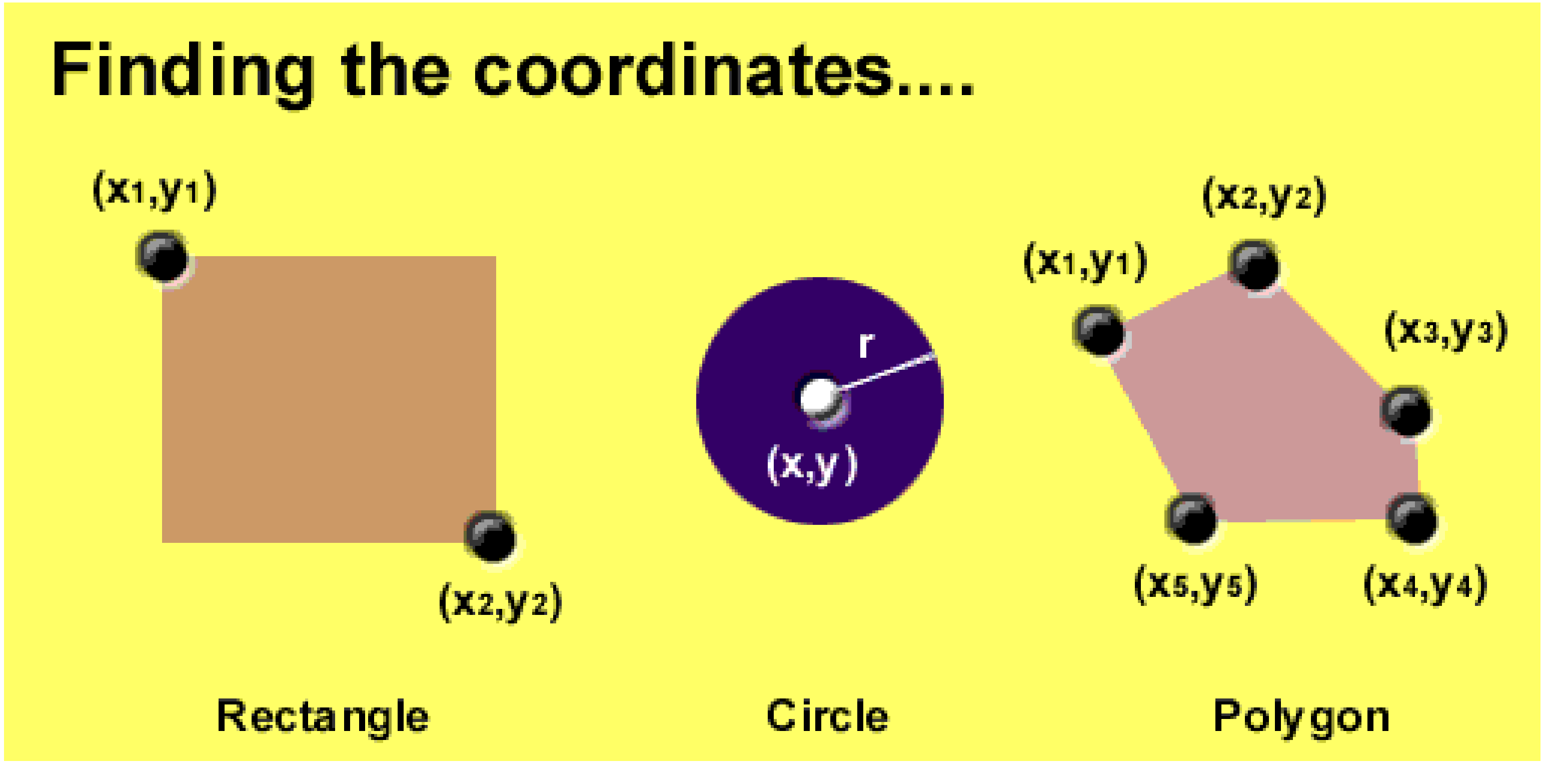
Image Maps

- Image maps are images, usually in gif format that have been divided into regions; clicking in a region of the image cause the web surfer to be connected to a new URL. Image maps are graphical form of creating links between pages.
- There are two type of image maps:

  Client side and server side

Both types of image maps involve a listing of co-ordinates that define the mapping regions and which URLs those coordinates are associated with. This is known as the map file.

Area Shapes Used



Client-Side Image Maps

- Client-side image maps (USEMAP) use a map file that is part of the HTML document (in an element called MAP), and is linked to the image by the Web browser.

```
<IMG SRC="note.GIF"  Width=200 Height=200
border="5" USEMAP="#map1">
<MAP NAME="map1">
<AREA SHAPE="RECT" COORDS="0,0,90,90"
HREF="hi.html"  ALT="see me…">
<AREA SHAPE="RECT" COORDS="100,100,160,160"
HREF="divPara.html"  ALT="see him…" >
<AREA SHAPE="CIRCLE" COORDS="150,50,20"
HREF="house.html"  ALT="see it…" >
</MAP>
```

We can use Poly as well as Rect……

Shapes, Coords

- Types of Shapes
  - Rect → used for squares and ordered shapes.
  - Circle → used for circles.
  - Poly → used for unordered shapes.
- Number of coordenations for each shape:
  - Rect →4 numbers for two corners
  - Circle →3 numbers for the center & R
  - Poly → depends on the number of corners of the shape(2 numbers for each corner)

Tables

In this chapter you will learn that tables have many uses in HTML.

Objectives:

Upon completing this section, you should be able to:

1. Insert a table.
2. Explain a table's attributes.
3. Edit a table.
4. Add a table header.

Tables

- The <TABLE></TABLE> element has four sub-elements:

1. Table Row<TR></TR>.
2. Table Header <TH></TH>.
3. Table Data <TD></TD>.
4. Caption <CAPTION></CAPTION>.

- The table row elements usually contain table header elements or table data elements.

Tables

```
<table border="1">
<tr>
<th> Column 1 header </th>
<th> Column 2 header </th>
</tr>
<tr>
<td> Row1, Col1 </td>
<td> Row1, Col2 </td>
</tr>
<tr>
<td> Row2, Col1 </td>
<td> Row2, Col2 </td>
</tr>
</table>
```

Tables

Column 1 Header	**Column 2 Header**
Row1, Col1	Row1, Col2
Row2, Col1	Row2, Col2

Tables Attributes

- **BGColor:** Some browsers support background colors in a table.
- **Width:** you can specify the table width as an absolute number of pixels or a percentage of the document width. You can set the width for the table cells as well.
- **Border:** You can choose a numerical value for the border width, which specifies the border in pixels.
- **CellSpacing:** Cell Spacing represents the space between cells and is specified in pixels.

Table Attributes

- **CellPadding:** Cell Padding is the space between the cell border and the cell contents and is specified in pixels.
- **Align:** tables can have left, right, or center alignment.
- **Background:** Background Image, will be titled in IE3.0 and above.
- BorderColor, BorderColorDark.

Table Caption

- A table caption allows you to specify a line of text that will appear centered above or bellow the table.

<TABLE BORDER=1 CELLPADDING=2>

<CAPTION ALIGN=“BOTTOM”> Label For My Table </CAPTION>

- The Caption element has one attribute ALIGN that can be either TOP (Above the table) or BOTTOM (below the table).

Table Header

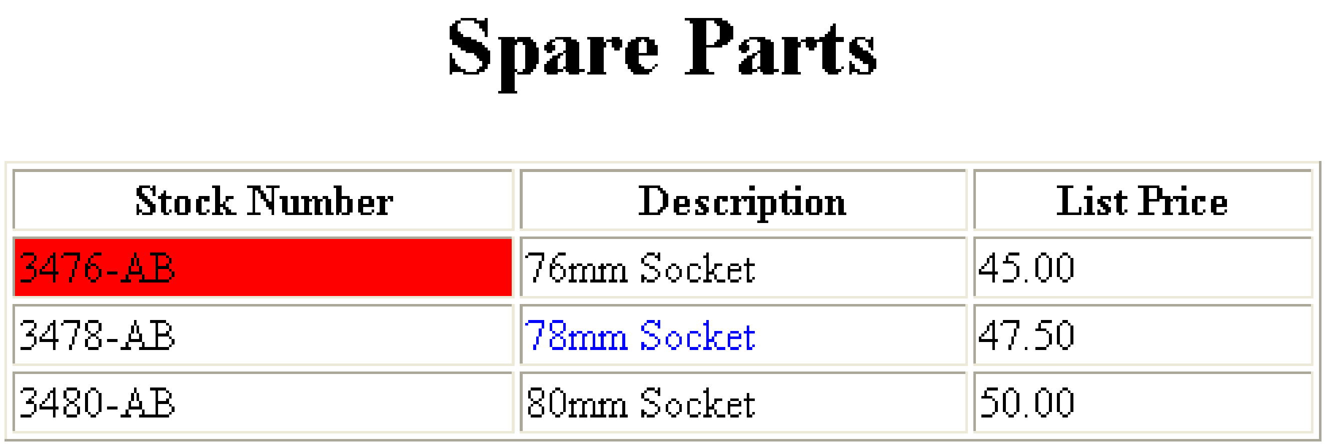
- Table Data cells are represented by the TD element. Cells can also be TH (Table Header) elements which results in the contents of the table header cells appearing centered and in bold text.

Table Data and Table Header Attributes

- **Colspan:** Specifies how many cell columns of the table this cell should span.
- **Rowspan***:* Specifies how many cell rows of the table this cell should span.
- **Align***:* cell data can have left, right, or center alignment.
- **Valign***:* cell data can have top, middle, or bottom alignment.
- **Width***:* you can specify the width as an absolute number of pixels or a percentage of the document width.
- **Height***:* You can specify the height as an absolute number of pixels or a percentage of the document height.

Basic Table Code

```
<TABLE BORDER=1 width=50%>
<CAPTION>  <h1>Spare Parts <h1> </Caption>
<TR><TH>Stock Number</TH><TH>Description</TH><TH>List
    Price</TH></TR>
<TR><TD bgcolor=red>3476-AB</TD><TD>76mm
    Socket</TD><TD>45.00</TD></TR>
<TR><TD >3478-AB</TD><TD><font color=blue>78mm Socket</font>
    </TD><TD>47.50</TD></TR>
<TR><TD>3480-AB</TD><TD>80mm Socket</TD><TD>50.00</TD></TR>
</TABLE>
```



Spare Parts

Stock Number	Description	List Price
3476-AB	76mm Socket	45.00
3478-AB	78mm Socket	47.50
3480-AB	80mm Socket	50.00

Table Data and Table Header Attributes

```
<Table border=1 cellpadding =2>
<tr> <th> Column 1 Header</th> <th>
  Column 2 Header</th> </tr>
<tr> <td colspan=2> Row 1 Col 1</td> </tr>
<tr> <td rowspan=2>Row 2 Col 1</td>
<td> Row 2 Col2</td> </tr>
<tr> <td> Row 3 Col2</td> </tr>
</table>
```

Table Data and Table Header Attributes

Column 1 Header	Column 2 Header
Row 1 Col 1	
Row 2 Col 1	Row 2 Col 2
	Row 3 Col 2

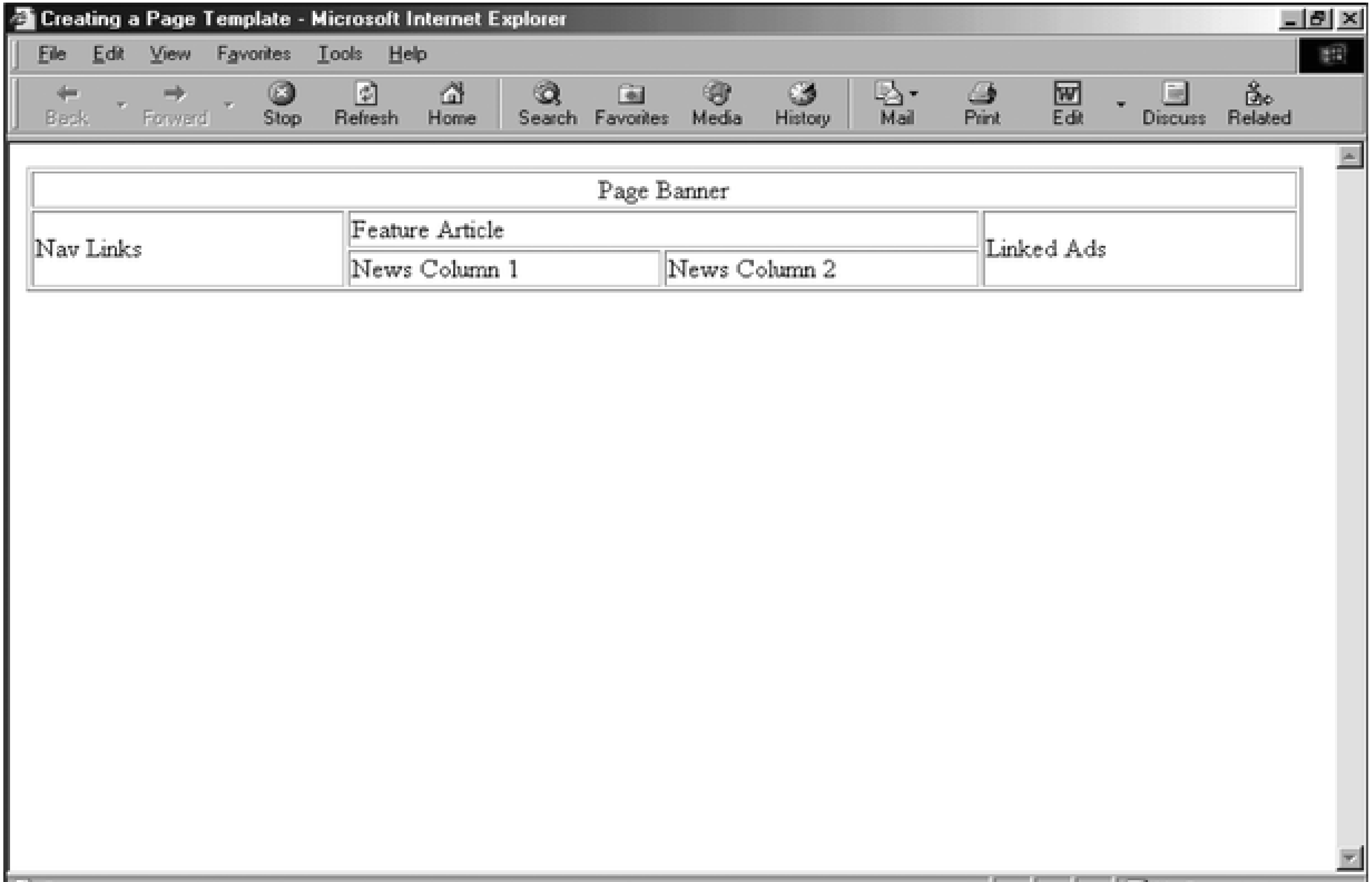
Special Things to Note

- **TH, TD and TR should always have end tags.**
Although the end tags are formally optional, many browsers will mess up the formatting of the table if you omit the end tags. In particular, you should ***always*** use end tags if you have a TABLE within a TABLE -- in this situation, the table parser gets hopelessly confused if you don't close your TH, TD and TR elements.
- **A default TABLE has no borders**
By default, tables are drawn without border lines. You need the BORDER attribute to draw the lines.
- **By default, a table is flush with the left margin**
TABLEs are plopped over on the left margin. If you want centered tables, You can either: place the table inside a DIV element with attribute ALIGN="center".
Most current browsers also supports table alignment, using the ALIGN attribute. Allowed values are "left", "right", or "center", for example: <TABLE ALIGN="left">. The values "left" and "right" float the table to the left or right of the page, with text flow allowed around the table. This is entirely equivalent to IMG alignment

What will be the output?

```
<TABLE BORDER width="750">
<TR> <TD colspan="4" align="center">Page
  Banner</TD></TR>

<TR> <TD rowspan="2" width="25%">Nav
  Links</TD><TD colspan="2">Feature
  Article</TD> <TD rowspan="2"
  width="25%">Linked Ads</TD></TR>

  <TR><TD width="25%">News Column 1 </TD>
  <TD width="25%"><News Column 2 </TD></TR>
</TABLE>
```

The Output


Creating a Page Template - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss Related

Page Banner			
Nav Links	Feature Article		Linked Ads
	News Column 1	News Column 2	

Frames

- Frames are a relatively new addition to the HTML standard. First introduced in Netscape Navigator 2.0.

Objectives:

Upon completing this section, you should be able to:

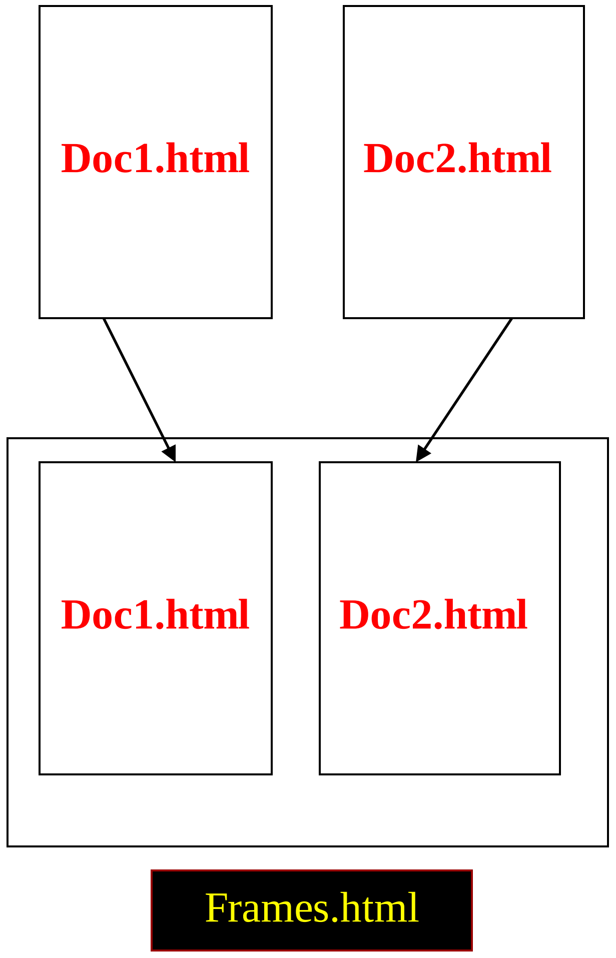
- Create a Frame based page.
- Work with the Frameset, Frame, and Noframes elements.
- Use the attributes of the Frames elements to control the display.
- Set Targets appropriately.

Frames

- A framed page is actually made up of multiple HTML pages. There is one HTML document that describes how to break up the single browser window into multiple windowpanes. Each windowpane is filled with an HTML document.

- For Example to make a framed page with a windowpane on the left and one on the right requires three HTML pages. ***Doc1.html*** and ***Doc2.html*** are the pages that contain content. ***Frames.html*** is the page that describes the division of the single browser window into two windowpanes.

Frames



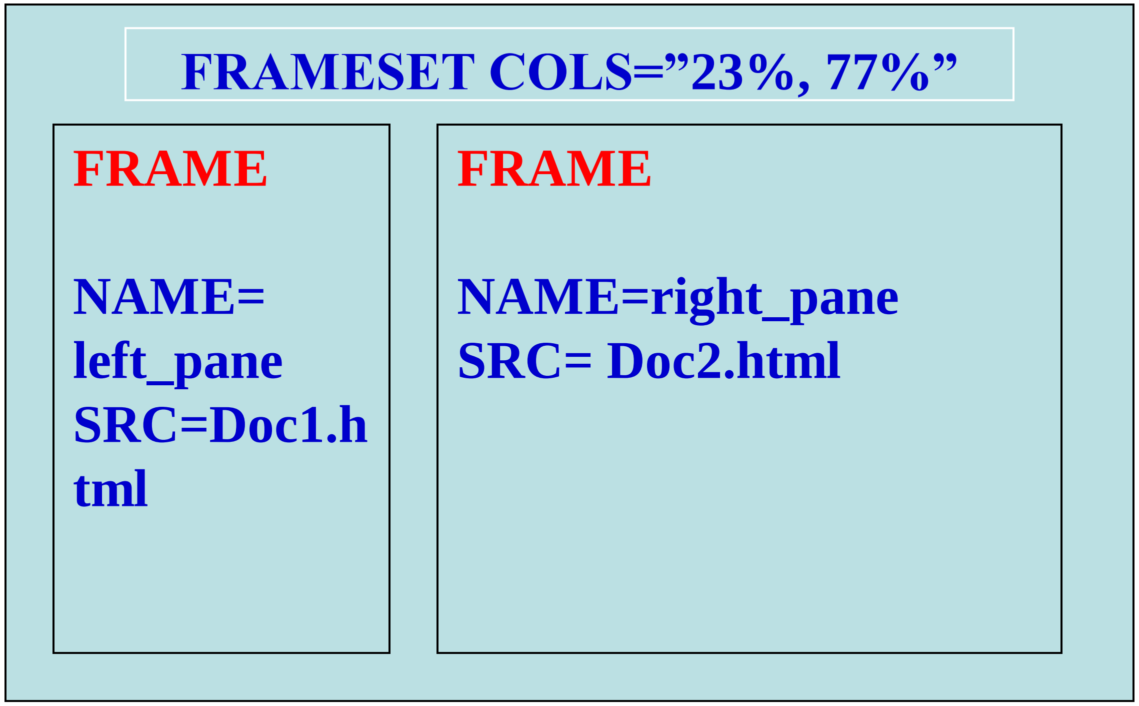
Frame Page Architecture

- A **<FRAMESET>** element is placed in the html document before the **<BODY>** element. The **<FRAMESET>** describes the amount of screen real estate given to each windowpane by dividing the screen into **ROWS** or **COLS**.
- The **<FRAMESET>** will then contain **<FRAME>** elements, **one per division** of the browser window.
- Note: Because there is no **BODY** container, FRAMESET pages can't have background images and background colors associated with them.

Frame Page Architecture

```
<HTML>
<HEAD>
<TITLE> Framed Page </TITLE>
<FRAMeSET COLS="23%,77%">
<FRAME SRC="Doc1.html">
<FRAME SRC="Doc2.html">
</FRAMeSET >
</HEAD>

</HTML>
```

The Diagram below is a graphical view of the document described above



<FRAMESET> **Container**

<FRAMESET> ***:*** The FRAMESET element creates divisions in the browser window in a single direction. This allows you to define divisions as either rows or columns.

- **ROWS** ***:*** Determines the size and number of rectangular rows within a <FRAMESET>. They are set from top of the display area to the bottom.

Possible values are:

- Absolute pixel units, I.e. “360,120”.
- A percentage of screen height, e.g. “75%,25%”.
- Proportional values using the asterisk (*). This is often combined with a value in pixels , e.g. “360,*”.
- <Frameset cols=“200,20%,*,2*”>

Creating a Frames Page

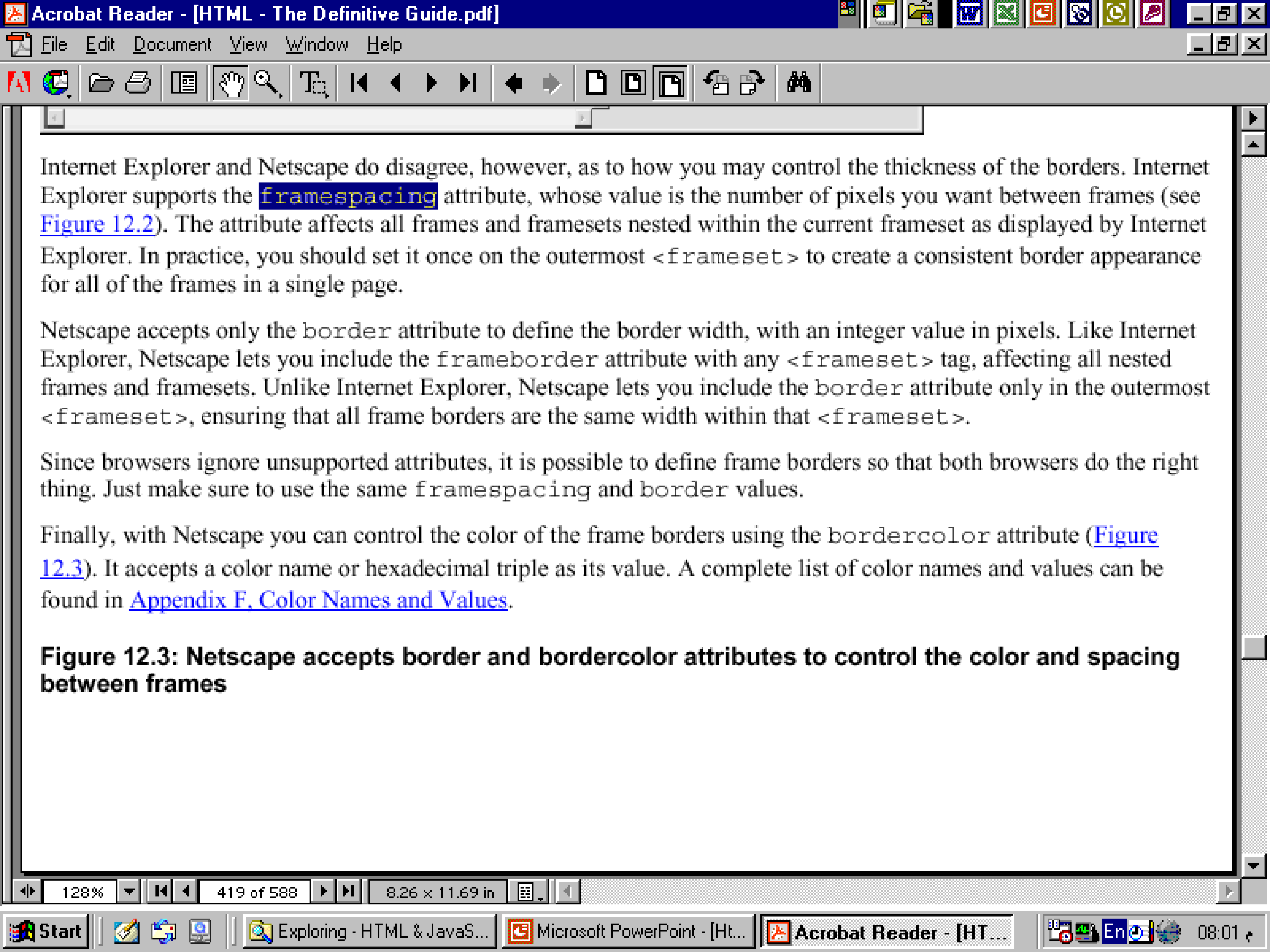
- **COLS*:*** Determines the size and number of rectangular columns within a <FRAMESET>. They are set from **left** to **right** of the display area.

Possible values are:

- Absolute pixel units, I.e. “480,160”.
- A percentage of screen width, e.g. “75%,25%”.
- Proportional values using the asterisk (*). This is often combined with a value in pixels , e.g. “480,*”.

