WEB TECHNOLOGY

Introduction to HTML

HTML stands for **HyperText Markup Language**. Developed by scientist Tim Berners-Lee in 1990, HTML is the "hidden" code that helps us communicate with others on the World Wide web (WWW). It is a markup language that web browsers use to interpret and compose text, images and other material into visual or audible web pages.

The definition of HTML is **HyperText Markup Language**.

- ***HyperText*** is the method by which you move around on the web — by clicking on special text called **hyperlinks** which bring you to the next page. The fact that it is *hyper* just means it is not linear — i.e. you can go to any place on the Internet whenever you want by clicking on links — there is no set order to do things in.
- ***Markup*** is what **HTML tags** do to the text inside them. They mark it as a certain type of text (*italicised* text, for example).
- **HTML** is a *Language*, as it has code-words and syntax like any other language.

HTML is written in the form of HTML elements consisting of *tags* enclosed in angle brackets (like <html>), within the web page content. HTML tags most commonly come in pairs like <h1>and </h1>, although some tags, known as *empty elements*, are unpaired, for example <img>. The first tag in a pair is the *start tag*, the second tag is the *end tag* (they are also called*opening tags* and *closing tags*). In between these tags web designers can add text, tags, comments and other types of text-based content.

HTML Document Structure

```
<HTML>
<HEAD>
<TITLE>The title of the document</TITLE>
<!-- Any other "head" information goes here-->
</HEAD>
<BODY>
<!-- Body of document goes here-->
</BODY>
</HTML>
```

The document is enclosed in <HTML> .... </HTML> tags, which tell the browser what it's reading; that is, that it's an HTML document and not something else.

Internally, the document consists of two sections; the head and its body. The head section contains various tags that provide meta-information about the file-its title, and its relationship to other documents on the web.

Document HEAD sections describe the document; the BODY section is the document. Under some circumstances, you can retrieve and examine the HEAD of a file without looking at the body.

Note the comment text. The tag <!-- --> is a comment (containing a single space); text inside it, <!-- like this,--> will not be displayed. It's often useful to scatter your documents with explanatory comments-notes about the document that you don't want your users to see.

The <HEAD> and <BODY> sections are, technically, optional. The HTML documents can safely exist without them - but it's good practice to use them.

Some Commonly Used HTML Tags

Tags	Use
(<HTML>. . . </HTML>)*	The entire HTML document

(<HEAD> . . . </HEAD>)*	The head, or prologue, of the HTML document
(<BODY> . . . </BODY>)*	All the other content in the HTML document
<TITLE> . . . </TITLE>	The title of the document
<H1> . . . </H1>	First-level heading large text size
<H2> . . . </H2>	Second-level heading
<H3> . . . </H3>	Third-level heading
<H4> . . . </H4>	Fourth-level heading
<H5> . . . </H5>	Fifth-level heading
<H6> . . . </H6>	Sixth-level heading small text size
<P> . . . (</P>)*	**Paragraph** Hitting a return in the HTML file will not make a new paragraph when the file is viewed. You need to use this tag to make a new paragraph.

	Line Break This tag will show a blank line.
<HR>	**Horizontal Rule** Creates a horizontal line on the page.
<!- . . . ->	**Comment** The comments you write in the middle will not show up on the page when viewed.
<A href=> . . . </A>	**Link** (A=Anchor) links the current HTML file to another file. Example: <A HREF="menu.html">Go back to Main Menu</A> This will display the file which is named in the quotes. The name of the link, which is the colored words you actually see goes between the first > and the second <. Here, the name of the link is Go back to the Main Menu Another example is : <A

	HREF="http://www.ilt.columbia.edu/">ILTNet</A> This link will take you to another page on the Internet. You can see the Internet address in the quotes.
<DL> <DT> <DD> </DL>	**Definition list** Put <DL> at the beginning, </DL> at the end, and <DT> for each :definition term" in the list. Use <DD> for each "definition" of each term. The definition will be indented. <DL> <DT>Item One <DD>Item One Definition </DL><="" td="">
<IMG SRC="image.gif">	**Inline Image** Put the name of the graphic (.gif or .jpg) in the quotes.
<B> . . . </B>	**Bold** Makes text bold
<I> . . . </I>	**Italic** Makes text italic
<font size="+3"...</font>	**Font Size** This tag is used to change the size of the font. It is better than using the header tag to make the font appear bigger.
<table> <TR> <TD> </TD> </TR> </Table>	"Table"=Starts a table. "TR" (Table Row) = Starts a row. "TD" (Table Data) = Starts a cell to enter data. "/TD" = Puts an End to data entry. "/TR" = Puts an end to a row. "/table" = Ends Table.