Creating a Frames Page

- **FRAMEBORDER :** Possible values **0**, **1**, **YES**, **NO**. A setting of zero will create a borderless frame.
- **FRAMESPACING:** This attribute is specified in **pixels**. If you go to borderless frames you will need to set this value to zero as well, or you will have a gap between your frames where the border used to be.
- **BORDER(thickness of the Frame):** This attribute specified in pixels. A setting of zero will create a borderless frame. Default value is 5.
- **BORDERCOLOR:** This attribute is allows you choose a color for your border. This attribute is rarely used.


Acrobat Reader - [HTML - The Definitive Guide.pdf]
File Edit Document View Window Help
128%
419 of 588
8.26 x 11.69 in
Start
Exploring - HTML & JavaS...
Microsoft PowerPoint - [Ht...
Acrobat Reader - [HT...
En
08:01

Internet Explorer and Netscape do disagree, however, as to how you may control the thickness of the borders. Internet Explorer supports the `framespacing` attribute, whose value is the number of pixels you want between frames (see Figure 12.2). The attribute affects all frames and framesets nested within the current frameset as displayed by Internet Explorer. In practice, you should set it once on the outermost `<frameset>` to create a consistent border appearance for all of the frames in a single page.

Netscape accepts only the `border` attribute to define the border width, with an integer value in pixels. Like Internet Explorer, Netscape lets you include the `frameborder` attribute with any `<frameset>` tag, affecting all nested frames and framesets. Unlike Internet Explorer, Netscape lets you include the `border` attribute only in the outermost `<frameset>`, ensuring that all frame borders are the same width within that `<frameset>`.

Since browsers ignore unsupported attributes, it is possible to define frame borders so that both browsers do the right thing. Just make sure to use the same `framespacing` and `border` values.

Finally, with Netscape you can control the color of the frame borders using the `bordercolor` attribute (Figure 12.3). It accepts a color name or hexadecimal triple as its value. A complete list of color names and values can be found in Appendix F, Color Names and Values.

Figure 12.3: Netscape accepts border and bordercolor attributes to control the color and spacing between frames

<FRAME>

<FRAME>: This element defines a single frame within a frameset. There will be a FRAME element for each division created by the FRAMESET element. This tag has the following attributes:

- **SRC:** Required, as it provides the URL for the page that will be displayed in the frame.
- **NAME:** Required for frames that will allow targeting by other HTML documents. Works in conjunction with the target attribute of the <A>, <AREA>, <BASE>, and <FORM> tags.

<FRAME>

- **MARGINWIDTH:** Optional attribute stated in pixels. Determines horizontal space between the <FRAME> contents and the frame's borders.
- **MARGINHEIGHT:** Optional attribute stated in pixels. Determines vertical space between the <FRAME> contents and the frame's borders.
- **SCROLLING**: Displays a scroll bar(s) in the frame. Possible values are:

1. **Yes** – always display scroll bar(s).
2. **No** – never display scroll bar(s).
3. **Auto** – browser will decide based on frame contents.

By default: scrolling is auto.

<FRAME>

- **NORESIZE:** Optional – prevents viewers from resizing the frame. By default the user can stretch or shrink the frame’s display by selecting the frame’s border and moving it up, down, left, or right.

<NOFRAMES>

- **<NOFRAMES>:** Frame – capable browsers ignore all HTML within this tag including the contents of the BODY element. This element does not have any attributes.

```
<HTML>
<HEAD>
<TITLE> Framed Page </TITLE>
</HEAD>
```

<NOFRAMES>

```
<FRAMESET COLS="23%,77%">
<FRAME  SRC=""   NAME="left_pane“>
<FRAME  SRC=""   NAME="right_pane">
<NOFRAMES>
<P> This is a Framed Page. Upgrade your
   browser to support frames.</P>
</NOFRAMES></FRAMESET>
```

Compound FRAMESET Divisions

- In this case a second **FRAMESET** element will be inserted in the place of the **FRAME** element that would describe the second row.
- The second **FRAMESET** element will divide the remaining screen real estate into **2** columns.
- This nested **FRAMESET** will then be followed by **2 FRAME** elements to describe each of the subsequent frame divisions created.

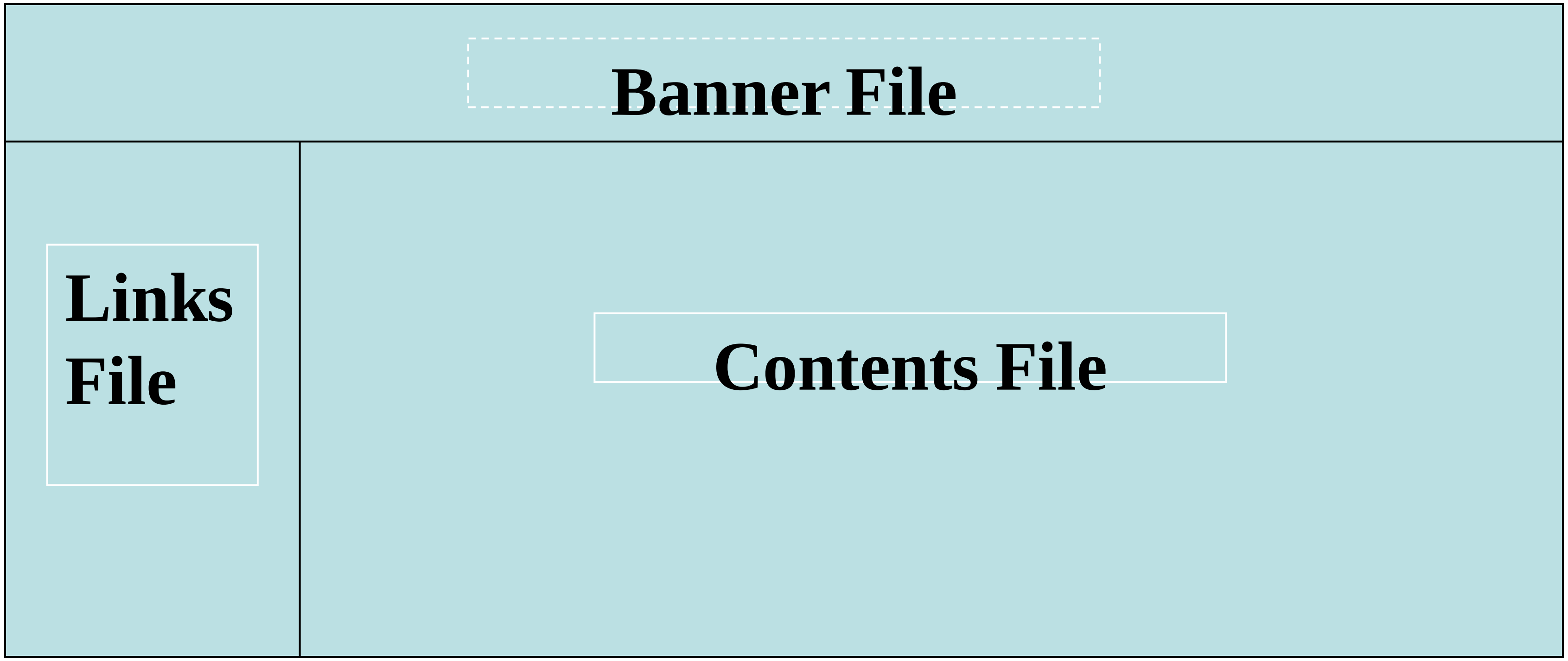
Compound FRAMESET Divisions

```
<html>
<head>
<title> Compound Frames Page</title>
</head>
<frameset rows="120,*">
<frame src="banner_file.html"
   name"banner">
<frameset cols="120,*">
<frame src="links_file.html"
   name="links">
<frame src="content_file.html"
   name="content">
```

```
<noframes>
<p>
Default
   message
</p>
</noframes>
</frameset>
</frameset>
</head>
```

Compound FRAMESET Divisions

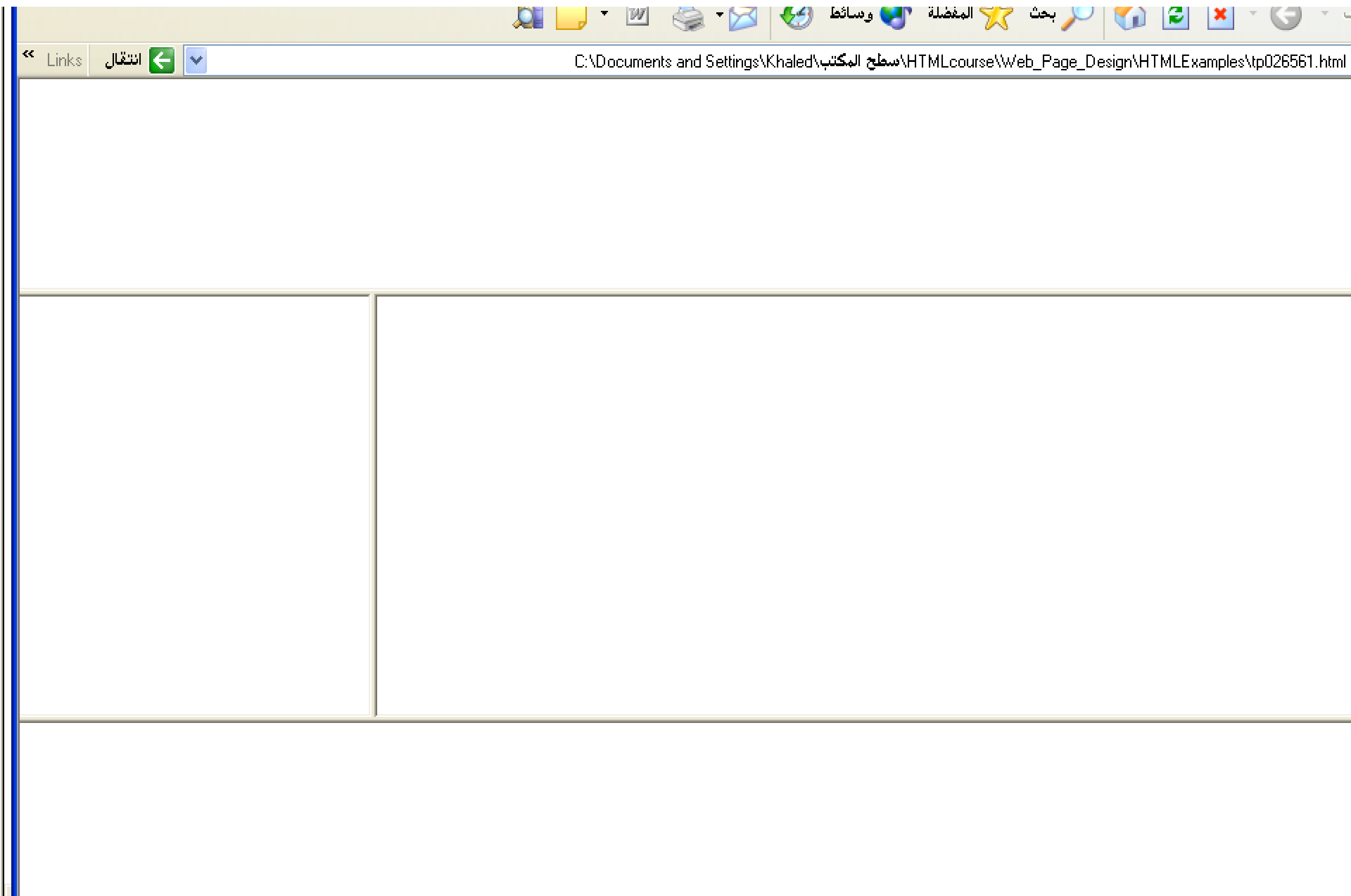
You may want to create a frames design with a combination of rows and columns.



Compound FRAMESET Divisions Example

```
<HEAD>
<FRAMESET ROWS="25%,50%,25%”
                    <FRAME SRC="">
<FRAMESET COLS="25%,*">
                              <FRAME SRC="">
                              <FRAME SRC="">
                                        </FRAMESET>
                    <FRAME SRC="">
</FRAMESET>
</HEAD>
```

Output


وسائط
المفضلة
بحث
Links
انتقال
C:\Documents and Settings\Khaled\سطح المكتب\HTMLcourse\Web_Page_Design\HTMLExamples\tp026561.html

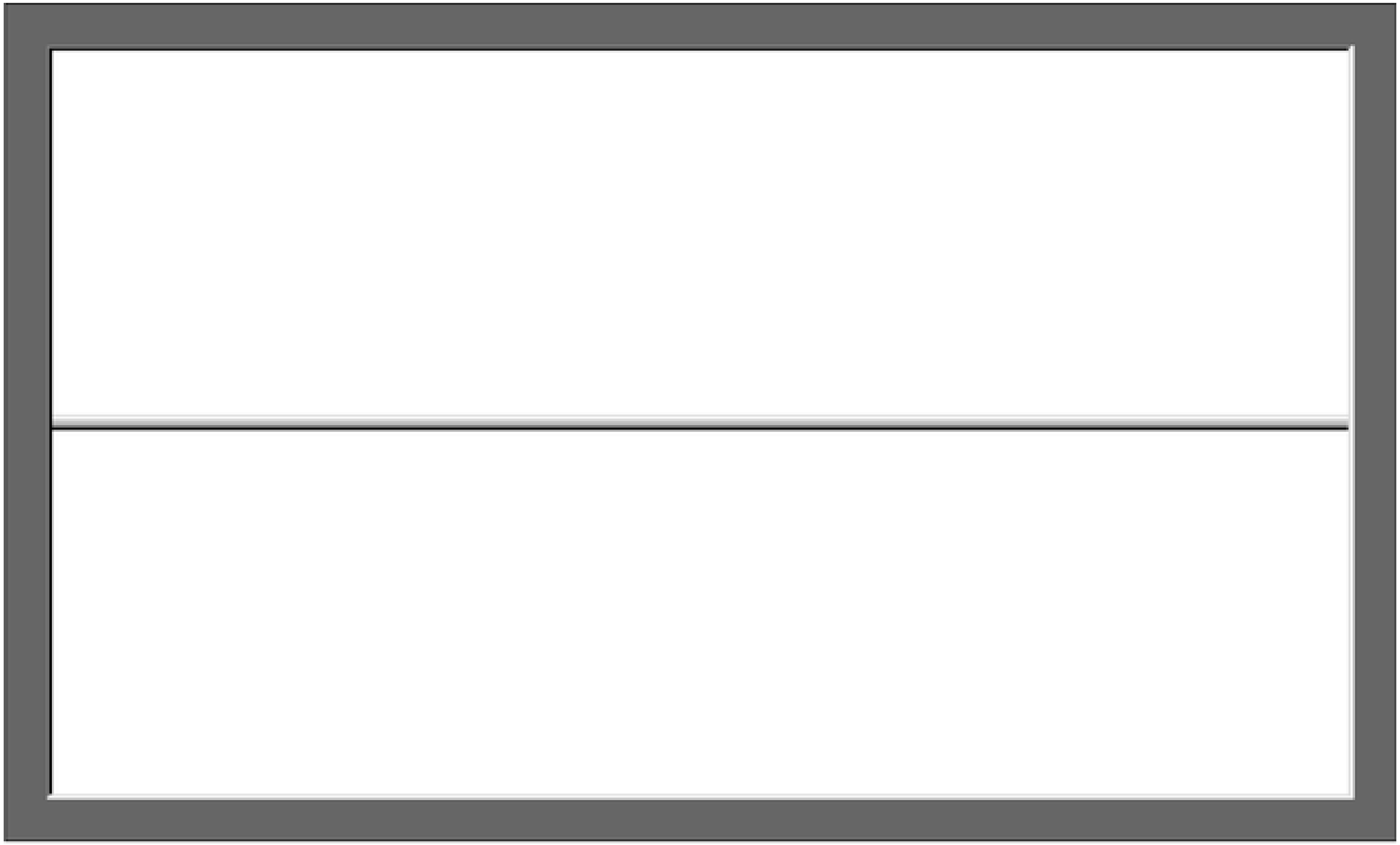


Figure 5-14: Frames created with <FRAMESET ROWS="50%, 50%">

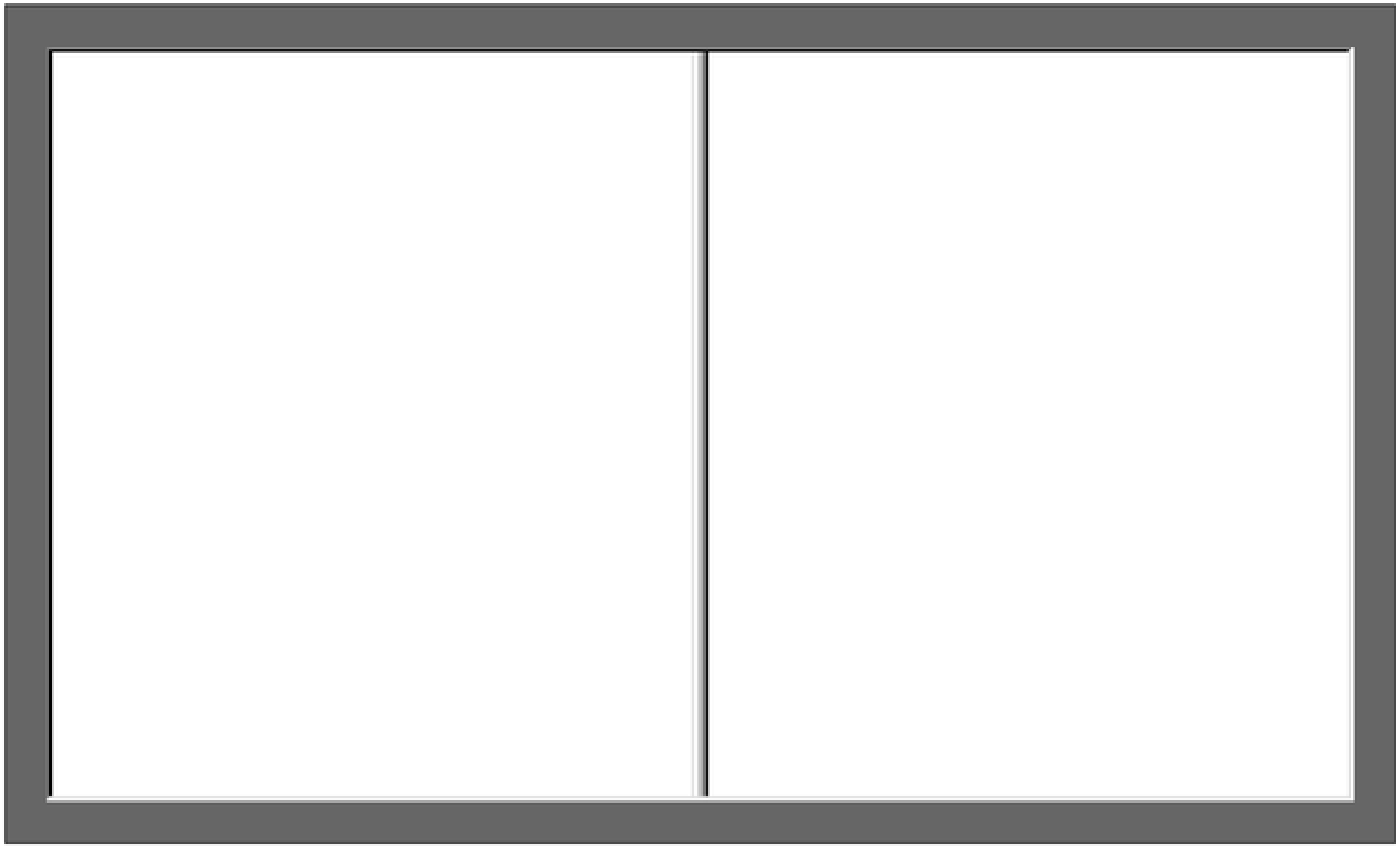


Figure 5-15: Frames created with <FRAMESET COLS="50%, 50%">

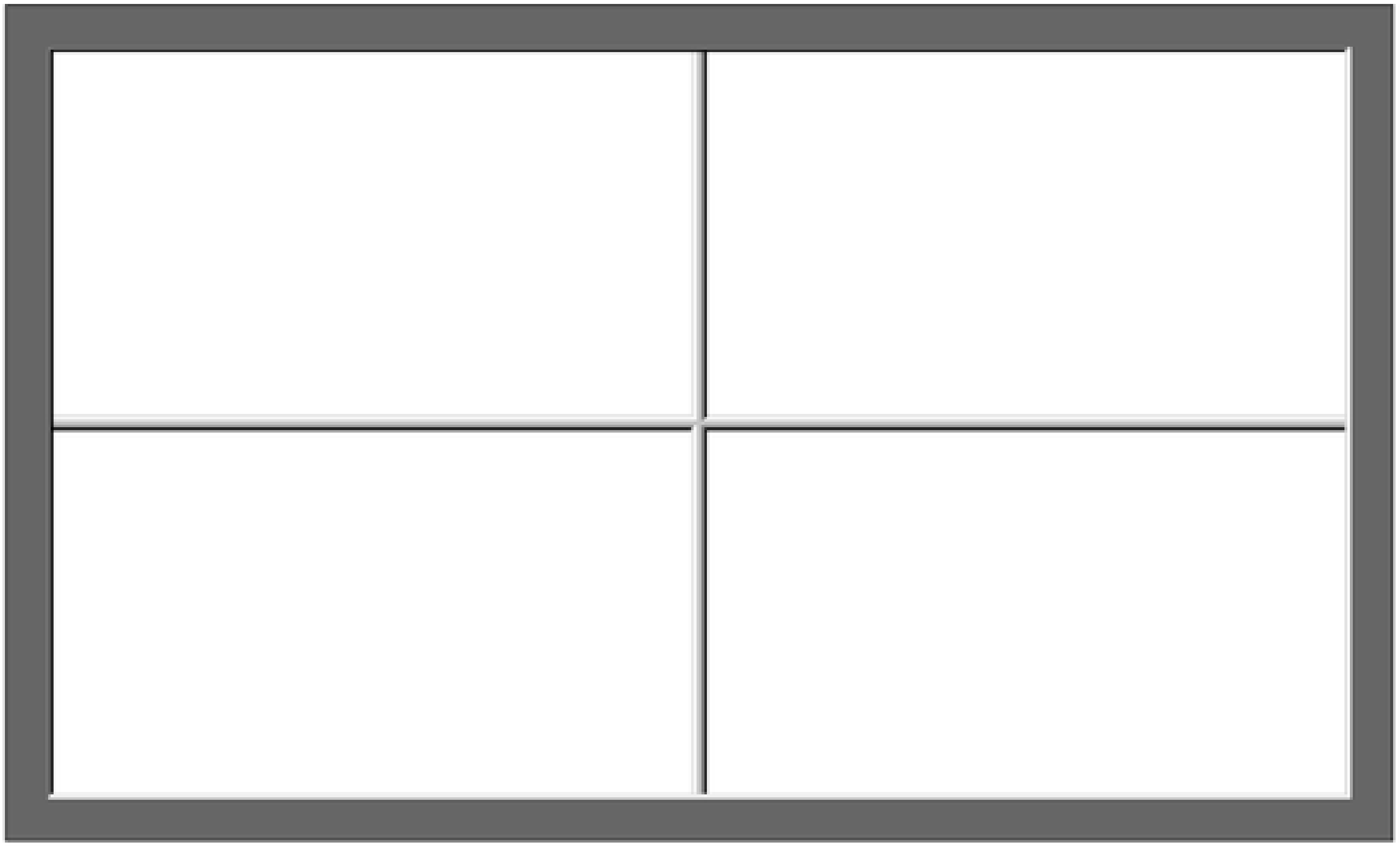


Figure 5-13: Frames created with <FRAMESET ROWS="50%, 50%" COLS="50%, 50%">

Frame Formatting

- Example:

```
<frameset rows="20%, *, 20%">
        <frame src="header.html" noresize
  scrolling=no>
        <frame src="body.html">
        <frame src="navigationbar.html"
  noresize   scrolling=no>
</frameset>
```

What do the following mean?

1) <FRAMESET COLS="2*, 3*, 5*">

2) <FRAMESET COLS="150, 20%, *, 3*">

So what are the space-allocation priorities?

Absolute pixel values are always assigned space first, in order from left to right. These are followed by percentage values of the total space. Finally, proportional values are divided based upon what space is left.

What will be the Output?

```
<FRAMESET ROWS="*, 2*, *"    COLS="2*, *">
<FRAME SRC=“”>
<FRAME SRC=“”>
<FRAME SRC=“”>
<FRAME SRC=“”>
<FRAME SRC=“”>
<FRAME SRC=“”>
</FRAMESET>
```

FORMS

- Forms add the ability to web pages to not only provide the person viewing the document with dynamic information but also to obtain information from the person viewing it, and process the information.

Objectives:

Upon completing this section, you should be able to

1. Create a FORM.
2. Add elements to a FORM.
3. Define CGI (Common Gateway Interface).
4. Describe the purpose of a CGI Application.
5. Specify an action for the FORM.

- Forms work in all browsers.
- Forms are Platform Independent.

FORMS

- To insert a form we use the <FORM></FORM> tags. The rest of the form elements must be inserted in between the form tags.

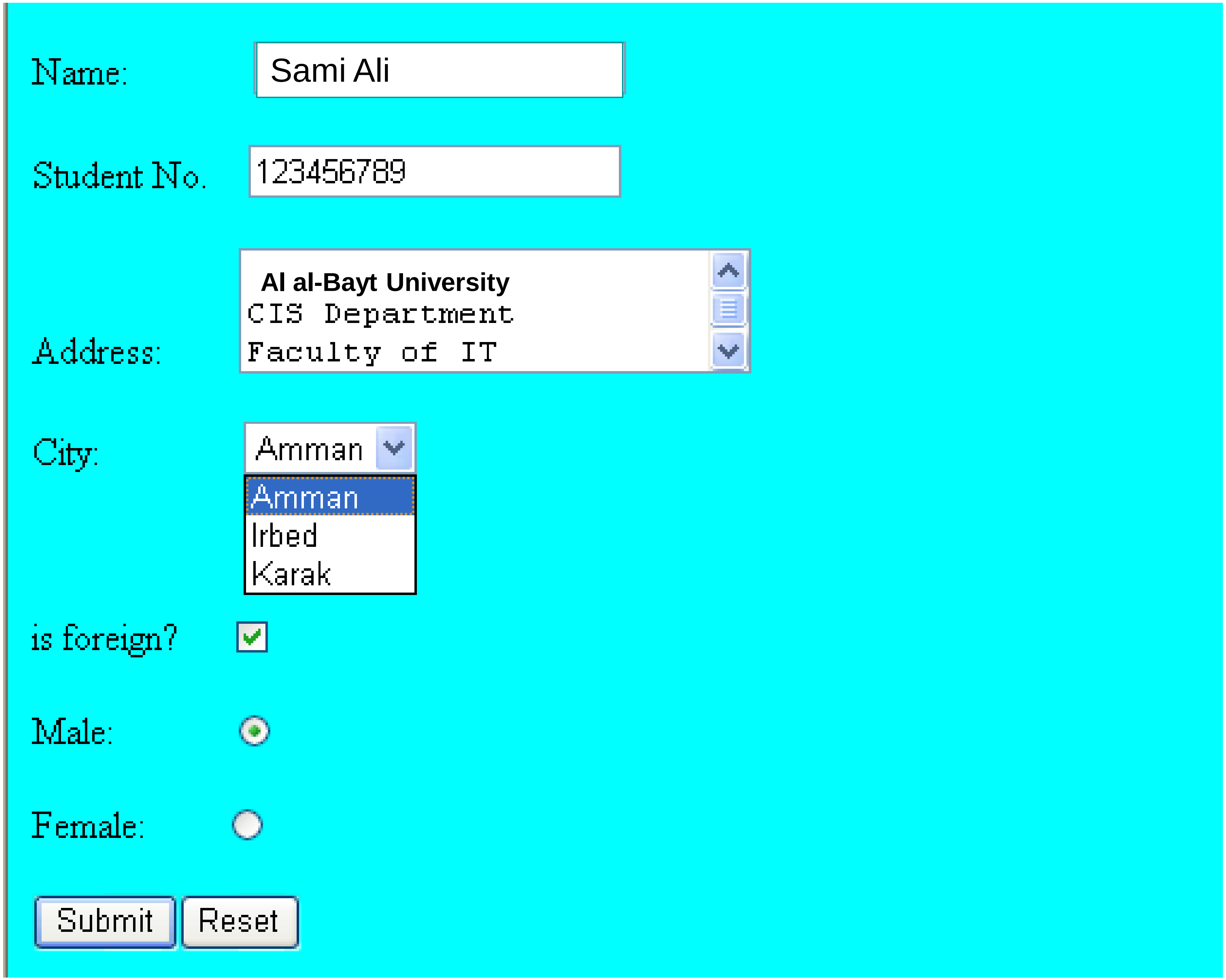
```
<HTML> <HEAD>
<TITLE> Sample Form</TITLE>
</HEAD>
<BODY BGCOLOR=“FFFFFF”>
<FORM ACTION = http://www.xnu.com/formtest.asp>
<P> First Name: <INPUT TYPE=“TEXT” NAME=“fname”
    MAXLENGTH=“50”> </P>
<P> <INPUT TYPE=“SUBMIT” NAME=“fsubmit1” VALUE=“Send Info”>
    </P>
</FORM>
</BODY> </HTML>
```

<FORM> element attributes

- **ACTION**: is the **URL** of the **CGI** (Common Gateway Interface) program that is going to accept the data from the form, process it, and send a response back to the browser.
- **METHOD**: **GET** (default) or **POST** specifies which **HTTP** method will be used to send the form's contents to the web server. The CGI application should be written to accept the data from either method.
- **NAME**: is a form name used by VBScript or JavaScripts.
- **TARGET**: is the target frame where the response page will show up.

Form Elements

- **Form elements have properties: Text boxes, Password boxes, Checkboxes, Option(Radio) buttons, Submit, Reset, File, Hidden and Image.**
- **The properties are specified in the TYPE Attribute of the HTML element <INPUT></INPUT>.**



Name: Sami Ali

Student No. 123456789

Address:
Al al-Bayt University
CIS Department
Faculty of IT

City: Amman
- Amman
- Irbed
- Karak

is foreign? [✓]

Male: ●

Female: ○

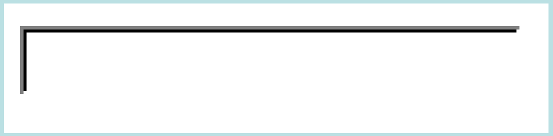
Form Elements

<INPUT> Element's Properties
TYPE= Type of INPUT entry field.
NAME = Variable name passed to CGI application
VALUE= The data associated with the variable name to be passed to the CGI application
CHECKED= Button/box checked
SIZE= Number of visible characters in text field
MAXLENGHT= Maximum number of characters accepted.

Text Box

- **Text boxes**: Used to provide input fields for text, phone numbers, dates, etc.

<INPUT TYPE= " TEXT " >



Browser will display

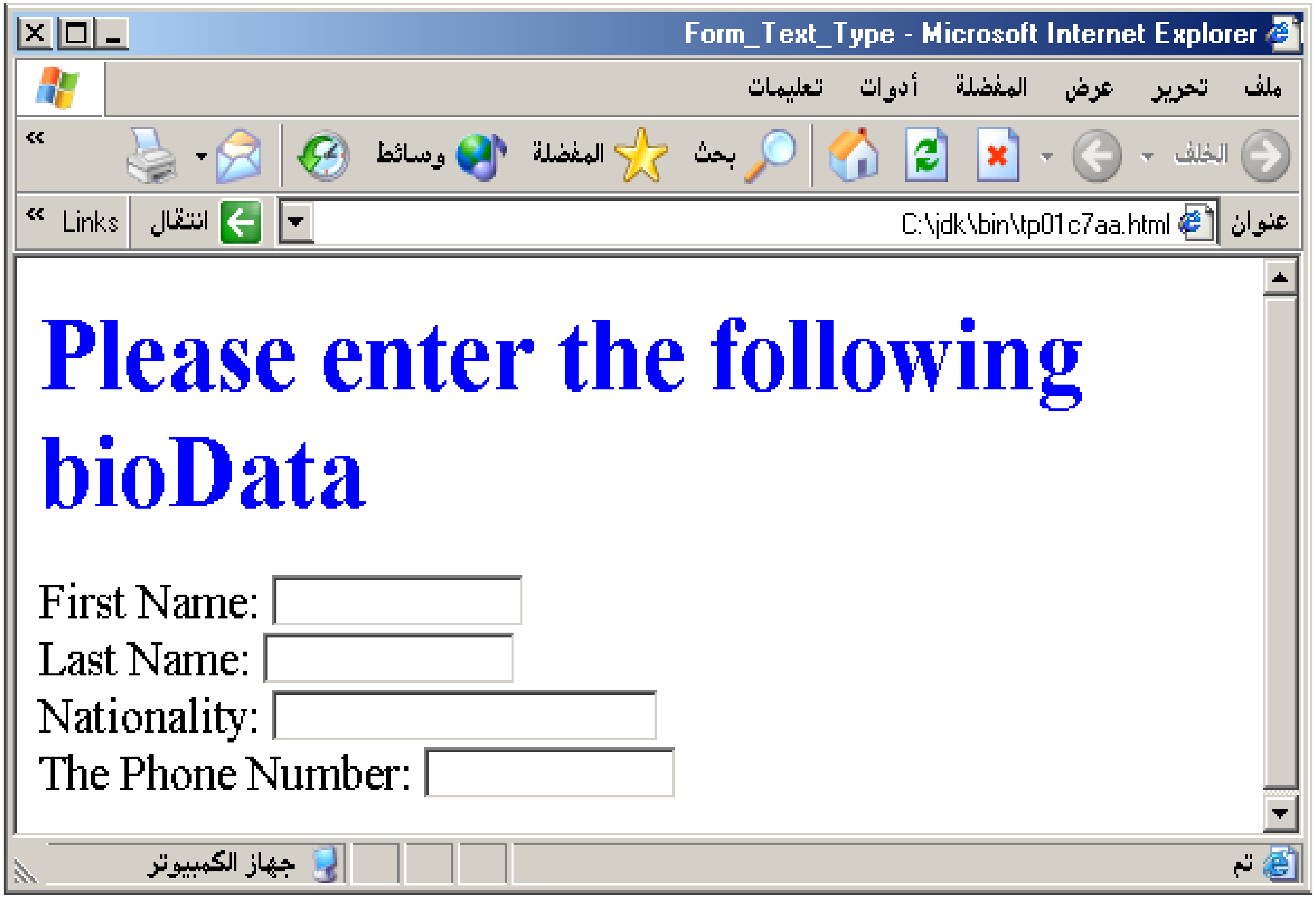
Textboxes use the following attributes:

- **TYPE:** text.
- **SIZE:** determines the size of the textbox in characters. **Default=20** characters.
- **MAXLENGHT :** determines the maximum number of characters that the field will accept.
- **NAME:** is the name of the variable to be sent to the CGI application.
- **VALUE:** will display its contents as the default value.

Example on Text Box

```
<TITLE>Form_Text_Type</TITLE>
</HEAD> <BODY>
<h1> <font color=blue>Please enter the following
   bioData</font></h1>
<FORM name="fome1"  Method= " get " Action= " URL " >
First Name: <INPUT TYPE="TEXT" NAME="FName"
SIZE="15" MAXLENGTH="25"><BR>
Last Name: <INPUT TYPE="TEXT" NAME="LName"
SIZE="15" MAXLENGTH="25"><BR>
Nationality: <INPUT TYPE="TEXT" NAME="Country"
SIZE="25" MAXLENGTH="25"><BR>
The Phone Number: <INPUT TYPE="TEXT" NAME="Phone"
SIZE="15" MAXLENGTH="12"><BR>
</FORM> </BODY> </HTML>
```

Output



Form_Text_Type - Microsoft Internet Explorer
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان C:\jdk\bin\tp01c7aa.html
انتقال Links
تم
جهاز الكمبيوتر

Please enter the following bioData

First Name:

Last Name:

Nationality:

The Phone Number:

Password

- **Password:** Used to allow entry of passwords.

<INPUT TYPE= " PASSWORD " >

Browser will display

Text typed in a password box is starred out in the browser display.

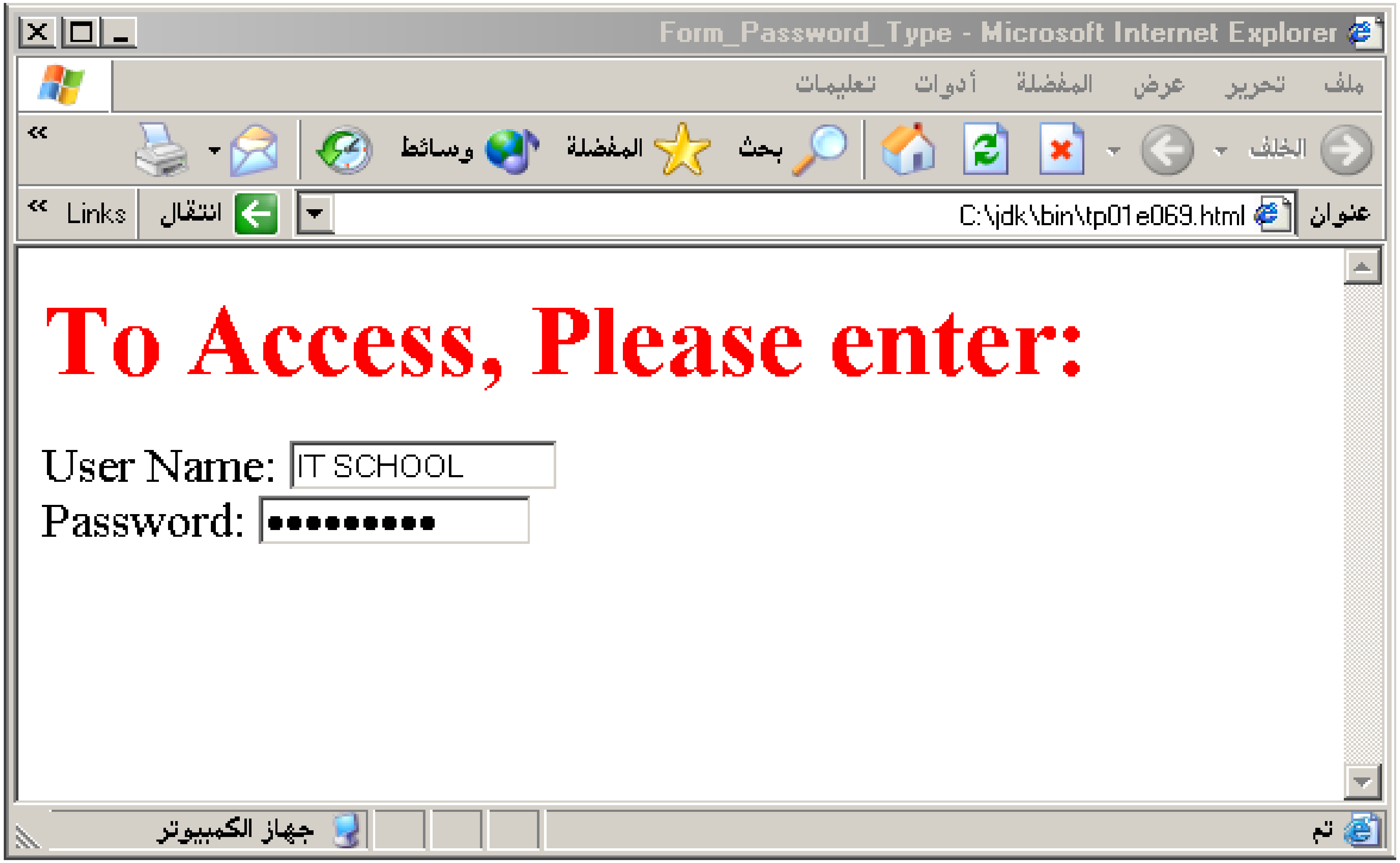
Password boxes use the following attributes:

- **TYPE:** password.
- **SIZE:** determines the size of the textbox in characters.
- **MAXLENGHT:** determines the maximum size of the password in characters.
- **NAME:** is the name of the variable to be sent to the CGI application.
- **VALUE:** is usually blank.

Example on Password Box

```
<HTML><HEAD>
<TITLE>Form_Password_Type</TITLE></HEAD>
<BODY>
<h1> <font color=red>To Access, Please
enter:</font></h1>
<FORM name="fome2"  Action="url"  method="get">
User Name: <INPUT TYPE="TEXT" Name="FName"
SIZE="15" MAXLENGTH="25"><BR>
Password: <INPUT TYPE="PASSWORD"
NAME="PWord"   value="" SIZE="15”
MAXLENGTH="25"><BR>
</FORM></BODY> </HTML>
```

Output


Form_Password_Type - Microsoft Internet Explorer
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف
بحث
المفضلة
وسائط
عنوان
C:\jdk\bin\tp01e069.html
انتقال
Links
تم
جهاز الكمبيوتر

To Access, Please enter:

User Name: IT SCHOOL

Password: ●●●●●●●●●●

Hidden

- **Hidden:** Used to send data to the CGI application that you don’t want the web surfer to see, change or have to enter but is necessary for the application to process the form correctly.

<INPUT TYPE=“HIDDEN”>

Nothing is displayed in the browser.

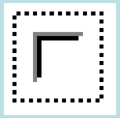
Hidden inputs have the following attributes:

- **TYPE:** hidden.
- **NAME:** is the name of the variable to be sent to the CGI application.
- **VALUE:** is usually set a value expected by the CGI application.

Check Box

- **Check Box:** Check boxes allow the users to select more than one option.

<INPUT TYPE="CHECKBOX">

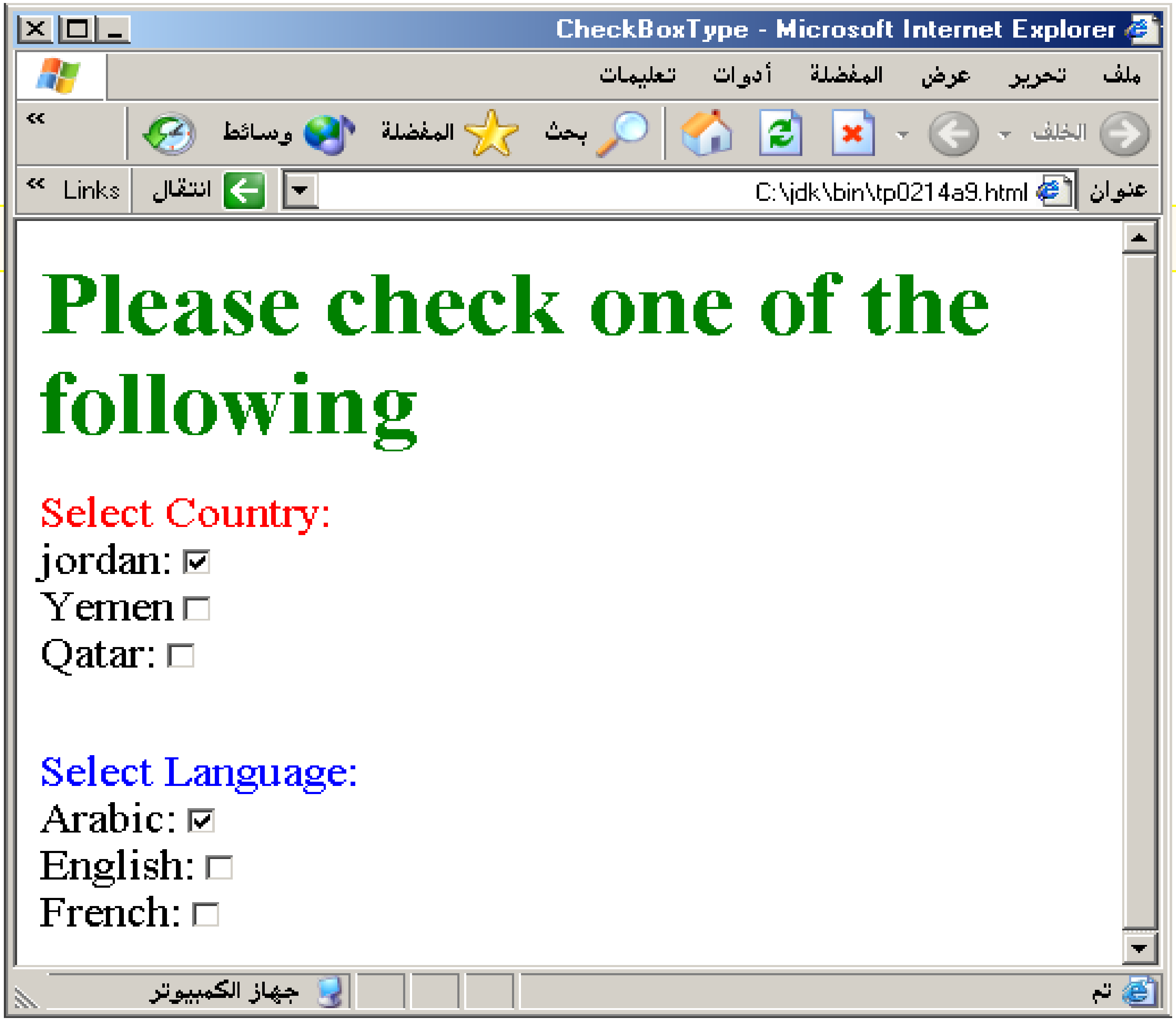
Browser will display 

Checkboxes have the following attributes:

- **TYPE:** checkbox.
- **CHECKED:** is blank or CHECKED as the initial status.
- **NAME*:*** is the name of the variable to be sent to the CGI application.
- **VALUE:** is usually set to a value.

```
<HTML> <HEAD><TITLE>CheckBoxType</TITLE> </HEAD>
<BODY>
<h1> <font color=green>Please check one of the
following</font></h1>
<FORM name="fome3"  Action="url"  method="get">
<font color=red> Select Country: </font><BR>
jordan:<INPUT TYPE="CheckBox" Name="country"
CHECKED><BR>
Yemen<INPUT TYPE="CheckBox"  Name="country"><BR>
Qatar:<INPUT TYPE="CheckBox" Name="country"><BR>
<BR>
<font color=blue>Select Language:</font><BR>
Arabic:<INPUT TYPE="CheckBox" Name="language"
CHECKED><BR> English:<INPUT TYPE="CheckBox"
Name="language"><BR>
French:<INPUT TYPE="CheckBox" Name="language">
<BR></FORM> </BODY></HTML>
```

Output


CheckBoxType - Microsoft Internet Explorer
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان C:\jdk\bin\tp0214a9.html انتقال Links
تم جهاز الكمبيوتر

Please check one of the following

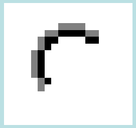
Select Country:
jordan: ☑
Yemen ☐
Qatar: ☐

Select Language:
Arabic: ☑
English: ☐
French: ☐

Radio Button

- **Radio Button:** Radio buttons allow the users to select only one option.

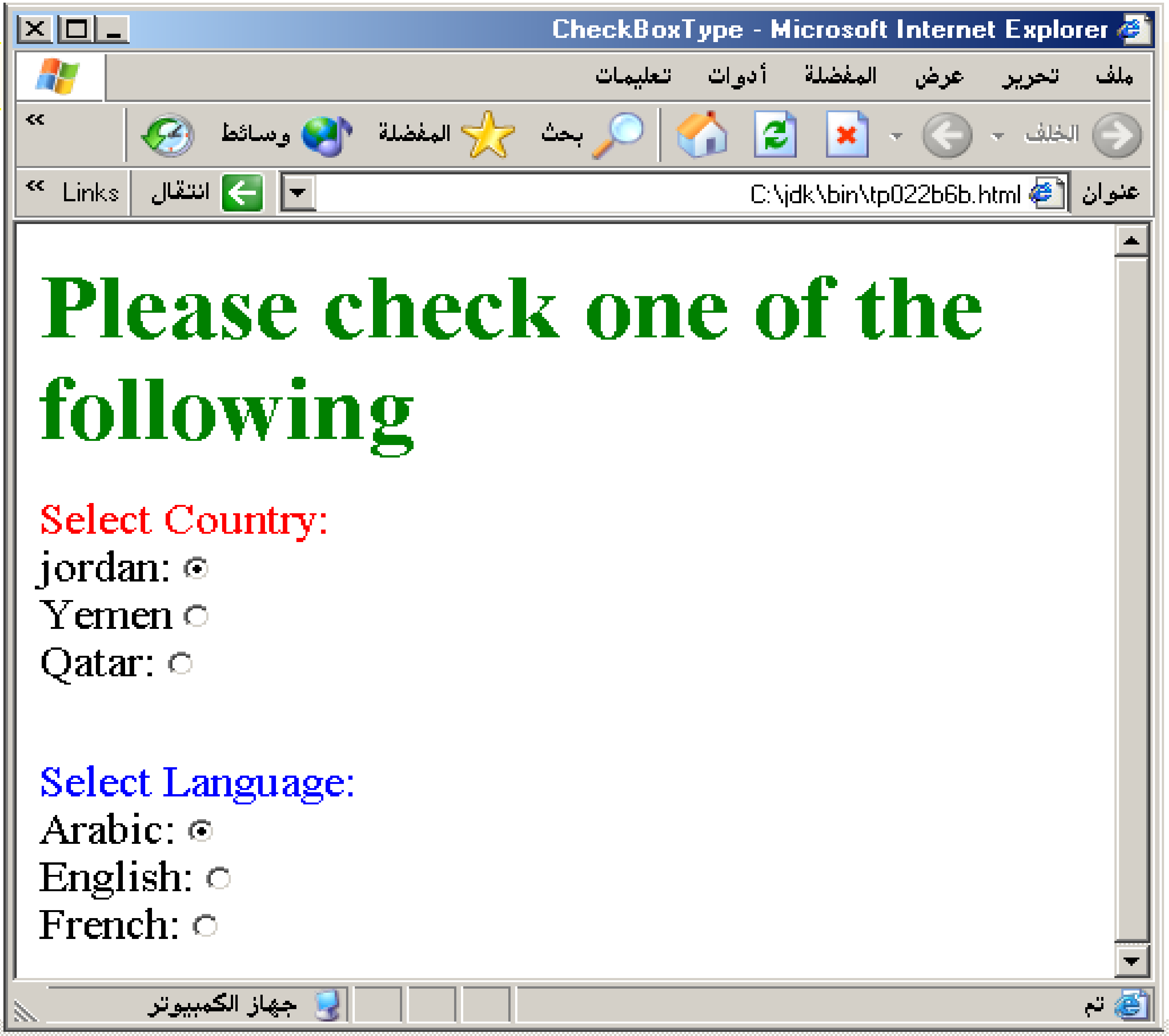
<INPUT TYPE="RADIO">

Browser will display 

Radio buttons have the following attributes:

- **TYPE:** radio.
- **CHECKED:** is blank or CHECKED as the initial status. Only one radio button can be checked
- **NAME:** is the name of the variable to be sent to the CGI application.
- **VALUE:** usually has a set value.

```
<HTML> <HEAD><TITLE>CheckBoxType</TITLE> </HEAD>
<BODY>
<h1> <font color=green>Please check one of the
following</font></h1>
<FORM name="fome3"  Action="url"  method="get">
<font color=red> Select Country: </font><BR>
jordan:<INPUT TYPE= "RADIO"  Name="country"
CHECKED><BR>
Yemen<INPUT TYPE="RADIO "  Name="country"><BR>
Qatar:<INPUT TYPE="RADIO"  Name="country"><BR>
<BR>
<font color=blue>Select Language:</font><BR>
Arabic:<INPUT TYPE="RADIO"  Name="language"
CHECKED><BR> English:<INPUT TYPE=" RADIO "
Name="language"><BR>
French:<INPUT TYPE=" RADIO "  Name="language">
<BR></FORM> </BODY></HTML>
```


CheckBoxType - Microsoft Internet Explorer
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان C:\jdk\bin\tp022b6b.html انتقال Links
تم جهاز الكمبيوتر

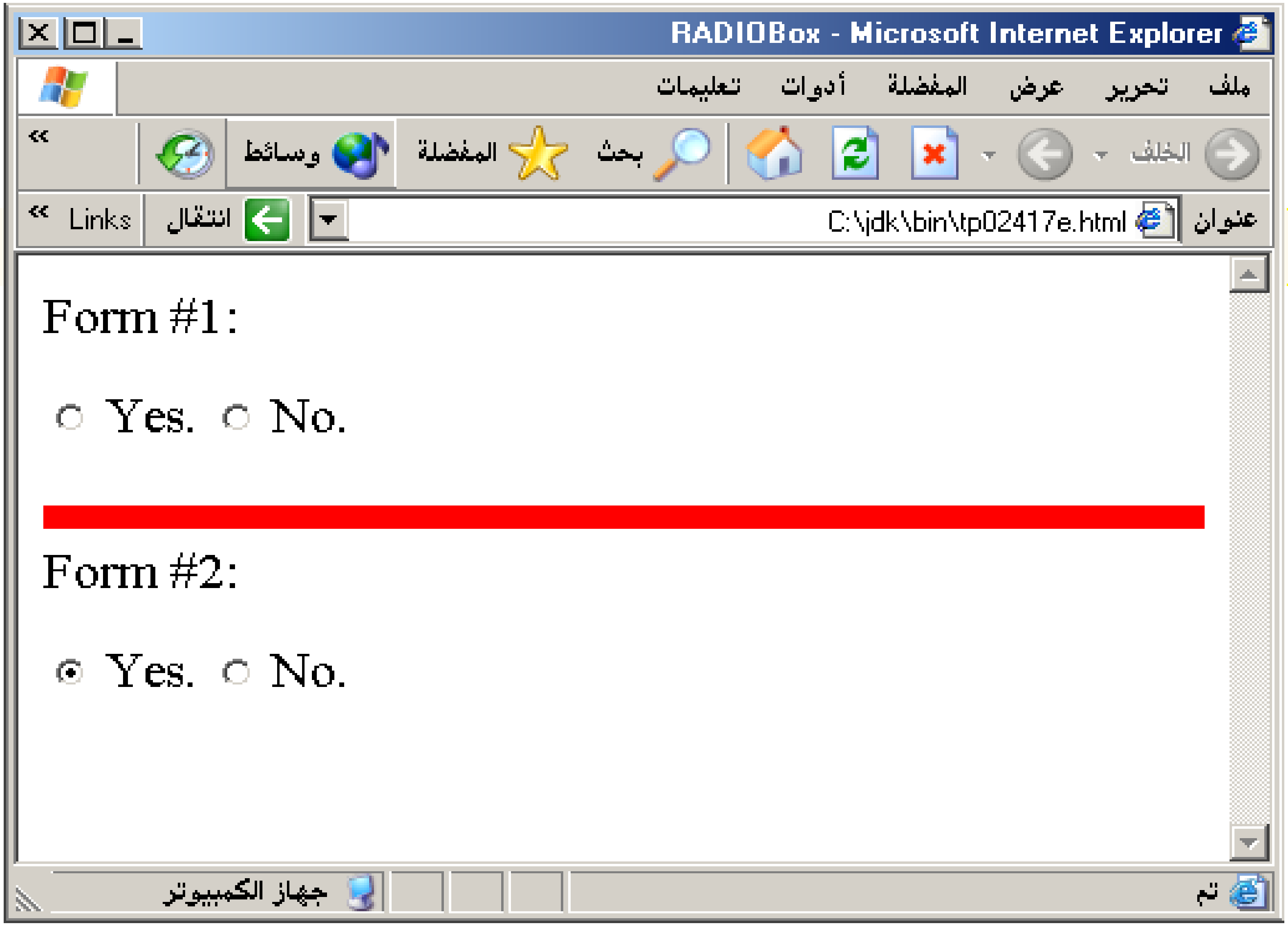
Please check one of the following

Select Country:
jordan: ●
Yemen ○
Qatar: ○

Select Language:
Arabic: ●
English: ○
French: ○

```
<HTML><HEAD>
<TITLE>RADIOBox</TITLE> </HEAD>
<BODY>
Form #1:
<FORM>
  <INPUT TYPE="radio" NAME="choice" VALUE="one"> Yes.
   <INPUT TYPE="radio" NAME="choice" VALUE="two"> No.
</FORM>
<HR color=red size="10" >
Form #2:
<FORM>
      <INPUT TYPE="radio" NAME="choice" VALUE="three"
CHECKED> Yes.
   <INPUT TYPE="radio" NAME="choice" VALUE="four"> No.
</FORM>
</BODY></HTML>
```

Output


RADIOBox - Microsoft Internet Explorer
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان C:\jdk\bin\tp02417e.html انتقال Links
تم جهاز الكمبيوتر

Form #1:

○ Yes. ○ No.