Advantages and Disadvantages of HTML

The **advantages** of HTML are:

- Simple and easy to understand.
- The HTML's syntax has always been looser and more forgiving.

The **disadvantages** of HTML are:

- Insufficient tags for some advance HTML coder.
- Hard to find exactly what are you looking for.
- Too much formatting built right into the documents, and by extension.
- Inconsistent formatting by browsers.
- Tags recognized by one browser may not be recognized by others.

Dynamic HTML (DHTML)

DHTML is a set of innovative features originally introduced in Microsoft Internet Explorer 4.0. By enabling authors to dynamically change the rendering and content of a Web page as the user interacts with it, DHTML enables authors to create visually compelling Web sites without the overhead of server-side programs or complicated sets of controls to achieve special effects.

With DHTML, you can easily add effects to your pages that previously were difficult to achieve. For example, you can:

- Hide content until a given time elapses or the user interacts with the page.
- Animate text and images in your document, independently moving each element from any starting point to any ending point, following a predetermined path or one chosen by the user.
- Embed a ticker that automatically refreshes its content with the latest news, stock quotes, or other data.
- Use a **form** to capture user input, and then instantly process and respond to that data.

Structure of DHTML Applications

DHTML applications are made up of the following pieces:

- One or more HTML pages.
- Visual Basic code that handles the events generated from the HTML pages.
- A run-time component that hosts the page in the Web browser or Web browser control.

- A project DLL that contains your Visual Basic code and is accessed by the run-time component generated automatically when you debug or compile.

There is a one-to-one relationship between the designers and the HTML pages in your project. For each page in your application, there is a page designer.

Advantages and Disadvantages of DHTML

The advantages are:

1. Fast and Zippy: - dHTML loads content on fly. Your whole page does not loads but only the content part that needs to be altered, so saving the crucial time for the users and giving the snazzy look to the website.
2. Plug-ins, we don't need them:- dHTML uses most of the features already present in the browsers, so there is no need to download any sort of Plug-ins.
3. Great Utility:- The dynamic features possessed by dHTML are helping web designers to create Web pages that posses compact looks, downloads fast, have graphic effects, provides greater functionality and can hold much more text or content all at the same time.

The disadvantages are:

1. Costly Editing tools: - although dHTML provides great functionality but the editors' available for that in market are pretty expensive. Examples of dHTML editors are Dreamweaver and Fusion.
2. Long and Complex coding: - dHTML coding is long and complex. Only the expert Javascript and HTML programmers can write them and edit them with good degree of functionality.
3. Browser Support problems: - dHTML suffers from browser support problems for different browsers. For example, a code written for Netscape might not work in Internet Explorer and Vice-Versa. The problem arises due to the different features of browsers.

XML

XML - Extensible Markup Language, is a markup language that you can use to create your own tags. It was created by the World Wide Web Consortium (W3C) to overcome the limitations of HTML, the Hypertext Markup Language that is the basis for all Web pages. Like HTML, XML is based on SGML -- Standard Generalized Markup Language. Although SGML has been used in the publishing industry for decades, its perceived complexity intimidated many people that otherwise might have used it (SGML also stands for "Sounds great, maybe later"). XML was designed with the Web in mind.

XML Structure

This page provides a description of XML structure including the document parts, the prologue, and provides a simple XML example document.

Document Parts

- Prolog
- Document Element (root element)

The Prologue

The prologue, equivalent to the header in HTML, may include the following:

- An XML declaration (optional) such as:
  ```
  <?xml version="1.0"?>
  ```
- A DTD or reference to one (optional). An example reference to an external DTD file:
  ```
  <!DOCTYPE LANGLIST SYSTEM "langlist.dtd">
  ```
- Processing instructions - An example processing instruction that causes style to be determined by a style sheet:
  ```
  <?xml-stylesheet type="text/css" href="xmlstyle.css"?>
  ```