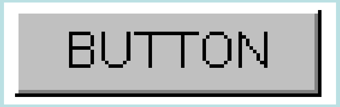
Form #2:

◉ Yes. ○ No.

Push Button

- **Push Button:** This element would be used with JavaScript to cause an action to take place.

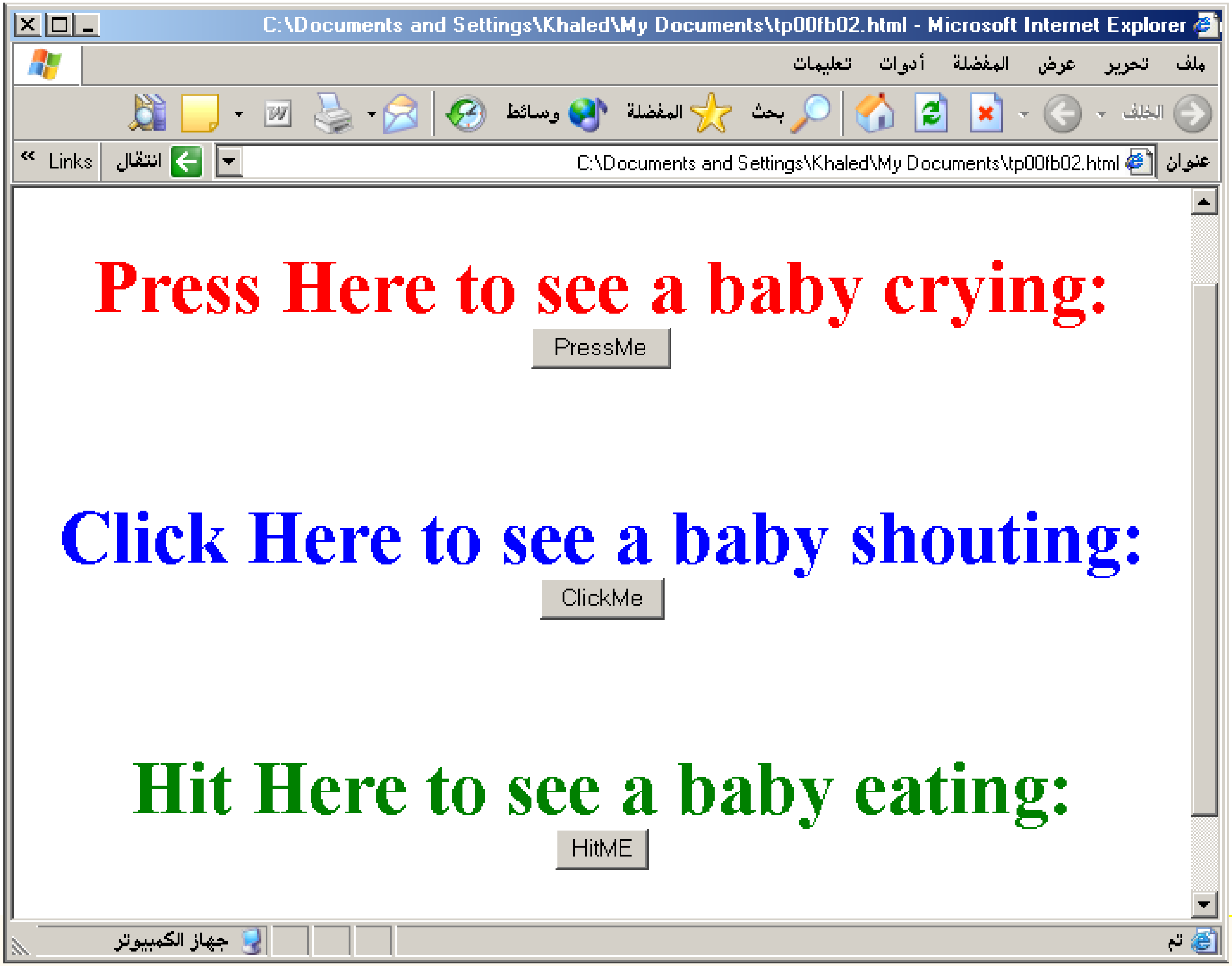
<INPUT TYPE="BUTTON">

Browser will display 

Push Button has the following attributes:

- **TYPE:** button.
- **NAME:** is the name of the button to be used in scripting.
- **VALUE:** determines the text label on the button.

```
<DIV align=center><BR><BR>
<FORM>
<FONT Color=red>
<h1>Press Here to see a baby crying:<BR>
<INPUT TYPE="button"
   VALUE="PressMe"><BR><BR>
<FONT Color=blue>
Click Here to see a baby shouting:<BR>
<INPUT TYPE="button" VALUE="ClickMe" >
   <BR><BR>
<FONT Color=green>
Hit Here to see a baby eating:<BR>
<INPUT TYPE="button" VALUE="HitME" > <BR><BR>
<FONT Color=yellow>
</FORM></DIV>
```


C:\Documents and Settings\Khaled\My Documents\tp00fb02.html - Microsoft Internet Explorer
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان C:\Documents and Settings\Khaled\My Documents\tp00fb02.html
انتقال Links

Press Here to see a baby crying:

PressMe

Click Here to see a baby shouting:

ClickMe

Hit Here to see a baby eating:

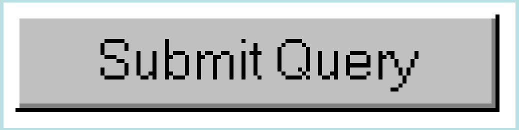
HitME

Submit Button

- **Submit:** Every set of Form tags requires a Submit button. This is the element causes the browser to send the names and values of the other elements to the CGI Application specified by the ACTION attribute of the FORM element.

<INPUT TYPE=“SUBMIT”>

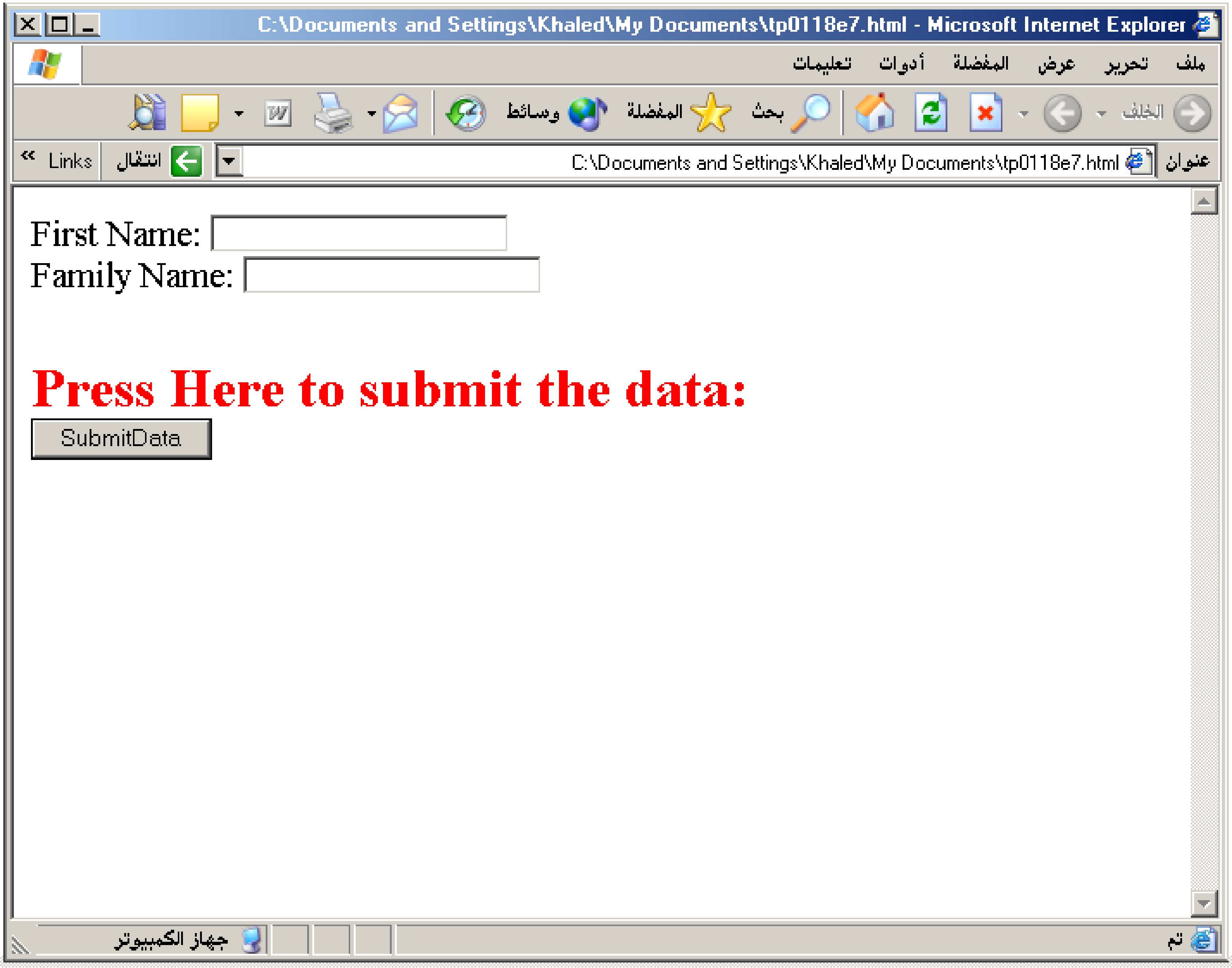
The browser will display



Submit has the following attributes:

- **TYPE:** submit.
- **NAME:** value used by the CGI script for processing.
- **VALUE:** determines the text label on the button, usually Submit Query.

```
<FORM    Action="URL"        method="get">
First Name: <INPUT TYPE="TEXT" Size=25
name="firstName"><BR>
Family Name: <INPUT TYPE="TEXT" Size=25
name="LastName"><BR>
<BR>
<FONT Color=red>
Press Here to submit the data:<BR>
<INPUT TYPE="submit" VALUE="SubmitData " >
</FORM>
```


C:\Documents and Settings\Khaled\My Documents\tp0118e7.html - Microsoft Internet Explorer
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان C:\Documents and Settings\Khaled\My Documents\tp0118e7.html
انتقال Links
تم جهاز الكمبيوتر

First Name:
Family Name:

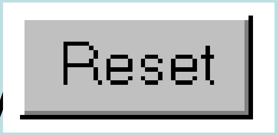
Press Here to submit the data:

SubmitData

Reset Button

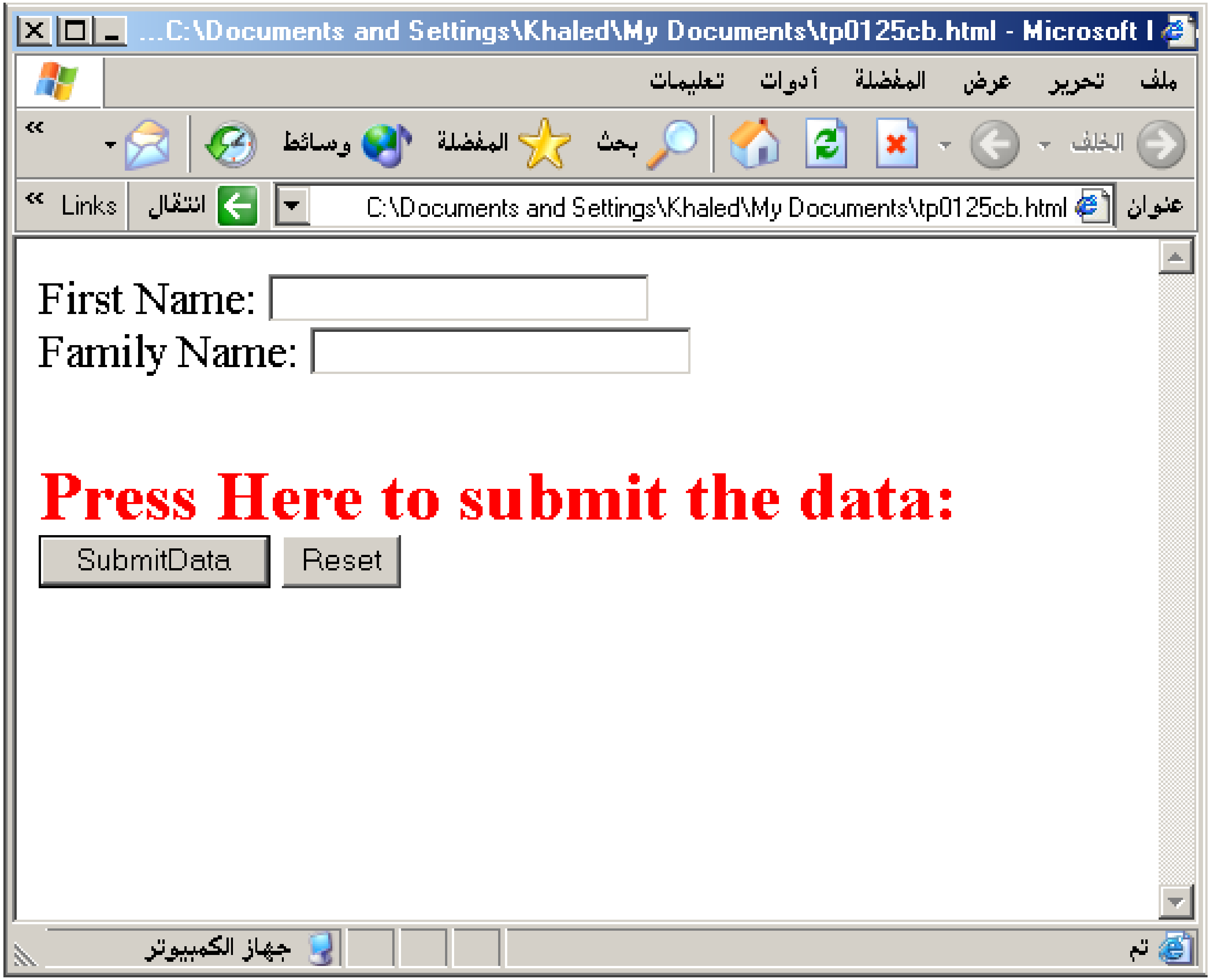
- **Reset:** It is a good idea to include one of these for each form where users are entering data. It allows the surfer to clear all the input in the form.

- **<INPUT TYPE=“RESET”>**

- Browser will display 

-
- Reset buttons have the following attributes:
- **TYPE**: reset.
- **VALUE**: determines the text label on the button, usually Reset.

```
<FORM    Action="URL"        method="get">
First Name: <INPUT TYPE="TEXT" Size=25
name="firstName"> <BR>
Family Name: <INPUT TYPE="TEXT" Size=25
name="LastName"><BR>
<BR>
<FONT Color = red>
<STRONG><font size=5>Press Here to submit
the data:</font></STRONG><BR>
<INPUT TYPE="submit" VALUE="SubmitData">
<INPUT TYPE="RESET" VALUE="Reset">
</FORM>
```


...C:\Documents and Settings\Khaled\My Documents\tp0125cb.html - Microsoft I
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف
بحث
المفضلة
وسائط
عنوان
C:\Documents and Settings\Khaled\My Documents\tp0125cb.html
انتقال
Links
تم
جهاز الكمبيوتر

First Name:

Family Name:

Press Here to submit the data:

SubmitData Reset

Image Submit Button

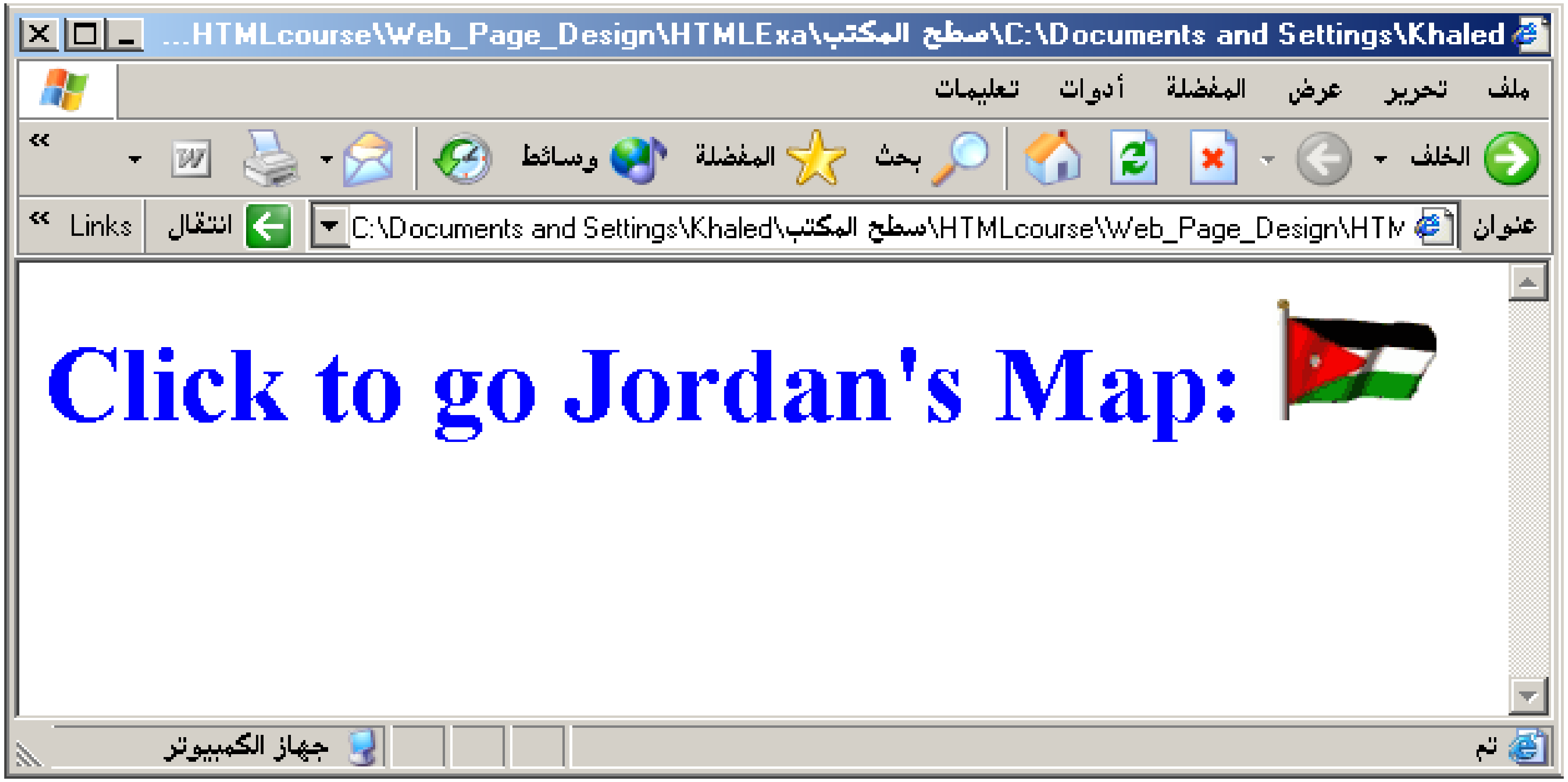
- **Image Submit Button:** Allows you to substitute an image for the standard submit button.

<INPUT TYPE="IMAGE" SRC="jordan.gif">

Image submit button has the following attributes:

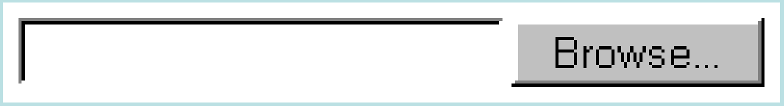
- **TYPE:** Image.
- **NAME:** is the name of the button to be used in scripting.
- **SRC:** URL of the Image file.

```
<form>
<H1><font color=blue>
Click to go Jordan’s Map:
<INPUT  TYPE="IMAGE"  SRC="jordan.gif">
</form>
```


...HTMLcourse\Web_Page_Design\HTMLExa\سطح المكتب\C:\Documents and Settings\Khaled
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان C:\Documents and Settings\Khaled\سطح المكتب\HTMLcourse\Web_Page_Design\HTM
انتقال Links
تم
جهاز الكمبيوتر

Click to go Jordan's Map:

File

- **File Upload:** You can use a file upload to allow surfers to upload files to your web server.
- **<INPUT TYPE=“FILE”>**
- Browser will display 

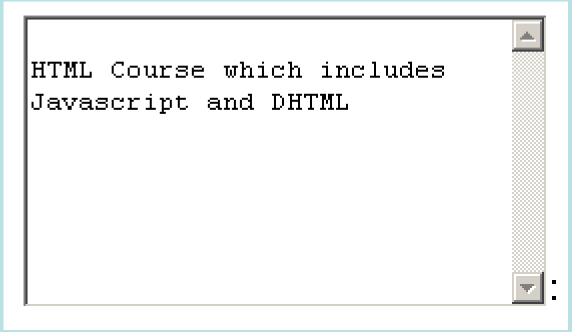
- File Upload has the following attributes:
- **TYPE:** file.
- **SIZE:** is the size of the text box in characters.
- **NAME:** is the name of the variable to be sent to the CGI application.
- **MAXLENGHT:** is the maximum size of the input in the textbox in characters.

```
<BODY bgcolor=lightblue>
<form>
<H3><font color=forestgreen>
Please attach your file here to for uploading to
My <font color =red>SERVER...<BR>

<INPUT  TYPE="File"  name="myFile"
   size="30">


<INPUT  TYPE="Submit"  value="SubmitFile">
</form>
</BODY>
```

Other Elements used in Forms

- **<TEXTAREA></TEXTAREA>:** is an element that allows for free form text entry.

Browser will display



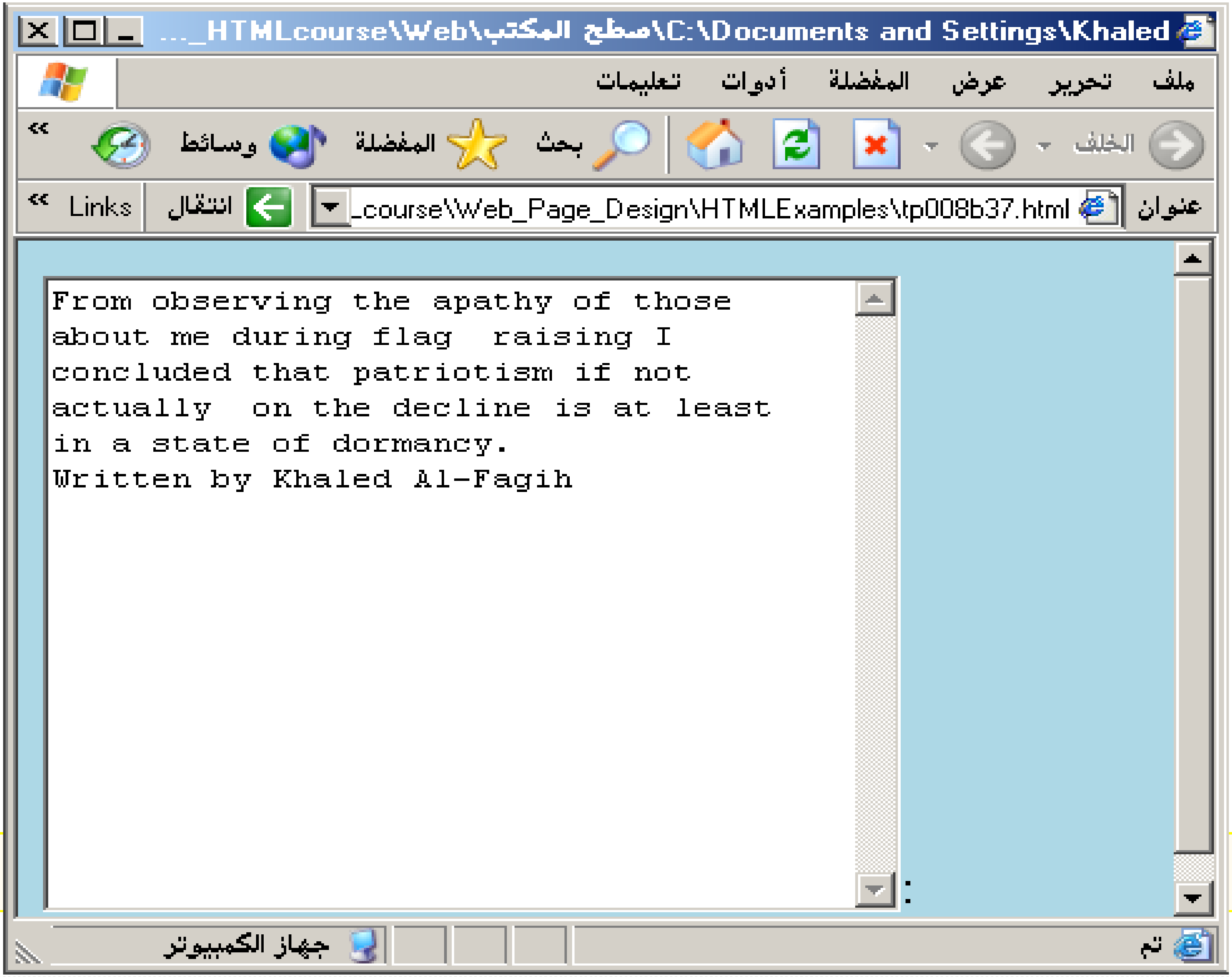
```
HTML Course which includes
Javascript and DHTML
```

:

Textarea has the following attributes:

- **NAME**: is the name of the variable to be sent to the CGI application.
- **ROWS**: the number of rows to the textbox.
- **COLS**: the number of columns to the textbox.

```
<BODY bgcolor=lightblue>
<form>
<TEXTAREA   COLS=40  ROWS=20
Name="comments"   >
From observing the apathy of those
about me during flag  raising I
concluded that patriotism if not
actually  on the decline is at least
in a state of dormancy.
Written by Khaled Al-Fagih
</TEXTAREA>:
</form>
</BODY>
```


..._HTMLcourse\Web\المكتب مطع\C:\Documents and Settings\Khaled
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان _course\Web_Page_Design\HTMLExamples\tp008b37.html
انتقال Links
تم
جهاز الكمبيوتر

```
From observing the apathy of those
about me during flag  raising I
concluded that patriotism if not
actually  on the decline is at least
in a state of dormancy.
Written by Khaled Al-Fagih
```

10.6.1.2 The wrap attribute

Normally, text typed in the text area by the user is transmitted to the server exactly as typed, with lines broken only where the user pressed the Enter key. Since this is often not the desired action by the user, you can enable word wrapping within the text area. When the user types a line that is longer than the width of the text area, the browser automatically moves the extra text down to the next line, breaking the line at the nearest point between words in the line.

With the `wrap` attribute set to `virtual`, the text is wrapped within the text area for presentation to the user, but the text is transmitted to the server as if no wrapping had occurred, except where the user pressed the Enter key.

With the `wrap` attribute set to `physical`, the text is wrapped within the text area and is transmitted to the server as if the user had actually typed it that way. This the most useful way to use word wrap, since the text is transmitted exactly as the user sees it in the text area.

To obtain the default action, set the `wrap` attribute to `off`.

As an example, consider the following 60 characters of text being typed into a 40-character-wide text area:

```
Word wrapping is a feature that makes life easier for users.
```

With `wrap=off`, the text area will contain one line and the user will have to scroll to the right to see all of the text. One line of text will be transmitted to the server.

With `wrap=virtual`, the text area will contain two lines of text, broken after the word "makes." Only one line of text will be transmitted to the server: the entire line with no embedded newline characters.

With `wrap=physical`, the text area will contain two lines of text, broken after the word "makes." Two lines of text will be sent to the server, separated by a newline character after the word "makes."

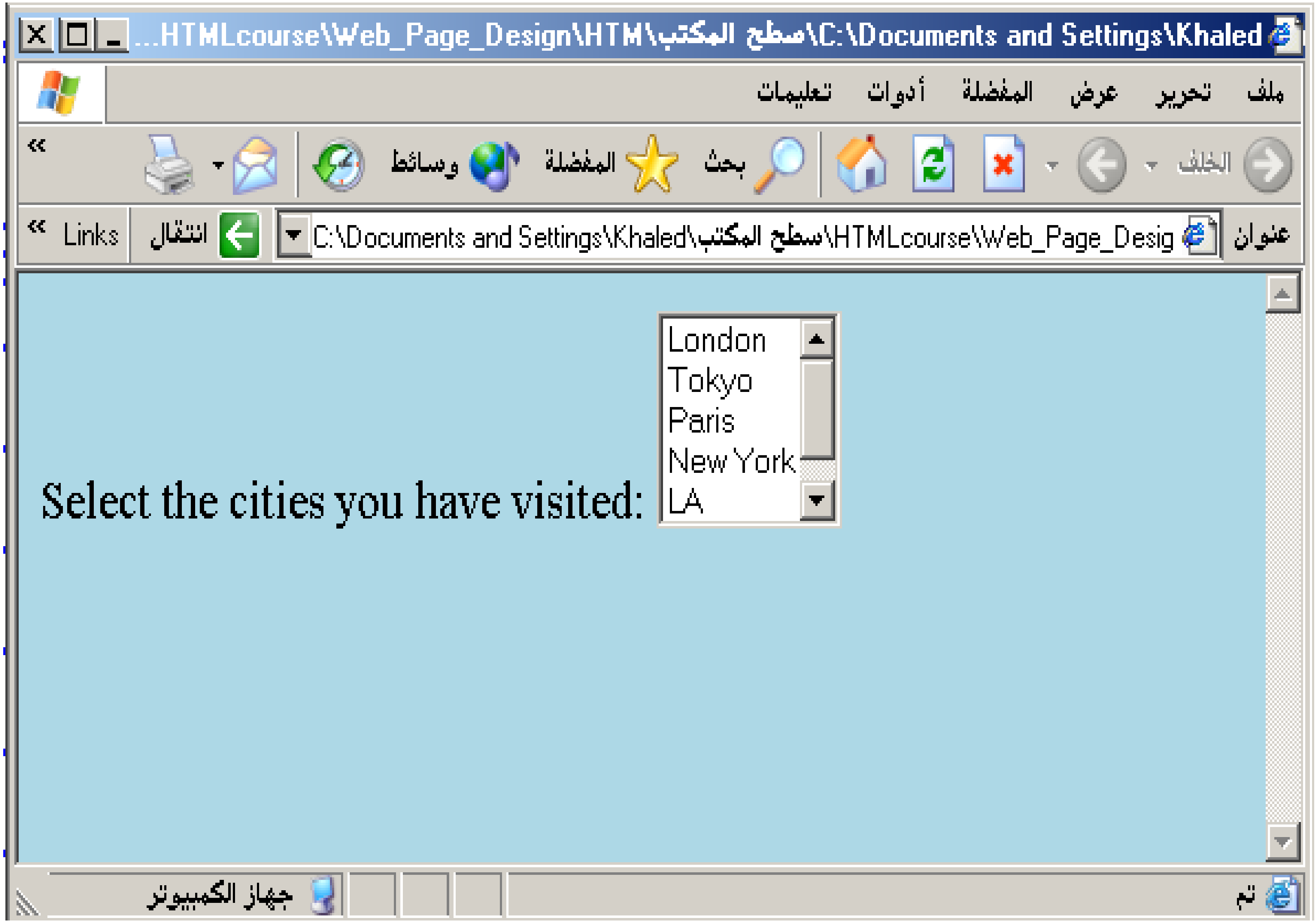
Other Elements used in Forms

- The two following examples are **<SELECT></SELECT>** elements, where the attributes are set differently.

The Select elements attributes are:

- **NAME:** is the name of the variable to be sent to the CGI application.
- **SIZE:** this sets the number of **visible** choices.
- **MULTIPLE:** the presence of this attribute signifies that the user can make multiple selections. By default only one selection is allowed.

```
<BODY bgcolor=lightblue>
<form>
Select the cities you have visited:
<SELECT name=“list”  size=5>
<option> London</option>
<option> Tokyo</option>
<option> Paris</option>
<option> New York</option>
<option> LA</option>
<option> KL</option>
</SELECT>
</form>
</BODY>
```

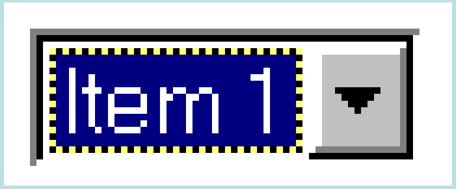

...HTMLcourse\Web_Page_Design\HTM\سطح المكتب\C:\Documents and Settings\Khaled
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان C:\Documents and Settings\Khaled\سطح المكتب\HTMLcourse\Web_Page_Desig
انتقال Links
تم
جهاز الكمبيوتر

Select the cities you have visited:

London
Tokyo
Paris
New York
LA

Other Elements used in Forms

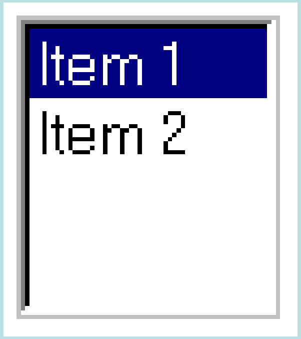
- **Drop Down List:**



- **Name:** is the name of the variable to be sent to the CGI application.
- **Size:** 1.

Other Elements used in Forms

- **List Box:**



- **Name:** **is the name of the variable to be sent to the CGI application.**
- **SIZE:** **is greater than one.**

Other Elements used in Forms

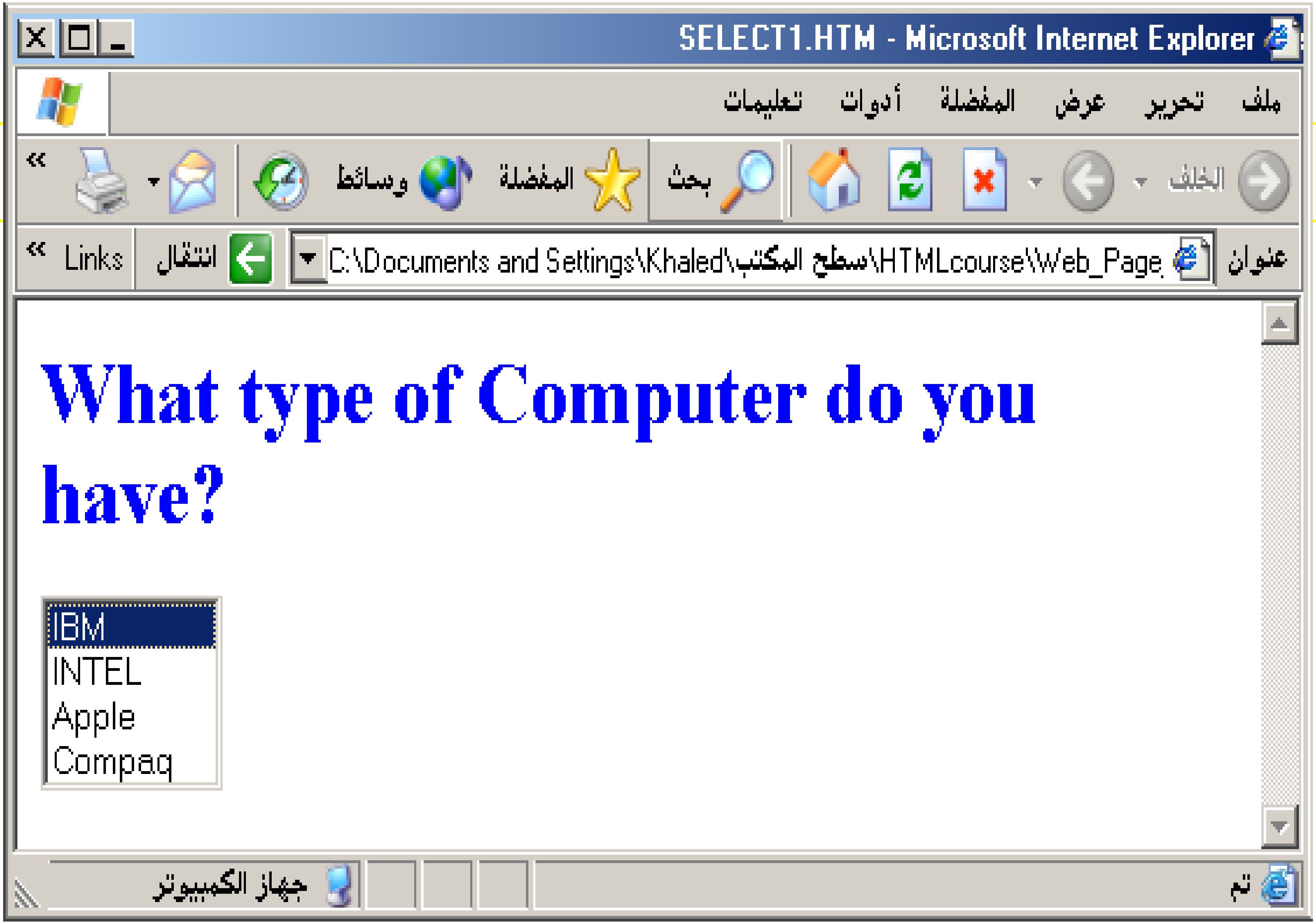
- **Option**

The list items are added to the **<SELECT>** element by inserting **<OPTION></OPTION>** elements.

The Option Element’s attributes are:

- **SELECTED:** When this attribute is present, the option is selected when the document is initially loaded. **It is an error for more than one option to be selected.**
- **VALUE:** Specifies the value the variable named in the select element.

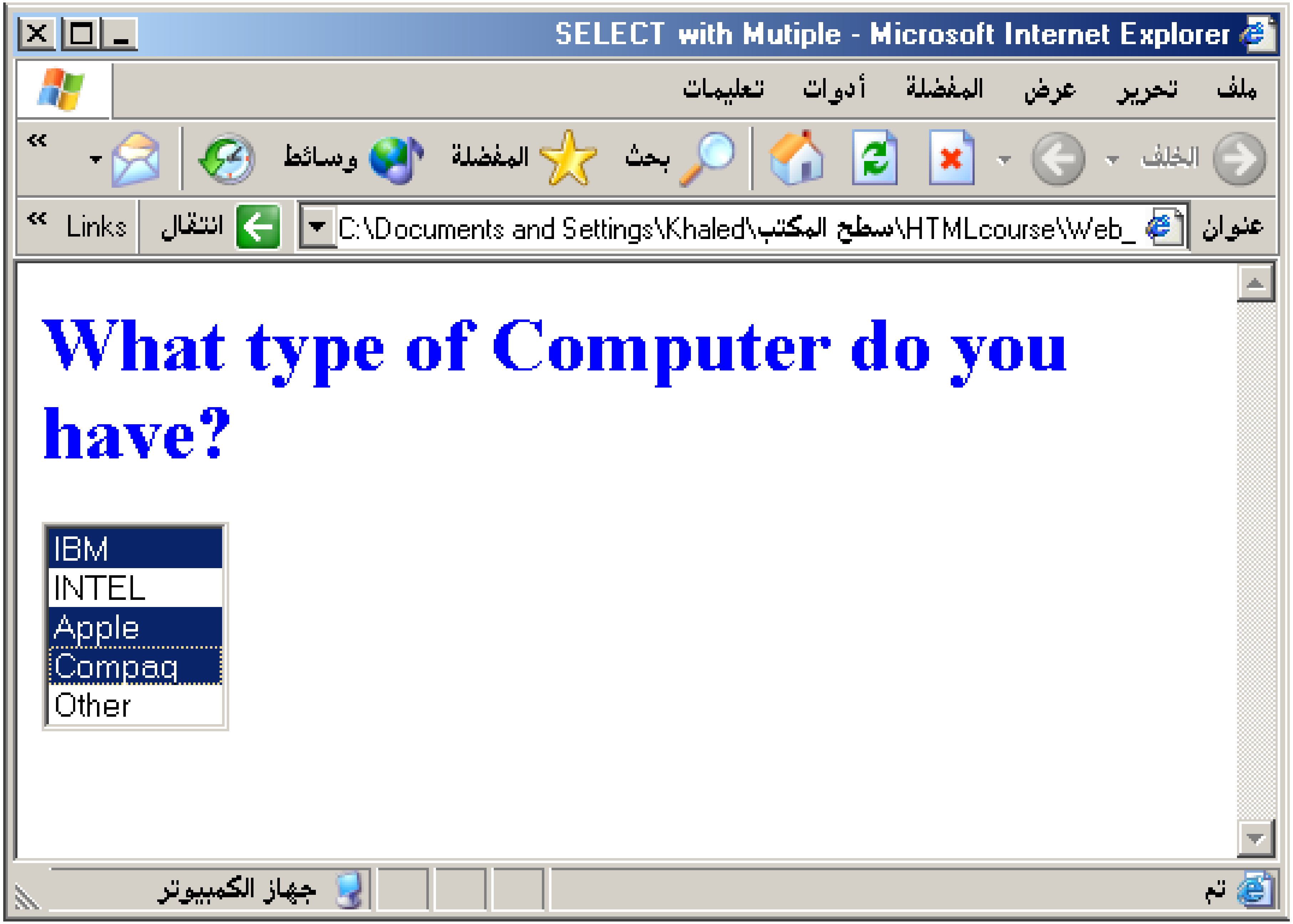
```
</HEAD>
<BODY>
<h2><font color=blue>What type of Computer do you
   have?</font><h2>
<FORM>
<SELECT NAME="ComputerType" size=4>
   <OPTION  value="IBM" SELECTED> IBM</OPTION>
   <OPTION  value="INTEL"> INTEL</OPTION>
   <OPTION value=" Apple"> Apple</OPTION>
   <OPTION value="Compaq"> Compaq</OPTION>
</SELECT>
</FORM></BODY></HTML>
```


SELECT1.HTM - Microsoft Internet Explorer
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان C:\Documents and Settings\Khaled\سطح المكتب\HTMLcourse\Web_Page
انتقال Links
تم
جهاز الكمبيوتر

What type of Computer do you have?

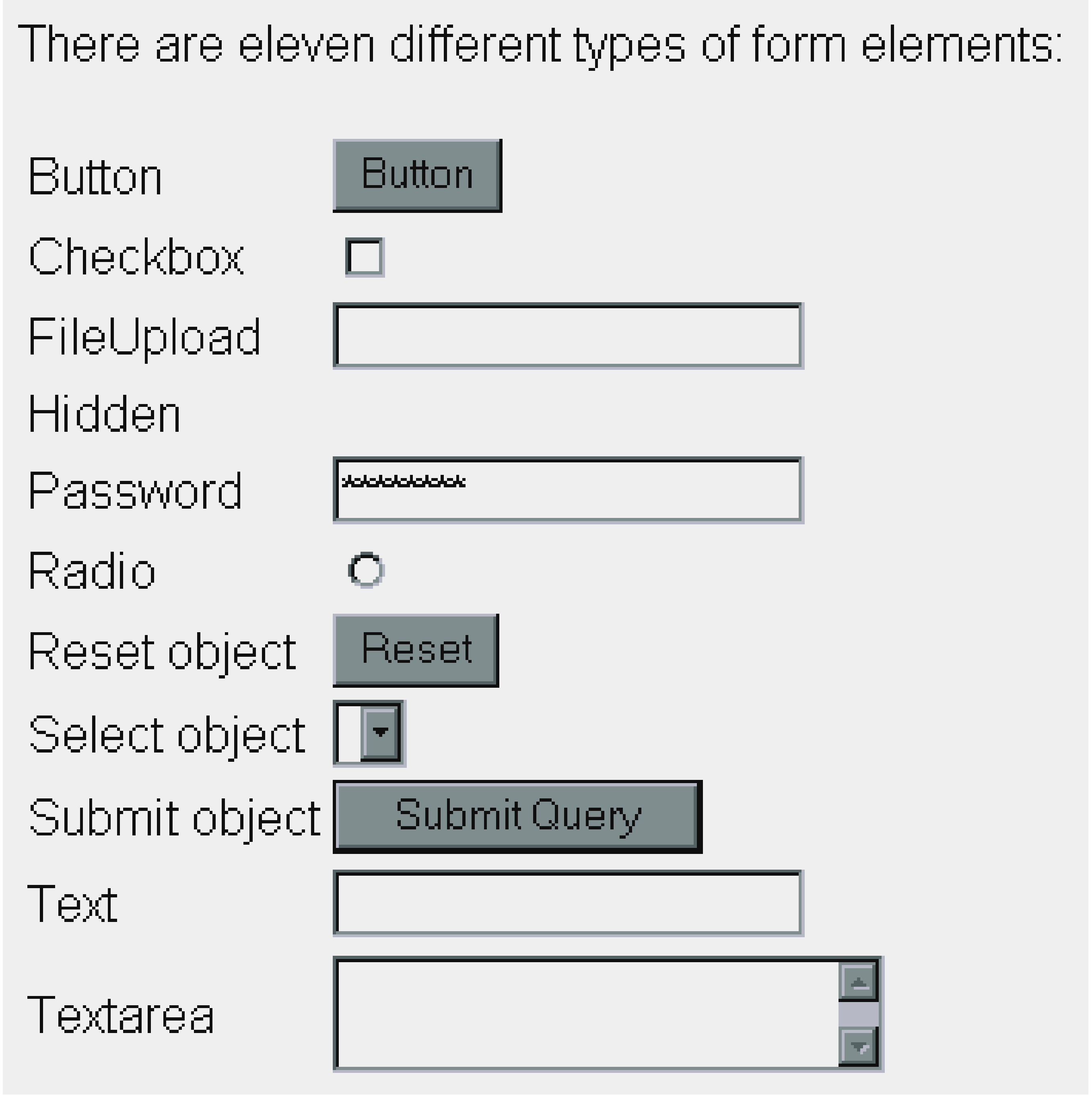
IBM
INTEL
Apple
Compaq

```
<HEAD> <TITLE>SELECT with Mutiple </TITLE>
   </HEAD>
<BODY>
<h2><font color=blue>What type of Computer do you
   have?</font><h2>
<FORM>
<SELECT NAME="ComputerType" size=5   multiple>
   <OPTION  value="IBM" > IBM</OPTION>
   <OPTION  value="INTEL"> INTEL</OPTION>
   <OPTION value=" Apple"> Apple</OPTION>
   <OPTION value="Compaq" SELECTED>
   Compaq</OPTION>
   <OPTION value=" other"> Other</OPTION>
</SELECT>
</FORM></BODY></HTML>
```


SELECT with Mutiple - Microsoft Internet Explorer
ملف تحرير عرض المفضلة أدوات تعليمات
الخلف بحث المفضلة وسائط
عنوان C:\Documents and Settings\Khaled\سطح المكتب\HTMLcourse\Web_
انتقال Links
تم جهاز الكمبيوتر

What type of Computer do you have?

IBM
INTEL
Apple
Compaq
Other



There are eleven different types of form elements:

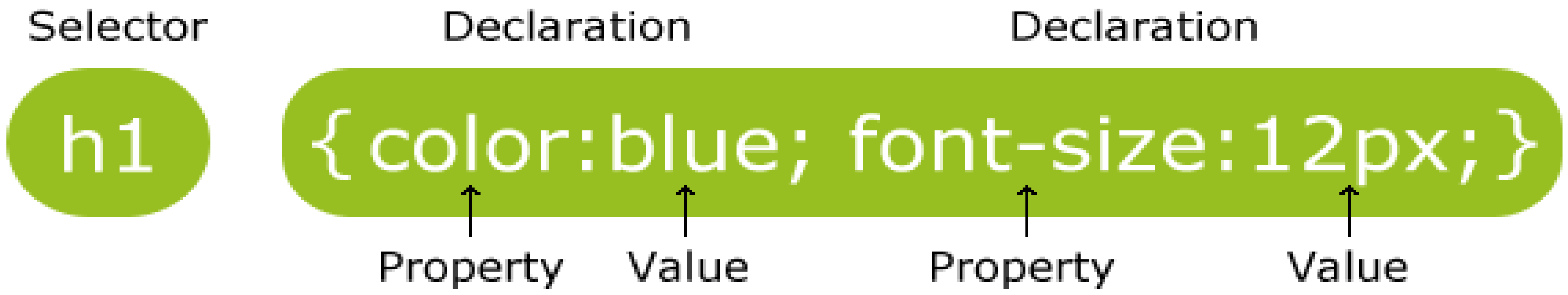
Button
Checkbox
FileUpload
Hidden
Password
Radio
Reset object
Select object
Submit object
Text
Textarea

Cascading Style Sheets

.

CSS

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



- A CSS rule has two main parts: a selector, and one or more declarations:
- The selector is normally the HTML element you want to style.
- Each declaration consists of a property and a value.
- The property is the style attribute you want to change. Each property has a value.

Internal Stylesheet

First we will explore the internal method.
This way you are simply placing the CSS code within the <head></head> tags of each HTML file you want to style with the CSS. The format for this is shown in the example below

- <head>
- <title><title>
- <style type="text/css">

  CSS Content Goes Here
- </style>
- </head>
- <body>

With this method each HTML file contains the CSS code needed to style the page. Meaning that any changes you want to make to one page, will have to be made to all. This method can be good if you need to style only one page, or if you want different pages to have varying styles.

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

```
<html>
<head>
<style type="text/css">
h1 {color:red;}
h2 {color:blue;}
p {color:green;}
</style>
</head>
<body>
<h1>All header 1 elements will be red</h1>
<h2>All header 2 elements will be blue</h2>
<p>All text in paragraphs will be green.</p>
</body>
</html>
```

JavaScript Introduction

JavaScript was released by Netscape and Sun Microsystems in 1995. However, JavaScript is not the same thing as Java.

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)

Put a JavaScript into an HTML page

To insert a JavaScript into an HTML page, we use the <script> tag. Inside the <script> tag we use the type attribute to define the scripting language.

```
<script type="text/JavaScript">
...some JavaScript

</script
```

Installation is not required, nor do you have to torturously work through any odd library path configurations. JavaScript works, straight out of the box and in most web browsers, including the big four: Firefox, Internet Explorer, Opera, and Safari. All you need to do is add a scripting block, and you’re in business.

The **document.write** command is a standard JavaScript command for writing output to a page.

By entering the document.write command between the <script> and </script> tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! to the page

```
<script type="text/javascript">
document.write("Hello World!");
</script>
```

The example below shows how to add HTML tags to the JavaScript:

```
document.write("<h1>Hello World!</h1>");
```

JavaScript is a sequence of statements to be executed by the browser.

JavaScript is Case Sensitive

Unlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.

JavaScript Statements

A JavaScript statement is a command to a browser. The purpose of the command is to tell the browser what to do. This JavaScript statement tells the browser to write "Hello Dolly" to the web page.

It is normal to add a semicolon at the end of each executable statement. Most people think this is a good programming practice, and most often you will see this in JavaScript examples on the web.

The semicolon is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. Because of this you will often see examples without the semicolon at the end.

JavaScript code (or just JavaScript) is a sequence of JavaScript statements. Each statement is executed by the browser in the sequence they are written.