An XML Document

Therefore a complete well formed XML document may look like:

```
<?xml version="1.0"?>
<LAND>
  <FOREST>
    <TREE>Oak</TREE>
    <TREE>Pine</TREE>
    <TREE>Maple</TREE>
  </FOREST>
  <MEADOW>
    <GRASS>Bluegrass</GRASS>
```

```
        <GRASS>Fescue</GRASS>
        <GRASS>Rye</GRASS>
      </MEADOW>
    </LAND>
```

Advantages and Disadvantages of XML

The **advantages** of XML are:

- It is much simpler compare to SGML
- XML documents can be Valid or Well Formed without a DTD (deducing the semantics from the structure of the document)
- Linking is very much simpler in XML than in SGML, but also much more intelligent than HTML
- No longer restricted a limited set of tags, you can create your own set of tags.
- XML is very powerful, yet easy to implement, not limited on data structures
- It is in text format, Human readable and easy to edit

The **disadvantages** of XML are:

- DTD's cannot BE located on the Internet
- There is no guarantee that all browser currently support XML, although now some browser now currently capable of interpreting XML, example are IE 5 and Netscape.

Introduction to JavaScript

JavaScript is a programming language that can be included on web pages to make them more interactive. You can use it to check or modify the contents of forms, change images, open new windows and write dynamic page content.

JavaScript has nothing to do with Java. JavaScript, originally nicknamed LiveWire and then LiveScript when it was created by Netscape, should in fact be called ECMAscript as it was renamed when Netscape passed it to the ECMA for standardisation.

JavaScript is a client side, interpreted, object oriented, high level scripting language, while Java is a client side, compiled, object oriented high level language.

Client side

Programs are passed to the computer that the browser is on, and that computer runs them. The alternative is server side, where the program is run on the server and only the results are passed to the computer that the browser is on. Examples of this would be PHP, Perl, ASP, JSP etc.

Interpreted

The program is passed as source code with all the programming language visible. It is then converted into machine code as it is being used. Compiled languages are converted into machine code first then passed around, so you never get to see the original programming language. Java is actually dual half compiled, meaning it is half compiled (to 'byte code') before it is passed, then executed in a virtual machine which converts it to fully compiled code just before use, in order to execute it on the computer's processor. Interpreted languages are generally less fussy about syntax and if you have made mistakes in a part they never use, the mistake usually will not cause you any problems.

Scripting

This is a little harder to define. Scripting languages are often used for performing repetitive tasks. Although they may be complete programming languages, they do not usually go into the depths of complex programs, such as thread and memory management. They may use another program to do the work and simply tell it what to do. They often do not create their own user interfaces, and instead will rely on the other programs to create an interface for them. This is quite accurate for JavaScript. We do not have to tell the browser exactly what to put on the screen for every pixel, we just tell it that we want it to change the document, and it does it. The browser will also take care of the memory management and thread management, leaving JavaScript free to get on with the things it wants to do.

High level

Written in words that are as close to english as possible. The contrast would be with assembly code, where each command can be directly translated into machine code.

Javascript: Advantages and Disadvantages

Javascript is one of the most simple, versatile and effective languages used to extend functionality in websites. Uses range from on screen visual effects to processing and calculating data on web pages with ease as well as extended functionality to websites using third party scripts among several other handy features, however it also possesses some negative effects that might make you want to think twice before implementing Javascript on your website. Let's look at some of its pros and cons.

Advantages

- **Javascript is executed on the client side -** This means that the code is executed on the user's processor instead of the web server thus saving bandwidth and strain on the web server.
- **Javascript is a relatively easy language -** The Javascript language is relatively easy to learn and comprises of syntax that is close to English. It uses the DOM model that provides plenty of prewritten functionality to the various objects on pages making it a breeze to develop a script to solve a custom purpose.
- **Javascript is relatively fast to the end user -** As the code is executed on the user's computer, results and processing is completed almost instantly depending on the task (tasks in javascript on web pages are usually simple so as to prevent being a memory hog) as it does not need to be processed in the site's web server and sent back to the user consuming local as well as server bandwidth.
- **Extended functionality to web pages -** Third party add-ons like Greasemonkey enable Javascript developers to write snippets of Javascript which can execute on desired web pages to extend its functionality. If you use a website and require a certain feature to be included, you can write it yourself and use an add-on like Greasemonkey to implement it on the web page.

Disadvantages

- **Security Issues -** Javascript snippets, once appended onto web pages execute on client servers immediately and therefore can also be used to exploit the user's system. While a certain restriction is set by modern web standards on browsers, malicious code can still be executed complying with the restrictions set.