This example will write a heading and two paragraphs to a web page:

```
<script type="text/javascript">
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Blocks

JavaScript statements can be grouped together in blocks. Blocks start with a left curly bracket {, and ends with a right curly bracket }. The purpose of a block is to make the sequence of statements execute together.

This example will write a heading and two paragraphs to a web page

```
<script type="text/javascript">
{
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
}
</script>
```

The example above is not very useful. It just demonstrates the use of a block. Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).

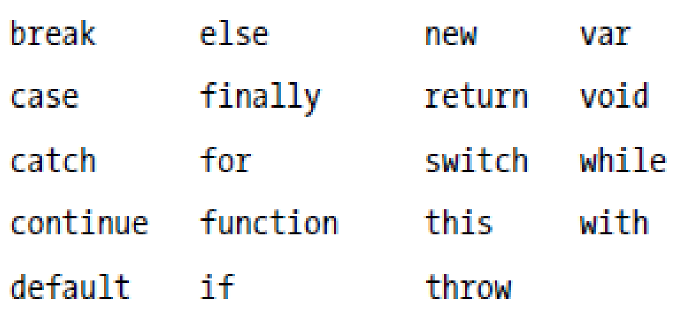
JavaScript Variables

Variables in JavaScript are much like those in any other language; you use them to hold values in such a way that the values can be explicitly accessed in different places in the code. Each has an identifier that is unique to the scope of use, consisting of any combination of letters, digits, underscores, and dollar signs. An identifier

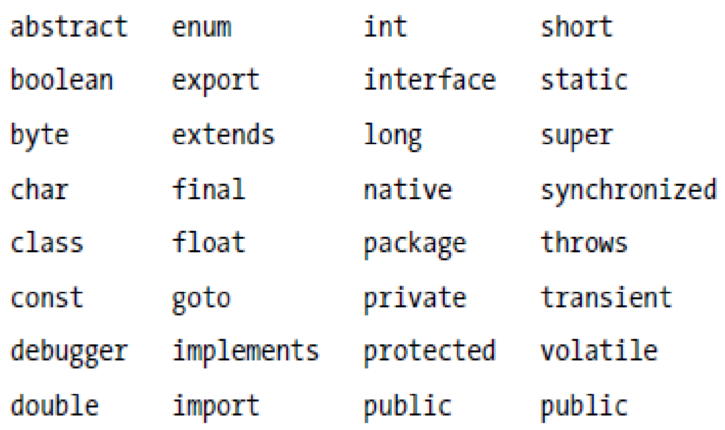
Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter or the underscore character

Table JavaScript keyword



break	else	new	var
case	finally	return	void
catch	for	switch	while
continue	function	this	with
default	if	throw	



abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	public

Creating JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the **var** keyword:

```
var x;
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).

However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Volvo";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.

Note: When you assign a text value to a variable, use quotes around the value.

The following snippet of code assigns a string literal containing a line-terminator escape sequence to a variable. When the string is used in a dialog window, the escape sequence, **\n**, is interpreted literally, and a newline is published:

var string_value = "This is the first line\nThis is the second line";

This results in:

This is the first line

This is the second line

Redeclaring JavaScript Variables

If you re declare a JavaScript variable, it will not lose its original value.

Var x=5;
var x;

After the execution of the statements above, the variable x will still have the value of 5. The value of x is not reset (or cleared) when you redeclare it.

JavaScript Arithmetic

As with algebra, you can do arithmetic operations with JavaScript variables:

y=x-5;
z=y+5;

You will learn more about the operators that can be used in the next chapter of this tutorial.

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3

*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	**Example**	**Same As**	**Result**
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".

To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

Adding Strings and Numbers

The rule is: If you add a number and a string, the result will be a string! **Example**

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for true or false.

Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false

!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

<u>Logical Operators</u>

Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	And	(x < 10 && y > 1) is true
||	Or	(x==5 || y==5) is false
!	Not	!(x==y) is true

<u>Conditional Operator</u>

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

```
variablename=(condition)?value1:value2
```

<u>Conditional Statements</u>

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false

- **if...else if....else statement** - use this statement to select one of many blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

If Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

```
If (condition)
  {
  code to be executed if condition is true   }
```

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10
var d=new Date();
var time=d.getHours();
if (time<10)
  {
  document.write("<b>Good morning</b>"); }
</script>
```

Notice that there is no ..else.. in this syntax. You tell the browser to execute some code **only if the specified condition is true**.

If...else Statement

Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
  {
  code to be executed if condition is true
  }
else
  {
```

```
 code to be executed if condition is not true
 }
```

If...else if...else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

Syntax

```
if (condition1)
 {
 code to be executed if condition1 is true
 }
else if (condition2)
 {
 code to be executed if condition2 is true
 }
else
 {
 code to be executed if condition1 and condition2 are not true
 }
```

The JavaScript Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch(n)
{
case 1:
 execute code block 1
 break;
case 2:
 execute code block 2
 break;
default:
 code to be executed if n is different from case 1 and 2
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

Example

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
var d=new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("Finally Friday");
  break;
case 6:
  document.write("Super Saturday");
  break;
case 0:
  document.write("Sleepy Sunday");
  break;
default:
  document.write("I'm looking forward to this weekend!");
}
</script>
```

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
Alert("sometext");
```

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
confirm("sometext");
```

Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
prompt("sometext","defaultvalue");
```

JavaScript Functions

A function will be executed by an event or by a call to the function. To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed by an event or by a call to the function.

You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax

```
function functionname(var1,var2,...,varX)
{
some code
}
```

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

Example

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");   }
</script>
</head>
<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```

If the line: alert("Hello world!!") in the example above had not been put within a function, it would have been executed as soon as the page was loaded. Now, the script is not executed before a user hits the input button. The function displaymessage() will be executed if the input button is clicked.

You will learn more about JavaScript events in the JS Events chapter.

The return Statement

The return statement is used to specify the value that is returned from the function.So, functions that are going to return a value must use the return statement. The example below returns the product of two numbers (a and b):

Example

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3));
</script></body> </html>
```

The Lifetime of JavaScript Variables

If you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are <html>

Example

```
<head>

<script type="text/javascript">

function myfunction(txt)

{ alert(txt);}
```

</script> </head> <body>

<form>

<input type="button" onclick="myfunction('Hello')" value="Call function">

</form>

<p>By pressing the button above, a function will be called with "Hello" as a parameter. The function will alert the parameter.</p>

</body> </html>

JavaScript Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
code to be executed
}
```

The while Loop

The while loop loops through a block of code while a specified condition is true.

Syntax

```
while (var<=endvalue)
  {
  code to be executed
  }
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
do
  {
  code to be executed
  }
while (var<=endvalue);
```

The break Statement

The break statement will break the loop and continue executing the code that follows after the loop (if any).

The continue Statement

The continue statement will break the current loop and continue with the next value

What is an Array?

An array is a special variable, which can hold more than one value, at a time. If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
cars1="Saab";
cars2="Volvo";
cars3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array! An array can hold all your variable values under a single name. And you can access the values by referring to the array name. Each element in the array has its own ID so that it can be easily accessed.

JavaScript For...In Statement

The for...in statement loops through the elements of an array or through the properties of an object.

Syntax

```
for (variable in object)
  {
  code to be executed
  }
```

Create an Array

An array can be defined in three ways. The following code creates an Array object called myCars

1:

```
var myCars=new Array(); // regular array (add an optional integer
myCars[0]="Saab";       // argument to control array's size)
myCars[1]="Volvo";
myCars[2]="BMW";
```

2:

```
var myCars=new Array("Saab","Volvo","BMW"); // condensed array
```

3:

var myCars=["Saab","Volvo","BMW"]; // literal array

Note: If you specify numbers or true/false values inside the array then the variable type will be Number or Boolean, instead of String.

Access an Array

You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0. The following code line:

document.write(myCars[0]);
will result in the following output: Saab

Modify Values in an Array

To modify a value in an existing array, just add a new value to the array with a specified index number:

myCars[0]="Opel";

Now, the following code line:

document.write(myCars[0]);
will result in the following output: Opel

Arrays - concat()

```
var parents = ["Jani", "Tove"];

var children = ["Cecilie", "Lone"];

var family = parents.concat(children);

document.write(family);
```

Insert Special Characters

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

Look at the following JavaScript code:

var txt="We are the so-called "Vikings" from the north.";

document.write(txt);

In JavaScript, a string is started and stopped with either single or double quotes. This means that the string above will be chopped to: We are the so-called

To solve this problem, you must place a backslash (\) before each double quote in "Viking". This turns each double quote into a string literal:

var txt="We are the so-called \"Vikings\" from the north."; document.write(txt);

JavaScript will now output the proper text string: We are the so-called "Vikings" from the north.

Here is another example:

document.write ("You \& I are singing!");

The example above will produce the following output:

You & I are singing!

The table below lists other special characters that can be added to a text string with the backslash sign:

Code	**Outputs**
\'	**single quote**
\"	**Double quote**
\&	**ampersand**
\\	**backslash**
\n	**new line**
\r	**carriage return**
\t	**Tab**
\b	**backspace**
\f	**form feed**

String object

1- Return the length of a string

```
var txt = "Hello World!";  document.write(txt.length);
```

2- Style strings

```
var txt = "Hello World!";

document.write("<p>Big: " + txt.big() + "</p>");

document.write("<p>Bold: " + txt.bold() + "</p>");

document.write("<p>Italic: " + txt.italics() + "</p>");
```

3- The toLowerCase() and toUpperCase() methods

```
var txt="Hello World!";

document.write(txt.toLowerCase() + "<br />");

document.write(txt.toUpperCase());
```

4- The match() method

```
var str="Hello world!";

document.write(str.match("world") + "<br />");

document.write(str.match("World") + "<br />");
```

5- Replace characters in a string - replace()

```
var str="Visit Microsoft!";

document.write(str.replace("Microsoft"," Schools"));
```

6- The indexOf() method: How to return the position of the first found occurrence of a specified value in a string.

```
var str="Hello world!";

document.write(str.indexOf("d") + "<br />");
```

```
        document.write(str.indexOf("world"));
```

The try...catch Statement

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

Syntax

```
Try
  {
  //Run some code here
  }
catch(err)
  {
  //Handle errors here
  }
```

Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Create a Date Object

The Date object is used to work with dates and times. Date objects are created with the Date() constructor. There are four ways of instantiating a date:

new Date() // current date and time

Once a Date object is created, a number of methods allow you to operate on it. Most methods allow you to get and set the year, month, day, hour, minute, second, and milliseconds of the object, using either local time or UTC (universal, or GMT) time.

All dates are calculated in milliseconds from 01 January, 1970 00:00:00 Universal Time (UTC) with a day containing 86,400,000 milliseconds.

Some examples of instantiating a date:

```
today = new Date()
d1 = new Date("October 13, 1975 11:13:00")
d2 = new Date(79,5,24)
d3 = new Date(79,5,24,11,33,0)
```

Set Dates

We can easily manipulate the date by using the methods available for the Date object. In the example below we set a Date object to a specific date (14th January 2010):

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
```

And in the following example we set a Date object to be 5 days into the future:

```
var myDate=new Date();
myDate.setDate(myDate.getDate()+5);
```

Note: If adding five days to a date shifts the month or year, the changes are handled automatically by the Date object itself!

Compare Two Dates

The Date object is also used to compare two dates. The following example compares today's date with the 14th January 2010:

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
var today = new Date();
if (myDate>today)
  {
  alert("Today is before 14th January 2010");
  }
else
  {
  alert("Today is after 14th January 2010");
  }
```

JavaScript Math Object

round()

How to use round().

```
document.write(Math.round(0.60) + "<br />");
```

random()
How to use random() to return a random number between 0 and 1.

```
//return a random number between 0 and 1

document.write(Math.random() + "<br />");

//return a random integer between 0 and 10

document.write(Math.floor(Math.random()*11));
```

max()
How to use max() to return the number with the highest value of two specified numbers.

```
document.write(Math.max(0,150,30,20,38) + "<br />");

document.write(Math.max(-5,10) + "<br />");
```

min()
How to use min() to return the number with the lowest value of two specified numbers.

```
document.write(Math.min(-5,-10) + "<br />");
document.write(Math.min(1.5,2.5));
```

Method of object

Function	Description	Returned Values
getDate()	Day of the month	1-31
getDay()	Day of the week (integer)	0-6
getFullYear()	Year (full four digit)	1900+
getHours()	Hour of the day (integer)	0-23
getMilliseconds()	Milliseconds (since last second)	0-999
getMinutes()	Minutes (since last hour)	0-59
getMonth()	Month	0-11
getSeconds()	Seconds (since last minute)	0-59
getTime()	Number of milliseconds since 1	

	January 1970	
getTimezoneOffset()	Difference between local time and GMT in minutes	0-1439
getYear()	Year	0-99 for years between 1900-1999 Four digit for 2000+
parse()	Returns the number of milliseconds since midnight 1 January 1970 for a given date and time string passed to it.	
setDate()	Sets the day, given a number between 1-31	Date in milliseconds
setFullYear()	Sets the year, given a four digit number	Date in milliseconds
setHours()	Sets the hour, given a number between 0-23	Date in milliseconds
setMilliseconds()	Sets the milliseconds, given a number	Date in milliseconds
setMinutes()	Sets the minutes, given a number between 0-59	Date in milliseconds
setMonth()	Sets the month, given a number between 0-11	Date in milliseconds
setSeconds()	Sets the seconds,l given a number between 0-59	Date in milliseconds
setTime()	Sets the date, given the number of milliseconds since 1 January 1970	Date in milliseconds
setYear()	Sets the year, given either a two digit or four digit number	Date in milliseconds

toGMTString()
toUTCString()
	GMT date and time as a string	day dd mmm yyyy hh:mm:ss GMT

toLocaleString()	Local date and time as a string	Depends on operating system, locale, and browser
toString()	Local date and time as a string	Depends on operating system, locale, and browser
UTC()	Returns the number of milliseconds since 1 January 1970 for a given date in year, month, day (and optionally, hours, minutes, seconds, and milliseconds)	Date in milliseconds
valueOf()	Number of milliseconds since 1 January 1970	Date in milliseconds

JavaScript Form Validation
There's nothing more troublesome than receiving orders, guestbook entries, or other form submitted data that are incomplete in some way. You can avoid these headaches once and for all with JavaScript's amazing way to combat bad form data with a technique called "form validation". JavaScript **document.getElementById**

JavaScript getElementById
Have you ever tried to use JavaScript to do some form validation? Did you have any trouble using JavaScript to grab the value of your text field? There's an easy way to access any HTML element, and it's through the use of *id* attributes and the *getElementById* function.
If you want to quickly access the value of an HTML input give it an *id* to make your life a lot easier. This small script below will check to see if there is any text in the text field "myText". The argument that *getElementById* requires is the *id* of the HTML element you wish to utilize.

```
<script type="text/javascript">
function notEmpty(){
	var myTextField = document.getElementById('myText');
	if(myTextField.value != "")
		alert("You entered: " + myTextField.value)
	else
		alert("Would you please enter some text?")
}
</script>
<input type='text' id='myText' />
<input type='button' onclick='notEmpty()' value='Form Checker' />
```

The `innerHTML` property can be used to modify your document's HTML on the fly.
When you use `innerHTML`, you can change the page's content without refreshing the page. This can make your website feel quicker and more responsive to user input.
The `innerHTML` property is used along with `getElementById` within your JavaScript code to refer to an HTML element and change its contents.
The `innerHTML` property is not actually part of the official DOM specification. Despite this, it is supported in all major browsers, and has had widespread use across the web for many years. DOM, which stands for Document Object Model, is the hierarchy that you can use to access and manipulate HTML objects from within your JavaScript.

The `innerHTML` Syntax

The syntax for using `innerHTML` goes like this:

```
document.getElementById('{ID of
element}').innerHTML = '{content}';
```

In this syntax example, `{ID of element}` is the ID of an HTML element and `{content}` is the new content to go into the element.

Basic `innerHTML` Example

Here's a basic example to demonstrate how `innerHTML` works.
This code includes two functions and two buttons. Each function displays a different message and each button triggers a different function.
In the functions, the `getElementById` refers to the HTML element by using its ID. We give the HTML element an ID of "myText" using `id="myText"`.
So in the first function for example, you can see that `document.getElementById('myText').innerHTML = 'Thanks!';` is setting the innerHTML of the "myText" element to "Thanks!".
Code:

```
<script type="text/javascript">
function Msg1(){
  document.getElementById('myText').innerHTML = 'Thanks!';
}
function Msg2(){
  document.getElementById('myText').innerHTML = 'Try message 1
again...';
}
</script>
<input type="button" onclick="Msg1()" value="Show Message 1" />
<input type="button"  onclick="Msg2()" value="Show Message 2" />
<p id="myText"></p>
```

Result:
Thanks!

Example 2: `innerHTML` With User Input

Here's an example of using `innerHTML` with a text field. Here, we display whatever the user has entered into the input field.
Code:

```
<script type="text/javascript">
function showMsg(){
  var userInput = document.getElementById('userInput').value;
  document.getElementById('userMsg').innerHTML = userInput;
}
</script>
```

```
<input type="input" maxlength="40" id="userInput"
  onkeyup="showMsg()" value="Enter text here..." />
<p id="userMsg"></p>
```

Result:

Example 3: Formatting with `getElementById`

In this example, we use the `getElementById` property to detect the color that the user has selected. We can then use `style.color` to apply the new color. In both cases, we refer to the HTML elements by their ID (using `getElementById`.)

Code:

```
<script type="text/javascript">
function changeColor(){
  var newColor = document.getElementById('colorPicker').value;
        document.getElementById('colorMsg').style.color = newColor;
}
</script>
<p id="colorMsg">Choose a color...</p>
<select id="colorPicker" onchange="JavaScript:changeColor()">
<option value="#000000">Black</option>
<option value="#000099">Blue</option>
<option value="#990000">Red</option>
<option value="#009900">Green</option>
</select>
```

Result:

Choose a color...

String Search Function

This string function takes a regular expression and then examines that string to see if there are any matches for that expression. If there is a match , it will return the position in the string where the match was found. If there isn't a match, it will return -1. We won't be going into great depth about regular expressions, but we will show you how to search for words in a string.

Search Function Regular Expression

The most important thing to remember when creating a regular expression is that it must be surrounded with slashes ***/regular expression/***. With that knowledge let's search a string to see if a common name "Alex" is inside it.

```
<script type="text/javascript">
var myRegExp = /Alex/;
var string1 = "Today John went to the store and talked with Alex.";
var matchPos1 = string1.search(myRegExp);
```

```
if(matchPos1 != -1)
        document.write("There was a match at position " +
matchPos1);
else
        document.write("There was no match in the first string");
</script>
```

The quickest way to check if an input's string value is all numbers is to use a regular expression /^[0-9]+$/ that will only match if the string is all numbers and is at least one character long.

JavaScript Code:

```
// If the element's string matches the regular expression it is all
numbers.
        var numericExpression = /^[0-9]+$/;
```

What we're doing here is using JavaScript existing framework to have it do all the hard work for us. Inside each string is a function called match that you can use to see if the string matches a certain regular expression. We accessed this function like so: elem.value.match(expressionhere).
We wanted to see if the input's string was all numbers so we made a regular expression to check for numbers [0-9] and stored it as numericExpression.
We then used the match function with our regular expression. If it is numeric then match will return true, making our if statement pass the test and our function isNumeric will also return true. However, if the expression fails because there is a letter or other character in our input's string then we'll display our helperMsg and return false.

```
        if(elem.value.match(numericExpression)){
                return true;
         else    return false;
```

<u>charAt</u>

`charAt()` gives you the character at a certain position. For instance, when you do

```
var b = 'I am a JavaScript hacker.'
document.write(b.charAt(5))
```

<u>split</u>

split() does not work in Netscape 2 and Explorer 3.
split() is a specialized method that you need sometimes. It allows you to split a string at the places of a certain character. You must put the result in an array, not in a simple variable. Let's split b on the spaces.

```
var b = 'I am a JavaScript hacker.'
var temp = new Array();
temp = b.split(' ');
```

Now the string has been split into 5 strings that are placed in the array temp. The **spaces themselves are gone.**

```
temp[0] = 'I';
temp[1] = 'am';
temp[2] = 'a';
temp[3] = 'JavaScript';
temp[4] = 'hacker.';
```

Changing HTML with innerHTML

You can also insert HTML into your elements in the exact same way. Let's say we didn't like the text that was displayed in our paragraph and wanted to updated it with some color. The following code will take the old black text and make it bright white. The only thing we're doing different here is inserting the html element span to change the color.

JavaScript Code:

```
	var oldHTML =
document.getElementById('para').innerHTML;
```

Form Validation - Checking for All Letters

This function will be identical to isNumeric except for the change to the regular expression we use inside the match function. Instead of checking for numbers we will want to check for all letters.

If we wanted to see if a string contained only letters we need to specify an expression that allows for both lowercase and uppercase letters: /^[a-zA-Z]+$/ .

JavaScript Code:

```
// If the element's string matches the regular expression it is all letters
function isAlphabet(elem, helperMsg){
	var alphaExp = /^[a-zA-Z]+$/;
	if(elem.value.match(alphaExp)){
		return true;
	}else{
		alert(helperMsg);
		elem.focus();
		return false;
	}
}
```

Form Validation - Restricting the Length

Being able to restrict the number of characters a user can enter into a field is one of the best ways to prevent bad data. For example, if you know that the zip code field should only be 5 numbers you know that 2 numbers is not sufficient.
Below we have created a lengthRestriction function that takes a text field and two numbers. The first number is the minimum number of characters and the second is the maximum number of a characters the input can be. If you just want to specify an exact number then send the same number for both minimum and maximum.
JavaScript Code:

```
function lengthRestriction(elem, min, max){
	var uInput = elem.value;
	if(uInput.length >= min && uInput.length <= max){
		return true;
	}else{
		alert("Please enter between " +min+ " and " +max+ "
characters");
		elem.focus();
		return false;
	}
}
```

Form Validation - Email Validation

And for our grand finale we will be showing you how to check to see if a user's email address is valid. Every email is made up for 5 parts:
A combination of letters, numbers, periods, hyphens, plus signs, and/or underscores
The at symbol @
A combination of letters, numbers, hyphens, and/or periods
A period
The top level domain (com, net, org, us, gov, ...)
Valid Examples:
bobby.jo@filltank.net
jack+jill@hill.com
the-stand@steven.king.com
Invalid Examples:
@deleted.net - no characters before the @
free!dom@bravehe.art - invalid character !
shoes@need_shining.com - underscores are not allowed in the domain name
The regular expression to check for all of this is a little overkill and beyond the scope of this tutorial to explain thoroughly. However, test it out and you'll see that it gets the job done

External JavaScript File
You can place all your scripts into an external file (with a .js extension), then link to that file from within your HTML document. This is handy if you need to use the same scripts across multiple HTML pages, or a whole website.
To link to an external JavaScript file, you add a `src` attribute to your HTML `script` tag and specify the location of the external JavaScript file.

<u>Linking to an external JavaScript file</u>
<script type="text/javascript" src="external_javascript.js"></script>
Contents of your external JavaScript file
The code in **your .js file** should be no different to the code you would normally have placed in between the script tags. But remember, you don't need to create script tag again - it is already on the HTML page calling the external file!

In the previous lesson, we used an *event handler* to trigger off a call to our function. There are 18 event handlers that you can use to link your HTML elements to a piece of JavaScript.
When you write a JavaScript function, you will need to determine when it will run. Often, this will be when a user does something like click or hover over something, submit a form, double clicks on something etc.
These are examples of *events*.
Using JavaScript, you can respond to an event using event handlers. You can attach an event handler to the HTML element for which you want to respond to when a specific event occurs.
For example, you could attach JavaScript's onMouseover event handler to a button and specify some JavaScript to run whenever this event occurs against that button.
The HTML 4 specification refers to these as intrinsic events and defines 18 as listed below:

Event Handler	**Event that it handles**
onBlur	User has left the focus of the object. For example, they clicked away from a text field that was previously selected.
onChange	User has changed the object, then attempts to leave that field (i.e. clicks elsewhere).
onClick	User clicked on the object.

Event Handler	**Event that it handles**
onDblClick	User clicked twice on the object.
onFocus	User brought the focus to the object (i.e. clicked on it/tabbed to it)
onKeydown	A key was pressed over an element.
onKeyup	A key was released over an element.
onKeypress	A key was pressed over an element then released.
onLoad	The object has loaded.
onMousedown	The cursor moved over the object and mouse/pointing device was pressed down.
onMouseup	The mouse/pointing device was released after being pressed down.
onMouseover	The cursor moved over the object (i.e. user hovers the mouse over the object).
onMousemove	The cursor moved while hovering over an object.
onMouseout	The cursor moved off the object
onReset	User has reset a form.
onSelect	User selected some or all of the contents of the object. For example, the user selected some text within a text field.
onSubmit	User submitted a form.
onUnload	User left the window (i.e. user closes the browser window).

The events in the above table provide you with many opportunities to trigger some JavaScript from within your HTML code.
I encourage you to bookmark this page as a reference - later on you may need a reminder of which events you can use when solving a particular coding issue.
Sometimes, you may need to call some JavaSript from within a link. Normally, when you click a link, the browser loads a new page (or refreshes the same page).
This might not always be desirable. For example, you might only want to dynamically update a form field when the user clicks a link.

JavaScript "On Double Click"

You could just have easily used another event to trigger the same JavaScript. For example, you could run JavaScript only when the double

clicks the HTML element. We can do this using the `onDblClick` event handler.
Code:
```
<input type="button" onDblClick="alert('Hey,
remember to tell your friends about
Quackit.com!');" value="Double Click Me!" />
```

To prevent the load from refreshing, you could use the JavaScript void() function and pass a parameter of 0 (zero).

Example of void(0):

We have a link that should only do something (i.e. display a message) upon two clicks (i.e. a double click). If you click once, nothing should happen. We can specify the double click code by using JavaScript's "ondblclick" method. To prevent the page reloading upon a single click, we can use "JavaScript:void(0);" within the anchor link.
Code:
```
<a href="JavaScript:void(0);"
ondblclick="alert('Well done!')">Double Click
Me!</a>
```
Result:

Double Click Me!

Same Example, but without void(0):

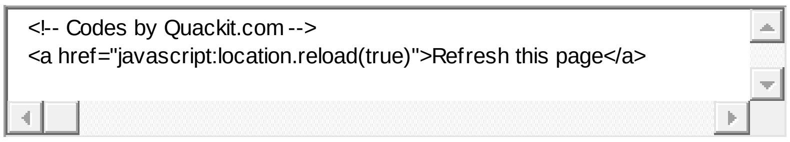
Look at what would happen if we didn't use "JavaScript:void(0);" within the anchor link...
Code:
```
<a href="" ondblclick="alert('Well
done!')">Double Click Me!</a>
```
Result:

Double Click Me!

Did you notice the page refresh as soon you clicked the link. Even if you double clicked and triggered the "ondbclick" event, the page still reloads!
Note: Depending on your browser, your browser might have redirected you to the "/javascript/tutorial/" index page. Either way, JavaScript's "void()" method will prevent this from happening.
Void(0) can become useful when you need to call another function that may have otherwise resulted in an unwanted page refresh.
Refresh code In JavaScript, you refresh the page using `location.reload`.
This code can be called automatically upon an event or simply when the user clicks on a link.

Example JavaScript Refresh code

Typing this code:



```
<!-- Codes by Quackit.com -->
<a href="javascript:location.reload(true)">Refresh this page</a>
```

You can use JavaScript to automatically open print dialogue box so that users can print the page. Although most users know they can do something like "File > Print", a "Print this page" option can be nice, and may even encourage users to print the page. Also, this code can be triggered automatically upon loading a printer friendly version.

Creating a "Print this page"

The following code creates a hyperlink and uses the Javascript print function to print the current page:

```
<a href="JavaScript:window.print();">Print this
page</a>
```

Open a new window in javascript

One of the more common requests I see is how to open a new Javascript window with a certain feature/argument. The Window.open method has been available since Netscape 2.0 (Javascript 1.0), but additional window decoration features have been added to the syntax since then.

Syntax

window.open([URL], [Window Name], [Feature List], [Replace]);

where:

[**URL**] **Optional.** Specifies the URL of the new window. If no URL is given, the window opens with no page loaded.

[**Window Name**] **Optional.** Specifies a name for the window. This name can be used to reference the window as a destination for a hyperlink using the HTML TARGET attribute.

[**Features**] **Optional.** Comma separated strings specifying the features added to the new window. Typical values are booleans (TRUE/FALSE.) Syntax is of the form

[**feature**] "=" [**value**] ("," [**feature**] "=" [**value**])*

[**Replace**] **Optional.** Boolean value specifying whether the new window replaces the current window in the browser's history.

window.open('dummydoc.htm',   'height=480,width=640');

write to new window

How do I write script-generated content to another window?
To write script-generated content to another window, use the method `winRef.document.write()` or `winRef.document.writeln()`, where `winRef` is the window reference, as returned by the `window.open()` method.

To make sure that your script's output actually shows up in the other window, use `winRef.document.close()` after writing the content.
As an example, consider the following function that opens a new window with the title **Console** and writes the specified `content` to the new window.

In the above example, you might notice that after you write something to the console *several times*, the console window will allow you to navigate back and forth in the output's history. This is not always a desired feature. If you would like to output the new content without creating a new *history entry*, add the following operator after opening the window (and before the first write):
docRef = top.*winRef*.document.open("text/html","replace");
Here `winRef` is the window reference returned by the `window.open()` method, and `docRef` is a global variable in which the script stores the reference to your new document.

```
<HTML>
<HEAD>
<TITLE>Writing to Subwindow</TITLE>
<SCRIPT LANGUAGE="JavaScript">
var newWindow
function makeNewWindow() {
   newWindow = window.open("","","status,height=200,width=300")
}
function subWrite() {
   if (newWindow.closed) {
      makeNewWindow()
   }
   newWindow.focus()
   var newContent = "<HTML><HEAD><TITLE>A New Doc</TITLE></HEAD>"
   newContent += "<BODY BGCOLOR='coral'><H1>This document is brand new.</
H1>"
   newContent += "</BODY></HTML>"
   newWindow.document.write(newContent)
   newWindow.document.close() // close layout stream
}
</SCRIPT>
</HEAD>
<BODY onLoad="makeNewWindow()">
```

```
<FORM>
<INPUT TYPE="button" VALUE="Write to Subwindow" onClick="subWrite()">
</FORM>
</BODY>
</HTML>
```

Moving and resizing windows

When you have created a window, you can use Javascript to move it or resize it.

Moving windows

A window object has two methods for moving it: `moveTo` and `moveBy`. Both take two numbers as arguments.

The first function moves the window to the specified coordinates, measured in pixels from the top left corner of the screen. The first argument is the horizontal, or X, coordinat, and the second is the vertical, or Y, coordinate.

The second method changes the current coordinates by the given amounts, which can be negative to move the window left or up.

Not all browsers allow you to move a window. A browser with a tabbed interface has one window in each tab, and cannot move or resize them individually. A browser with MDI (e.g., Opera) can only move the windows inside the parent application's window, and only if the document window isn't maximized.

```
window.moveBy(-10,20);
```

Resizing windows

Similar to the methods for moving, there are two methods for resizing a window: `resizeTo` and `resizeBy`. Like for moving, they either set the size absolutely or modify the size relatively to the current size.

Most browsers' standard security setting will not let you resize a window to less than 100 pixels in any direction.

```
window.resizeBy(100,-100);
```

Closing windows

Closing a window is simple when it works. All windows have a `close` method, so you can attempt to close any window you have a referene to.

```
myWindow.close();
```

However, not all windows can be closed without asking the user for permission. This is to protect the user from accidentally losing his browsing history.

The cases where you can close a window without a warning are:

- If a window was opened using Javascript, then it can be closed again without a warning.
- In some browsers, if the browser history contains only the current page, then the window can be closed without warning.
- In some browsers, setting the "window.opener" property to any window object will make the browser believe that the window was opened with Javascript.

Checking whether window has been closed

Given a reference to a window, it is possible to see whether the window has been closed.

```
if (myWindow.closed) { /* do something, e.g., open it again */ }
```

Browsers will generally not allow access to another window's properties, if it contains a page from a different domain. In some browsers, this even includes the `closed` property.

Problems with opening windows

Opening windows is not as safe as it used to be. Pages opening unwanted windows with advertisements have made people distrust new windows. Many users have installed software that prevents unwanted windows from opening.

Such ***popup blockers*** work in many different ways, which makes it hard to say anything definite. Some examples:

- Proximitron, a rewriting web proxy, can add Javascript that changes the `window.open` function in different ways. One option is to have it always return the current window instead of a new one.
- Some popup blockers work at the operating system level, and automatically close all new windows created by a browser. The call to `window.open` succeedes, but the window disappears before the user can see it.
- Modern browsers have popup-blockers built in (e.g., Mozilla, Opera, or MyIE2). They can be set to disable all new windows, or to allow only those that result from a user action (clicking on a link or button). A blocked popup can give a Javascript security error, stopping the script.
- Some specialized browsers, e.g., PDA's and mobile phones, completely lack the ability to open new windows, and might have no `window.open` function at all. Another specialized browser is WebTV, which can only do full screen pages, and it ignores new windows with sizes below 400 by 300 pixels and new windows with no size specified.

In the face of such diverse and unpredictable opposition, anybody trying to open a window should be prepared for it to fail. And it can fail silently, visibly, or irrecoverably. Some are of the opinion that `window.open` is no longer safe enough to use for anything important.

Windows status

- window.status = "This message will display in the window status bar."

The window status bar will not always work when written to immediately when the page loads. Normally it is used when an event happens such as when the user clicks on something or moves the mouse over an item or area on the page. The following code will display a status bar message when the mouse is moved over the home button

- blur() - This function will take the focus away from the window. Calling it as follows will perform the function.
  - <script language="JavaScript">
  - blur()
  - </script>
- close() - This function will close the current window or the named window. Normally a form button is provided to allow the user to close the window and the function is called by the onClick action. Otherwise some other event may be used to close the window.

  window.close()

- focus() - This function will give the focus to the window.
- moveBy(x,y) - The window is moved the specified number of pixels in the X and Y direction.
- moveTo(x,y) - The window is moved to the specified X and Y pixel location in the browser.
- open("URLname","Windowname",["options"]) - A new window is opened with the name specified by the second parameter. The document specified by the first parameter is loaded. The options are not required, and may be set to values of yes or now as in the below example. Many options are set to default values depending on whether others are set. For example, if the width and height are specified, the resizable option is set as default to no. Also if the toolbar is off, the menubar is set to default of no. The options are:
  - alwaysRaised - If yes, the created window is raised.
  - directories - The value of yes or no determines if the window shows directory buttons or not.
  - height - Specifies the window height in pixels. This is set to an integer value, rather than yes or no.
  - left - Specifies the distance in pixels the new window will be placed from the left side of the screen. This is set to an integer value, rather than yes or no.
  - location - The value of yes or no determines if the window has a location displayed on the address line (or has an address line) or not.

- menubar - The value of yes or no determines if the window has a menubar or not.
- outerWidth - Specifies the outer window width in pixels. This is set to an integer value, rather than yes or no.
- outerHeight - Specifies the outer window height in pixels. This is set to an integer value, rather than yes or no.
- resizable - The value of yes or no determines if the window can be resized by the user or not
- status - The value of yes or no determines if the window has a status bar or not.
- scrollbars - The value of yes or no determines if the window has scroll bars or not.
- toolbar - The value of yes or no determines if the window has a toolbar or not.
- top - Specifies the distance in pixels the new window will be placed from the top of the screen. This is set to an integer value, rather than yes or no.
- width - Specifies the window width in pixels. This is set to an integer value, rather than yes or no.
- z-lock - If yes, the created window is in the background.

Example:

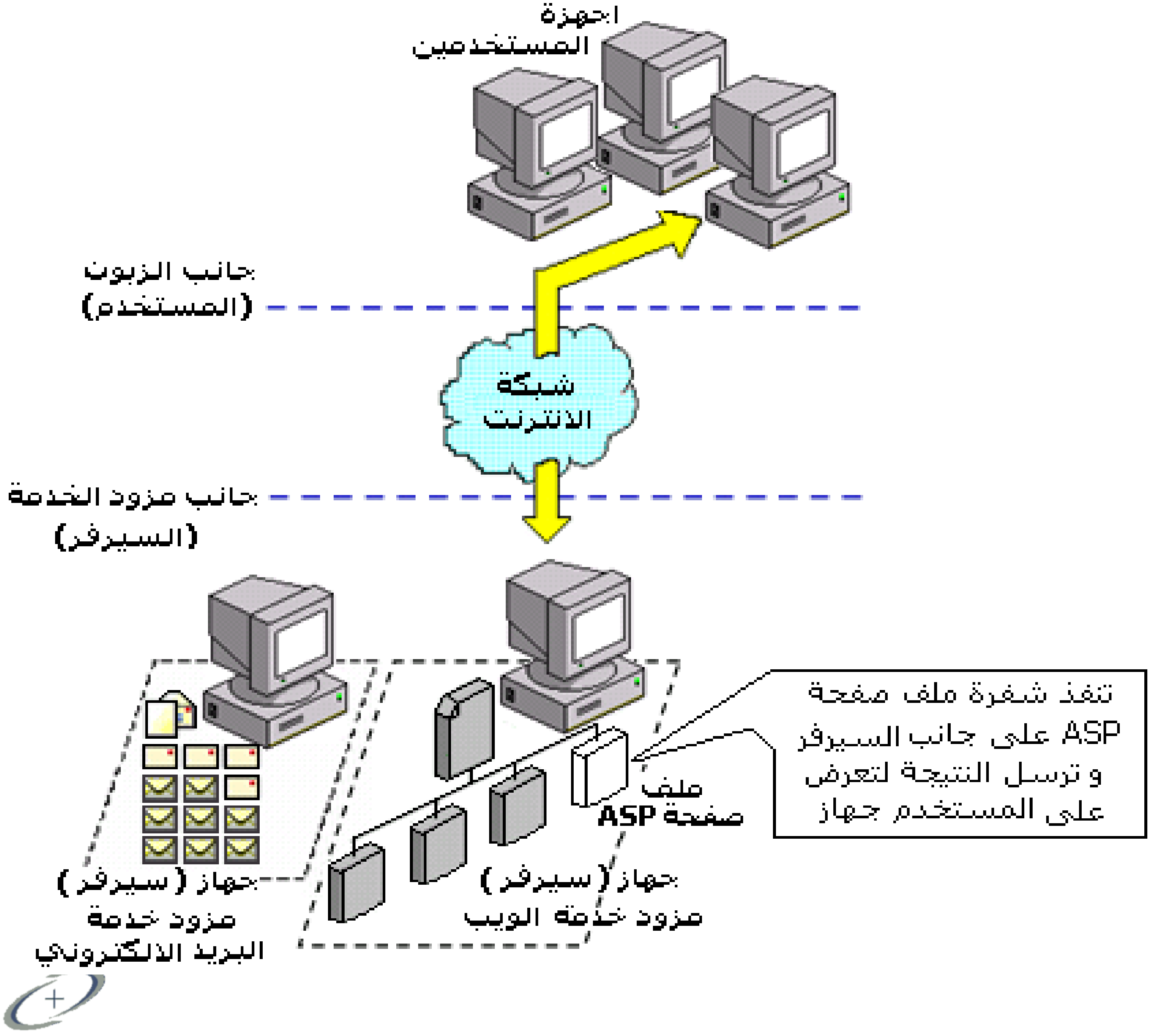
```
<script language="JavaScript">
open("javahere.html","test",
"toolbar=no,menubar=no,width=200,height=200,resizab
le=yes")
</script>
```

•

Active Server Pages

Introduction to the ASP

- "ASP is a technology developed by Microsoft . to create pages and Web applications are strong and dynamic. To create these pages you can add HTML or a scripting languages Scripting language such as VBScript or JavaScript, and you can also connect these pages to a database such as Access or SQL Server".
- ASP is abbreviated to (Active Server Pages)
- using ASP programming language web pages to do some events code.
- ASP file code implemented on the server side (Server Side), either the result be sent to the user's computer to display.


اجهزة المستخدمين
جانب الزبون (المستخدم)
شبكة الانترنت
جانب مزود الخدمة (السيرفر)
تنفذ شفرة ملف صفحة ASP على جانب السيرفر و ترسل النتيجة لتعرض على المستخدم جهاز
ملف صفحة ASP
جهاز (سيرفر) مزود خدمة البريد الالكتروني
جهاز (سيرفر) مزود خدمة الويب

ASP formats & Basic concepts

- You can not see ASP code
  By displaying the source code in the browser, you are the only one who can repeat to see ASP code when your design for active server pages, because script implemented on the server before you show the results sent to the browser.
  Files ASP naturally contain medals of HTML, and also contain script server, and the texts of the ASP writes always between those delineated by <% %>, and script server implemented on the server, and contains Script server on equations, and sentences, and transactions correct languages script that you use

The difference between the ASP and HTML?

1 - When a user requests the HTML file, the server sends the same HTML file to display the matching program.

2 - When a user requests the ASP file, IIS passes the request to the ASP engine, the ASP engine reads the ASP file line by line, and executes the script inside the file, in the end, the ASP file is due to the browser as an HTML file

- ASP pages different from HTML pages that address these pages are from the side of the device server not the user's computer, where it is installed in a server program features a dynamic link library called ASP.DLL, when a user requests a page extension asp service provider processes the orders existing ASP this and sends the result, and this result is a HTML commands to display in the browser on the user's computer. This method provides a degree of safety to these pages as they preserve the rights of programmed versions, where the user can not see ASP commands when viewing the page source, but sees HTML commands a result of the treatment.The question arises, do we need a host (server) to tackle the ASP pages that we create?!! Of course, not! All we need is a web server programs that address these pages, and the inauguration of this program we have made our server system and a user at the same time.

Web server software:

There are two suitable:

1 - PWS program, a shortcut Personal Web Server

2 - Program IIS which is an a shortcut for Internet Information Server

The choice of one of these programs on the operating system you have

Active Server Pages

IIS - Internet Information Server

- IIS is a set of Internet-based services for servers created by Microsoft for use with Microsoft Windows.
- IIS comes with Windows 2000, XP, Vista, and Windows 7. It is also available for Windows NT.
- IIS is easy to install and ideal for developing and testing web applications.

PWS - Personal Web Server

- PWS is for older Windows system like Windows 95, 98

How to Install IIS on Windows 7 and Windows Vista

Follow these steps to install IIS:

- Open the Control Panel from the Start menu
- Double-click Programs and Features
- Click "Turn Windows features on or off" (a link to the left)
- Select the check box for Internet Information Services (IIS), and click OK

How to Install IIS on Windows XP and Windows 2000

Follow these steps to install IIS:

- On the Start menu, click Settings and select Control Panel
- Double-click Add or Remove Programs
- Click Add/Remove Windows Components
- Click Internet Information Services (IIS)
- Click Details
- Select the check box for World Wide Web Service, and click OK
- In Windows Component selection, click Next to install IIS

After you have installed IIS or PWS follow these steps

- Look for a new folder called **Inetpub** on your hard drive
- Open the Inetpub folder, and find a folder named **wwwroot**
- Create a new folder, like "MyWeb", under wwwroot
- Write some ASP code and save the file as "test1.asp" in the new folder
- Make sure your Web server is running (see below)
- Open your browser "http://localhost/MyWeb/test1.asp", to view your first web page

- An ASP file normally contains HTML tags, just like an HTML file. However, an ASP file can also contain server scripts, surrounded by the delimiters <% and %>.
- Server scripts are executed on the server, and can contain any expressions, statements, procedures, or operators valid for the scripting language you prefer to use.

Script languages used with ASP:

There are two different types of (Script Languages) can be used to write the active code on the ASP pages are:

1 - VBScript (the default language permitted use in the pages of the ASP).

2 - JavaScript.

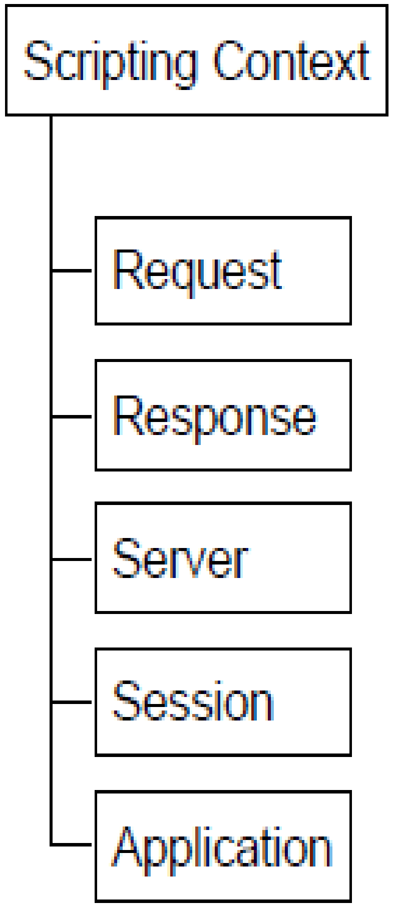
Vbscript

```
<%@ language= "VBscript" %>
```

Javascript

```
<%@ language= "Javascript" %>
```

Overview



Request:	To get information from the user
Response:	To send information to the user
Server:	To control the Internet Information Server
Session:	To store information about and change settings for the user's current Web-server session
Application:	To share application-level information and control settings for the lifetime of the application

Response object

- used the Response object to send information / results from a server to the user's computer
- Response Object , number of properties (Attributes) and approach (Methods) are as follows

-
  ```
  <%@ language= "VBscript" %>
  <html dir="rtl">
  <body>
    <%
      response.write ("مرحباً بكم في برمجة مواقع الويب")
    %>
  </body>
  </html>
  ```

- Definition of variables:
  Variables are used to store data. The following example will explain how variables are defined in the ASP

-
  ```
  <html dir="rtl">
  <head>
  </head>
  <body>
    <%
       dim name
       name="   برمجة مواقع الويب "
       response.write("<center><B> مرحباً بكم في " &name&
  "</b</center>")
    %>
  </body>
  </html>
  ```

- dim : used to define the variables.
  name: the name of the variable, and we set a value stored inside.
  In print command notes that we did not put the variable in quotation marks "" because we want to print the value of this variable and not print the same variable.
  When we want to print the value of the variable, and we want to include HTML elements with him must be separated between the two signal & As is shown in the top

```
<%
    dim fname(5),i
     fname(0)="مرحبا"
     fname(1)="بكم"
     fname(2)=" في"
     fname(3)="برمجة"
     fname(4)="مواقع الويب"
     for i=0 to 4
        response.write("<center><b>" &fname(i)& "</b><center>")
     next
  %>
```

Active Server Pages

- **What is localhost**

Let us first see, what we mean by a hostname. Whenever you connect to a

remote computer using it's URL, you are in effect calling it by its hostname.

For example, when you type in http://www.google.com/

you are really asking the network to connect to a computer named

www.google.com. It is called the "hostname" of that computer.

localhost is a special hostname. It always references your own machine. So what you just did, was to try to access a webpage on your own machine)which is what you wanted to do anyway.) For testing all your pages, you will need to use localhost as the hostname.

- **Response**
- The Response Object is responsible for sending data from the client to the server.

يمتلك الكائن Response Objectعدد من الخصائص (Attributes) و المنهاج (Methods) وهي كالتالي:-

مناهج الكائن Response Object

المنهج	الوصف
AddHeader	تضيف عنوان جديد للـ HTTP وقيمة جديدة.
Clear	تقوم بمسح جميع البيانات المخزنة داخل الـ Buffer
End	توقف عملية المعالجة للسكربات المخزنة داخل ال buffer، وتظهر الناتج الحالي.
Flush	يرسل المحتوي المخزن داخل ال buffer إلى برنامج التصفح.
Redirect	ترجع المستخدم مباشرة إلى رابط آخر.

Request object

used to deliver data sent by the user's computer to a server through HTTP request.

- **Required object**
- There are two ways to get form information:
- 1-The Request.QueryString command
- 2- The Request.Form command.

- **1- Request.QueryString**

The Request.QueryString command collects the values in a form as text. Information sent from a form with the GET method is visible to everybody (in the address field). Remember that the GET method limits the amount of information to send.

- Request.Querystring
  Method="get”
  action = “filename.asp"
- **Syntax**
- **Request.QueryString(variable)[(index)|.Count]**

- **2-Request.Form**
- Command Request.form, which is not much different from something Request.querystring, it is used to gather the values of the form with the use of method Post, which may not be visible to anyone, and it does not specify the amount of.
- **Syntax**
- **Request.Form(element)[(index)|.Count]**
- Request.Form

  Method="post"

  action = "filename.asp"

GET and POST

- One thing we ignored in our discussion about forms was that the METHOD by which the form is submitted may be one of the two: GET or POST.
- **When to use GET?**
- Any data that you pass via a GET can be retrieved in your script by using the Request.QueryString collection. GET may be used for small amounts of data
- – the reason being that, the data items are appended to the URL by your
- browser, and obviously, you cannot have an infinitely long URL (with the
- QueryString).
- **When to use POST?**
- Almost always. Stick to POST for your forms, and be sure to use the Request.Form collection to access them (and *not* the Request.QueryString
- collection.)

- Simpleform.asp
-
```
<html>
<body>
  <%
     request.querystring("fname“)
     request.querystring("lname“)
   %>
  <form method="get" action="simpleform.asp">
      fname : <input type="text" name="fname">
     </br >
  lname: <input type="text" name="lname">
      </br ></br >
      <input type="submit" value=“sent data” >
  </form>
</body>
</html>
```
- Run
- http://localhost/simpleform.asp?fname=shatha &lname=Habeeb

```
• <html dir="rtl">
  <body>
    <%
       request.form("username")
       request.form("pass")
    %>
    <form method="post" action="form2.asp">
       اسم المستخدم: <input type="text" name="fname"></br >
       كلمة المرور: <input type="text" name="lname">
       </br ></br >
       <input type="submit" value="إرسال البيانات">
    </form>
    <% ="اسم المستخدم:" &request.form("username") %></br>
    <% ="كلمة المرور:" &request.form("pass") %>
  </body>
  </html>
```

<u>Form1.asp</u>

```
<html dir="rtl">
    <body>
       <form method="get" action="form2.asp">
            fname: <input type="text" name="fname">
            </br >
            lname: <input type="text" name="lname">
            </br ></br >
            <input type="submit" value=“send data">
       </form>
    </body>
    </html>
```

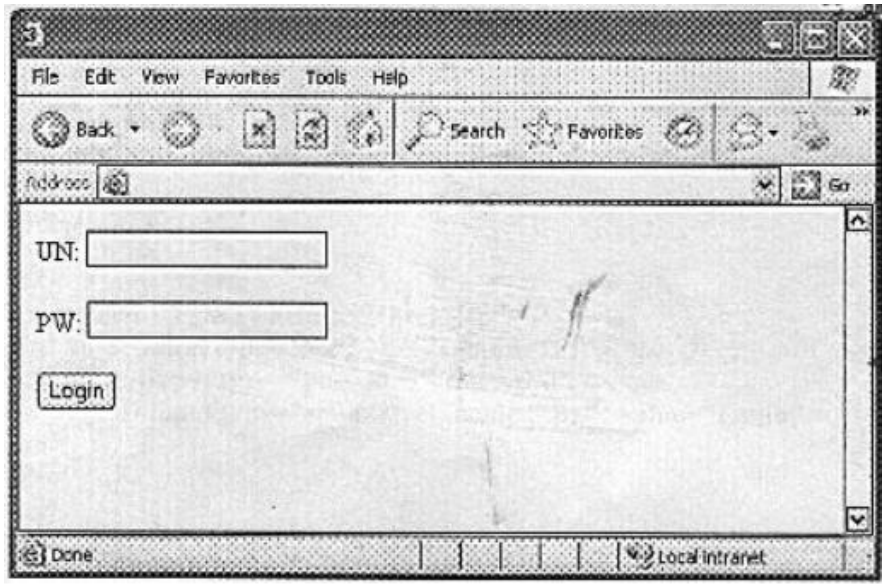
<u>Form2.asp</u>

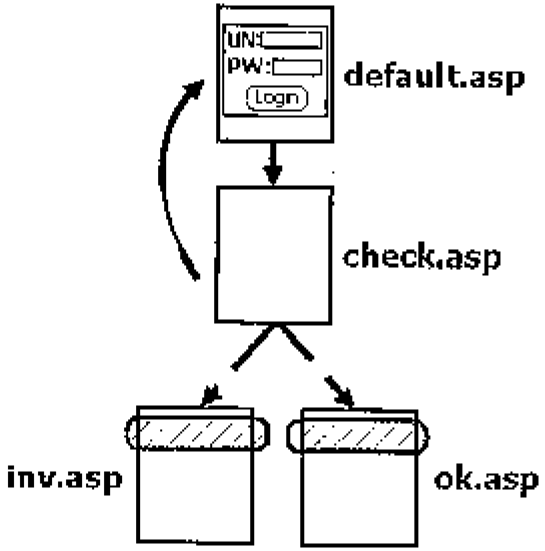
```
<html dir="rtl">
    <body>
      <%
         dim fname,lname
         request.querystring("fname“)
         request.querystring("lname“)
      %>
    <b>fname:<%="<B>"&fname&"</b>"%></br>lname:<%="<B>"&ln
    ame &"</B>"%>
    </body>
    </html>
```

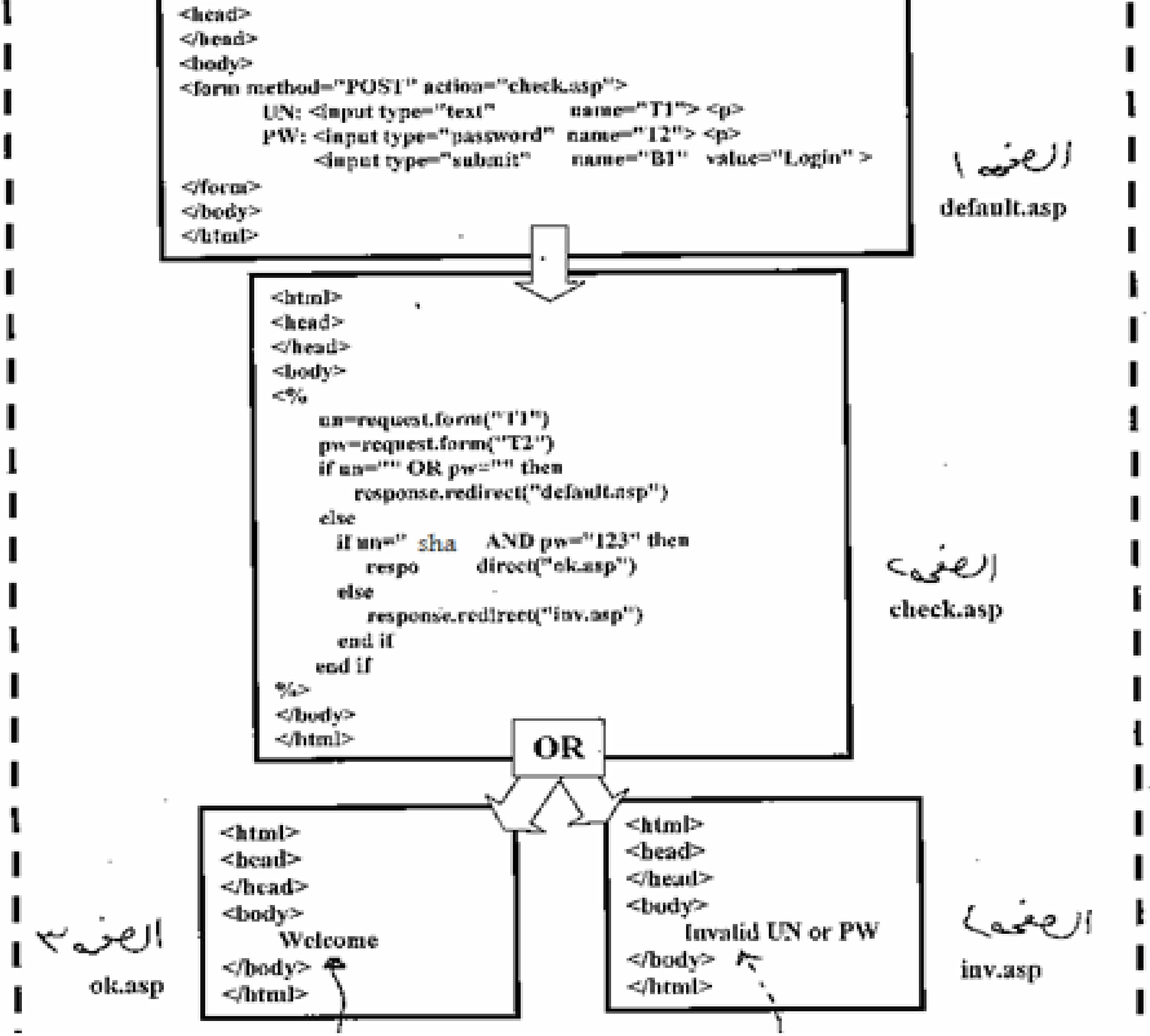
Login Application:

Typically, "Login Application" is used to allow the authorized users to

access private and secured areas, by using User-Name and Password strings.







```
<head>
</head>
<body>
<form method="POST" action="check.asp">
        UN: <input type="text"          name="T1"> <p>
        PW: <input type="password"  name="T2"> <p>
              <input type="submit"      name="B1"   value="Login" >
</form>
</body>
</html>
```

```
<html>
<head>
</head>
<body>
<%
      un=request.form("T1")
      pw=request.form("T2")
      if un="" OR pw="" then
          response.redirect("default.asp")
      else
        if un=" sha    AND pw="123" then
            respo       direct("ok.asp")
        else
            response.redirect("inv.asp")
        end if
      end if
%>
</body>
</html>
```

```
<html>
<head>
</head>
<body>
        Welcome
</body>
</html>
```

```
<html>
<head>
</head>
<body>
        Invalid UN or PW
</body>
</html>
```

- However, the internal pages ('ok. asp' and 'in .asp' pages) must be having "**Guarding Code**" to prevent any direct access to them. So if any user write the following address (http:// .... / ok. asp) to access the 'ok. Asp' page without coming from login-form (in other word to by- passing the checking code), the guarding-code must be prevent this unauthorized access and the process must be return to the 'default, asp' page The "Guarding Code" must be written inside all internal pages

- The Guarding-Code is work by check if the user access to this page by follows authorized-path or not. If the user access 'default.asp' page and enter correct username and password strings, then the 'check.asp' page must be not only redirect the process to the 'ok.asp' page but must register the username inside cookie file. So when access the 'ok.asp' page the process must be check cookie file, if cookie hold username then this means that the user access 'ok.asp' page by follow the authorized path, else if cookie hold nothing then this means that the user try to access to the 'ok.asp' page by by-passing checking code.

Cookie

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With ASP, you can both create and retrieve cookie values.

How to Create a Cookie?

- The "Response.Cookies" command is used to create cookies.
- <%
  Response.Cookies("firstname")="Alex"
- %>
- It is also possible to assign properties to a cookie, like setting a date when the cookie should expire:
- <%
  Response.Cookies("firstname")="Alex"
  Response.Cookies("firstname").Expires=#May 10,2012#
  %>

How to Retrieve a Cookie Value?

- The "Request.Cookies" command is used to retrieve a cookie value.

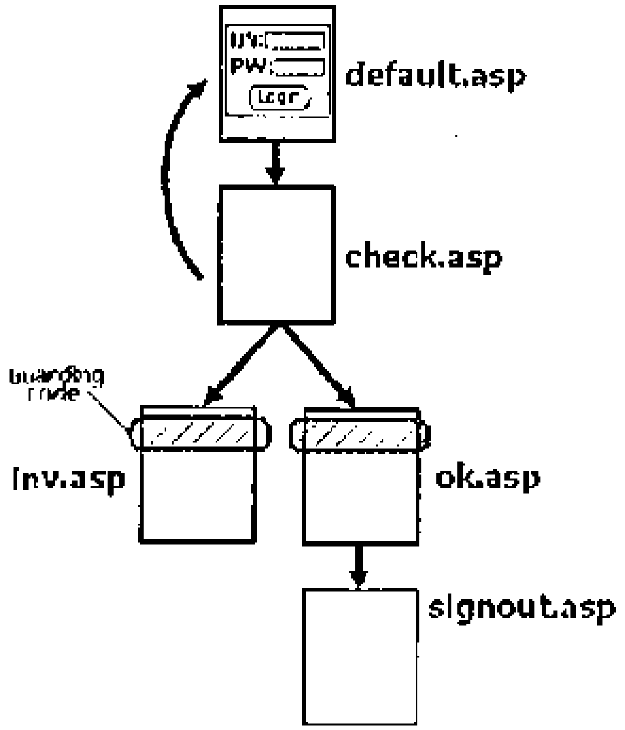
-
  ```
  <%
  fname=Request.Cookies("firstname")
  response.write("Firstname=" & fname)
  %>
  ```
- **Output:** Firstname=Alex

A Cookie with Keys

- If a cookie contains a collection of multiple values, we say that the cookie has Keys.
- In the example below, we will create a cookie collection named "user". The "user" cookie has Keys that contains information about a user:

-
  ```
  <%
  Response.Cookies("user")("firstname")="John"
  Response.Cookies("user")("lastname")="Smith"
  Response.Cookies("user")("country")="Norway"
  Response.Cookies("user")("age")="25"
  %>
  ```

- However, when user wants to leave 'ok.asp' page, so the user must be sign out this page, this means that the user must put nothing inside cookie file.



Read all Cookies
Look at the following code:

-
  ```
  <%
  Response.Cookies("firstname")="Alex"
  Response.Cookies("user")("firstname")="John"
  Response.Cookies("user")("lastname")="Smith"
  Response.Cookies("user")("country")="Norway"
  Response.Cookies("user")("age")="25"
  %>
  ```
- Assume that your server has sent all the cookies above to a user

Active Server Pages: Open, Read and Create files

FileSystemObject

- The FileSystemObject object is used to access the file system on a server. This object can manipulate files, folders, and directory paths. It is also possible to retrieve file system information with this object.

Syntax

- **Scripting.FileSystemObject**
- Set fs = CreateObject("**Scripting.FileSystemObject**")
- Set a = fs.CreateTextFile("c:\testfile.txt", True)
- a.WriteLine("This is a test.")
- a.Close

1- Open and Read content from a text file

```
<%
Set fs = CreateObject("Scripting.FileSystemObject")
Set wfile = fs.OpenTextFile("c:\Mydir\myfile.txt")
   filecontent = wfile.ReadAll
wfile.close
   Set wfile=nothing
   Set fs=nothing
response.write(filecontent)
   %>
```

ReadLine

```
<%
    Set fs = CreateObject("Scripting.FileSystemObject")
Set wfile = fs.OpenTextFile("c:\Mydir\myfile.txt")
    firstname = wfile.ReadLine
    lastname = wfile.ReadLine
    theage = wfile.ReadLine
wfile.close
    Set wfile=nothing
    Set fs=nothing
%>
Your first name is <% =firstname %><BR>
Your last name is <% =lastname %><BR>
Your are <% =thaage %> years old<BR>
```

AtEndOfStream

```
<%
Set fs = CreateObject("Scripting.FileSystemObject")
Set wfile = fs.OpenTextFile("c:\Mydir\myfile.txt")
counter=0
do while not wfile.AtEndOfStream
 counter=counter+1
 singleline=wfile.readline
 response.write (counter & singleline & "<br>")
loop
wfile.close
Set wfile=nothing
Set fs=nothing
%>
```

2- **Create and Write a text file**

Example 1: The basic code we need to create a file is very similar to that one we have used to open a file:

```
<%
   dim fs,f
   set fs=Server.CreateObject("Scripting.FileSystemObject")
   set f=fs.CreateTextFile("c:\test.txt",true)
   f.WriteLine("Hello World!")
   f.Close
   set f=nothing
   set fs=nothing
   %>
```

The #include Directive

- You can insert the content of one ASP file into another ASP file before the server executes it, with the #include directive.
- The #include directive is used to create functions, headers, footers, or elements that will be reused on multiple pages.

Here is a file called "mypage.asp"

-
  ```
  <html>
  <body>
  <h3>Words  :</h3>
  <p><!--#include file=“inclu.inc"--></p>
  <h3>The time is:</h3>
  <p><!--#include file="time.inc"--></p>
  </body>
  </html>
  ```
- <%
- response.write(date())
- %>

What is ADO?

ADO can be used to access databases from your web pages

- ADO is a Microsoft technology
- ADO stands for **A**ctiveX **D**ata **O**bjects
- ADO is automatically installed with Microsoft IIS
- ADO is a programming interface to access data in a database

Accessing a Database from an ASP Page

The common way to access a database from inside an ASP page is:

- Create an ADO connection to a database
- Open the database connection
- Create an ADO recordset
- Open the recordset
- Extract the data you need from the recordset
- Close the recordset
- Close the connection

Before a database can be accessed from a web page, a database connection has to be established.

- The ADO Connection Object is used to create an open connection to a data source. Through this connection, you can access and manipulate a database.
- If you want to access a database multiple times, you should establish a connection using the Connection object. You can also make a connection to a database by passing a onnection string via a Command or Recordset object. However, this type of connection is only good for one specific, single query.

```
set  objConnection
             =Server.CreateObject("ADODB.connection")
```

Methods

- Cancel
- Cancels an execution
- Close
- Closes a connection
- Execute
- Executes a query, statement, procedure or provider specific text
- Open
- Opens a connection
- OpenSchema
- Returns schema information from the provider about the data source

Create a DSN-less Database Connection

- The easiest way to connect to a database is to use a DSN-less connection. A DSN-less connection can be used against any Microsoft Access database on your web site.
- If you have a database called "northwind.mdb" located in a web directory like "c:/webdata/", you can connect to the database with the following ASP code:
-
```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
%>
```

- Note, from the example above, that you have to specify the Microsoft Access database driver (Provider) and the physical path to the database on your computer.

ADO Recordset

- The ADO Recordset object is used to hold a set of records from a database table. A Recordset object consist of records and columns (fields).
- To be able to read database data, the data must first be loaded into a recordset.
- Suppose we have a database named "Northwind", we can get access to the "Customers" table inside the database with the following lines:

set objRecordset=Server.CreateObject("ADODB.recordset")

- **<%**
  set conn=Server.CreateObject("ADODB.Connection")
  conn.Provider="Microsoft.Jet.OLEDB.4.0"
  conn.Open "c:/webdata/northwind.mdb"
  set rs=Server.CreateObject("ADODB.recordset")
  rs.Open "Customers", conn
  %>
- When you first open a Recordset, the current record pointer will point to the first record and the BOF and EOF properties are False. If there are no records, the BOF and EOF property are True.

Create an ADO SQL Recordset

We can also get access to the data in the "Customers" table using SQL:

-
  ```
  <%
  set conn =
  Server.CreateObject("ADODB.Connection")
  conn.Provider="Microsoft.Jet.OLEDB.4.0"
  conn.Open "c:/webdata/northwind.mdb"
  set rs=Server.CreateObject("ADODB.recordset")
  rs.Open "Select * from Customers", conn
  %>
  ```

Structured Query Language (SQL)

- The Structured Query Language (SQL) forms the backbone of all relational databases. This language offers a flexible interface for databases of all shapes and sizes and is used as the basis for all user and administrator interactions with the database.

Definition of a Relational Database

- A relational database is a collection of relations or two-dimensional tables.

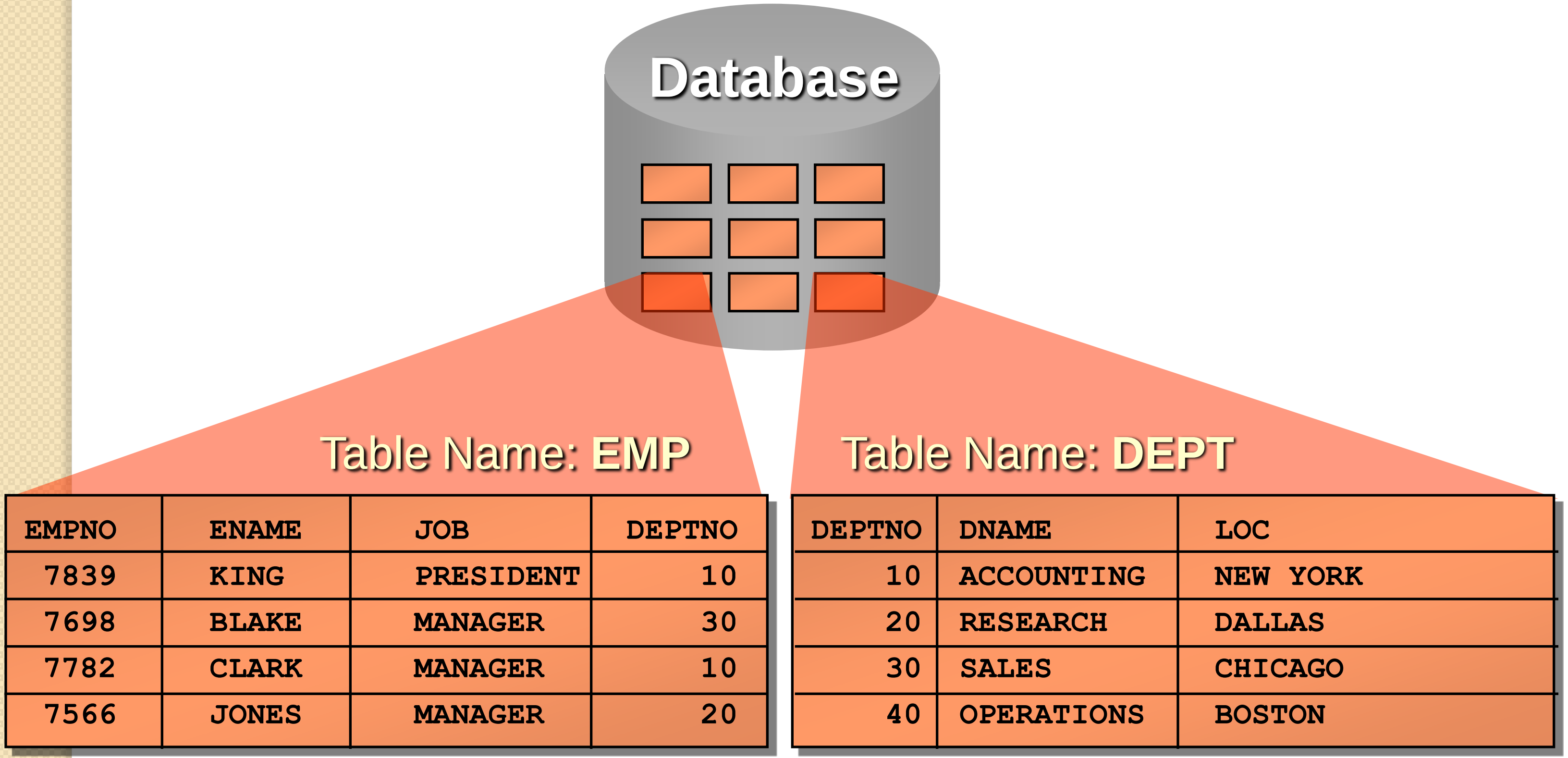
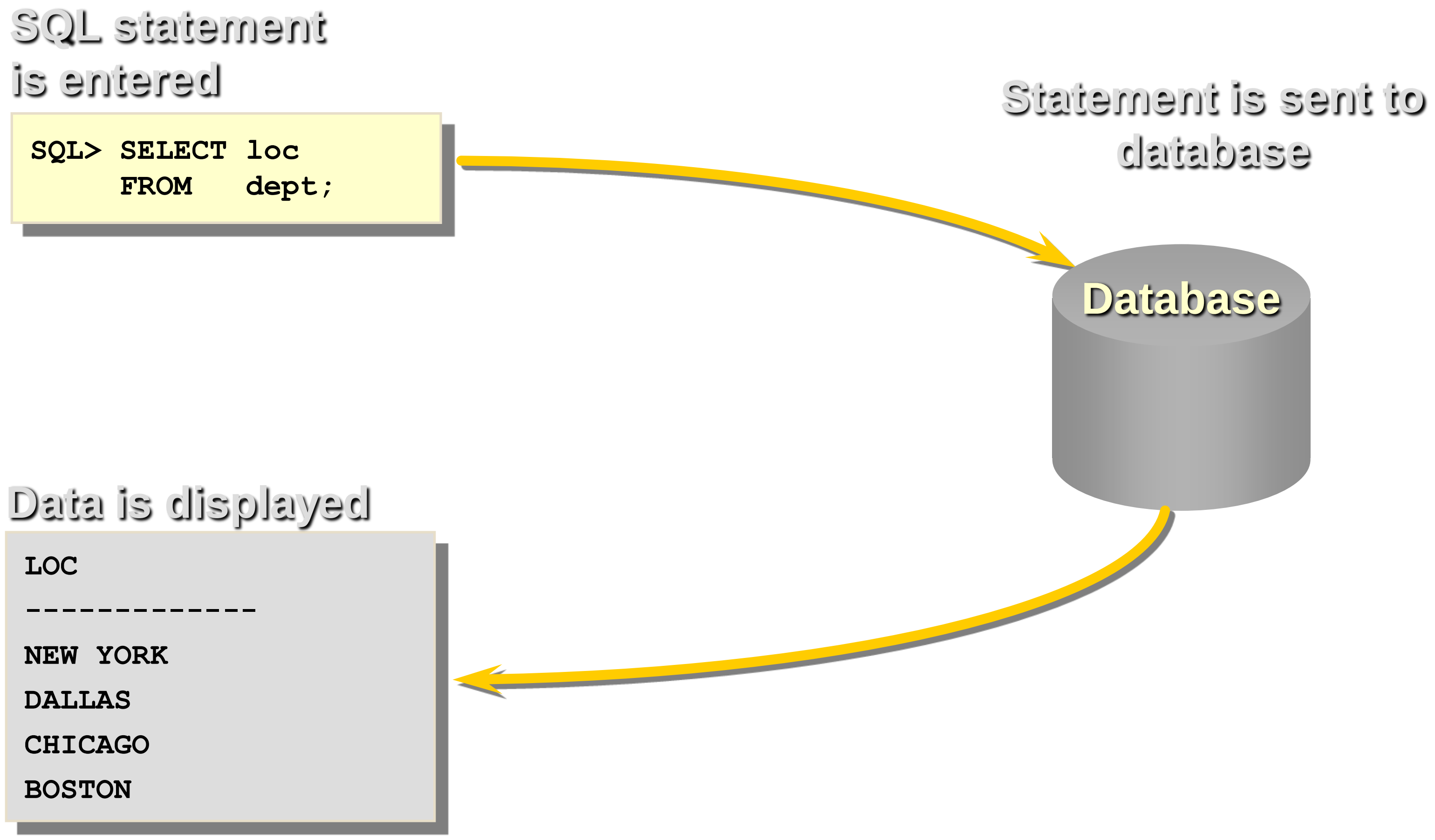


Table Name: **EMP**

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20

Table Name: **DEPT**

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON



```
SQL> SELECT loc
     FROM   dept;
```

```
LOC
-------------
NEW YORK
DALLAS
CHICAGO
BOSTON
```

Basic SELECT Statement

```
SELECT    [DISTINCT] {*, column [alias],...}
FROM      table;
```

- SELECT identifies *what* columns.
- FROM identifies *which* table.

Selecting All Columns

```
SQL> SELECT *
     FROM   dept;
```

```
   DEPTNO DNAME          LOC
--------- -------------- -------------
       10 ACCOUNTING     NEW YORK
       20 RESEARCH       DALLAS
       30 SALES          CHICAGO
       40 OPERATIONS     BOSTON
```

Selecting Specific Columns

```
SQL> SELECT deptno, loc
     FROM   dept;
```

```
   DEPTNO LOC
--------- -------------
       10 NEW YORK
       20 DALLAS
       30 CHICAGO
       40 BOSTON
```

Arithmetic Expressions

- Create expressions on NUMBER and DATE data by using arithmetic operators.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

Using Arithmetic Operators

```
SQL> SELECT ename, sal, sal+300
emp;      FROM
```

```
ENAME            SAL    SAL+300
---------- ---------- ----------
KING            5000       5300
BLAKE           2850       3150
CLARK           2450       2750
JONES           2975       3275
MARTIN          1250       1550
ALLEN           1600       1900
...
14 rows selected.
```

Limiting Rows Selected

- Restrict the rows returned by using the WHERE clause.

```
SELECT        [DISTINCT] {*| column [alias], ...}
FROM          table
[WHERE        condition(s)];
```

- The WHERE clause follows the FROM clause.

Using the WHERE Clause

```
SQL> SELECT ename, job, deptno
     FROM   emp
     WHERE  job='CLERK';
```

```
ENAME      JOB          DEPTNO
---------- --------- ---------
JAMES      CLERK            30
SMITH      CLERK            20
ADAMS      CLERK            20
MILLER     CLERK            10
```

Comparison Operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Using the Comparison Operators

```
SQL> SELECT ename, sal, comm
     FROM   emp
     WHERE  sal<=comm;
```

```
ENAME             SAL       COMM
---------- ---------- ----------
MARTIN           1250       1400
```

Other Comparison Operators

Operator	Meaning
BETWEEN ...AND...	Between two values (inclusive)
IN(list)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Using the BETWEEN Operator

- Use the BETWEEN operator to display rows based on a range of values.

```
SQL> SELECT   ename, sal
     FROM     emp
     WHERE    sal BETWEEN 1000 AND 1500;
```

```
ENAME             SAL
---------- ----------
MARTIN           1250
TURNER           1500
WARD             1250
ADAMS            1100
MILLER           1300
```

Lower limit

Higher limit

Using the IN Operator

- Use the IN operator to test for values in a list.

```
SQL> SELECT   empno, ename, sal, mgr
     FROM     emp
     WHERE    mgr IN (7902, 7566, 7788);
```

```
    EMPNO ENAME             SAL        MGR
--------- ---------- ---------- ----------
     7902 FORD             3000       7566
     7369 SMITH             800       7902
     7788 SCOTT            3000       7566
     7876 ADAMS            1100       7788
```

Using the LIKE Operator

- Use the LIKE operator to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers.
  - % denotes zero or many characters.
  - _ denotes one character.

```
SQL> SELECT   ename
     FROM     emp
     WHERE    ename LIKE 'S%';
```

Using the LIKE Operator

- You can combine pattern-matching characters.

```
SQL> SELECT   ename
     FROM     emp
     WHERE    ename LIKE '_A%';
```

```
ENAME
----------
MARTIN
JAMES
WARD
```

Using the IS NULL Operator

- Test for null values with the IS NULL operator.

```
SQL> SELECT   ename, mgr
     FROM     emp
     WHERE    mgr IS NULL;
```

```
ENAME             MGR
---------- ----------
KING
```

Using the NOT Operator

```
SQL> SELECT ename, job
     FROM   emp
     WHERE  job NOT IN ('CLERK','MANAGER','ANALYST');
```

```
ENAME      JOB
---------- ---------
KING       PRESIDENT
MARTIN     SALESMAN
ALLEN      SALESMAN
TURNER     SALESMAN
WARD       SALESMAN
```

ORDER BY Clause

- Sort rows with the ORDER BY clause
  - ASC: ascending order, default
  - DESC: descending order
- The ORDER BY clause comes last in the SELECT statement.

```
SQL> SELECT    ename, job, deptno, hiredate
     FROM      emp
     ORDER BY hiredate;
```

```
ENAME      JOB           DEPTNO HIREDATE
---------- --------- ---------- ---------
SMITH      CLERK             20 17-DEC-80
ALLEN      SALESMAN          30 20-FEB-81
...
14 rows selected.
```

The INSERT Statement

- Add new rows to a table by using the INSERT statement.

```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

- Only one row is inserted at a time with this syntax.

Inserting New Rows

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally list the columns in the INSERT clause.

```
SQL> INSERT INTO     dept (deptno, dname, loc)
     VALUES          (50, 'DEVELOPMENT', 'DETROIT');
1 row created.
```

- Enclose character and date values within single quotation marks.

The UPDATE Statement

- Modify existing rows with the UPDATE statement.

```
UPDATE          table
SET             column = value [, column = value, ...]
[WHERE          condition];
```

- Update more than one row at a time, if required.

Updating Rows in a Table

- Specific row or rows are modified when you specify the WHERE clause.

```
SQL> UPDATE   emp
     SET      deptno = 20
     WHERE    empno = 7782;
1 row updated.
```

- All rows in the table are modified if you omit the WHERE clause.

```
SQL> UPDATE   employee
     SET      deptno = 20;
14 rows updated.
```

The DELETE Statement

- You can remove existing rows from a table by using the DELETE statement.

```
DELETE [FROM]   table
[WHERE          condition];
```

Deleting Rows from a Table

- Specific rows are deleted when you specify the WHERE clause.

```
SQL> DELETE FROM     department
     WHERE           dname = 'DEVELOPMENT';
1 row deleted.
```

- All rows in the table are deleted if you omit the WHERE clause.

```
SQL> DELETE FROM     department;
4 rows deleted.
```