- **Javascript rendering varies -** Different layout engines may render Javascript differently resulting in inconsistency in terms of functionality and interface. While the latest versions of javascript and rendering have been geared towards a universal standard, certain variations still exist.**Website Usability Consultants all over the world** make a living on these differences, but it enrages thousands of developers on a daily basis.

Structure of Java script

All JavaScript code should begin with the tag "<script language= "type/javascript">. Adding this line allows for JavaScript to be validated and thus less errors. In cases in which the browser does not support JavaScript the following lines are added to hide the code:

```
<script language= "type/javascript">
<!---document.write("JavaScript platform initiated!")//- - >
</script>
```

The above code can be inserted anywhere on a HTML code window to display the text "JavaScript platform initiated!" if the browser supports JavaScript.

Variables

- These are containers that are used to hold data and that can be referenced anywhere on the code. Variables thus allow for data reuse, simplifying the programming process. Variables can be created and initiated at the same time as shown below:

  var MyName = "David";

  The above variable is called "MyName" and holds the value "David." I can now refer to that variable anywhere on the code by simply calling "MyName."

Loops

- Looping in JavaScript is implemented when you do not want code to follow a straight, linear fashion. This is one of the flow control techniques that allows for certain blocks of code to be run when certain conditions are met. Loops can be implemented using either a single command or multiple commands. Loops form an important structure of any JavaScript that has repeating script blocks. Loops can either be "for" loops, "while" loops or "dowhile" loops.

Forms

- After creating forms in your average text editor, you can access and manipulate the text field contents of the forms using JavaScript. The contents of these fields can be displayed in a dialog box as a pop up using JavaScript. Form fields can also be updated dynamically using JavaScript by use of what is called an "OnClick" event handler that specifies the JavaScript code that executes when the user clicks on a link on the form. Form structures thus form an important element that allows for client-side updating of form fields.

Functions

- These are blocks of code that contain statements executed when a certain event or call is made to the function. These blocks of code are thus not executed when, say, a page loads. They require the user to trigger their execution. They thus form an important code structure within a page.

ASP.NET

ASP.NET presents a whole new approach to developing dynamic content for the Internet or your intranet. With ASP, the process was very linear. A page was requested, and your ASP code returned HTML tags either directly or indirectly through the Response object. You then received the visitor’s input through the Request object. With ASP.NET, the approach is much more objecting driven. The process is more like developing a standard Windows-based application instead of a Web page. With ASP.NET, you place controls such as Label controls and TextBox controls on your page. You then assign values to the properties of these controls, which allows you to affect how the controls are rendered in the visitor’s browser.
Your controls have methods that allow you to take some action against them. You know that the visitor has taken some action because events fire, which you can write code for.

Simple structure of ASP.NET page

The page simply displays some text to the visitor through their browser. Figure 1 displays the output of this page.

HelloWeb.aspx

Here is the entire code for this simple ASP.NET page:

```
<%@ Page Language=VB Debug=true %>
<script runat=server>
Sub Page_Load(ByVal Sender as Object, ByVal E as EventArgs)
lblMessage.Text = "Hello Web!"
End Sub
</script>
<html>
<head>
<title>Hello Web Sample Page</title>
</head>
<body
background="./bg.gif"
text="black"
link="darkred"
vlink="darkred"
alink="red"
leftmargin="40"
topmargin="30"
>
<form runat="server">
<font face="Tahoma">
<asp:Label
id="lblTitle"
BorderWidth="7px"
BorderStyle=9
Width="90%"
Font-Size="25pt"
Font-Name="Arial"
Text="Test Page"
runat="server"
```

```
/>
<br><br><br>
<asp:Label
id="lblMessage"
runat="Server"
Font-Bold="True"
/>
</font>
</form>
</body>
</html>
```

First, notice the general structure of the contents of the ASP.NET page. Like an ASP page, the page does contain standard HTML tags. But notice that the controls placed on this page are defined like some specialized HTML tags. This is very different from ASP, where you would have found Response.Write statements.

Also notice that all the code on the page is contained within a procedure. In ASP, you could see code anywhere and the code was simply contained within code tags. The first line on the page is called a compiler directive. This directive tells the compiler that the language you are using on this page is Visual Basic.NET. It also tells the compiler that you want the page to be in debug mode. The following directive will be discussed further later in this chapter:

<%@ Page Language=VB Debug=true %>

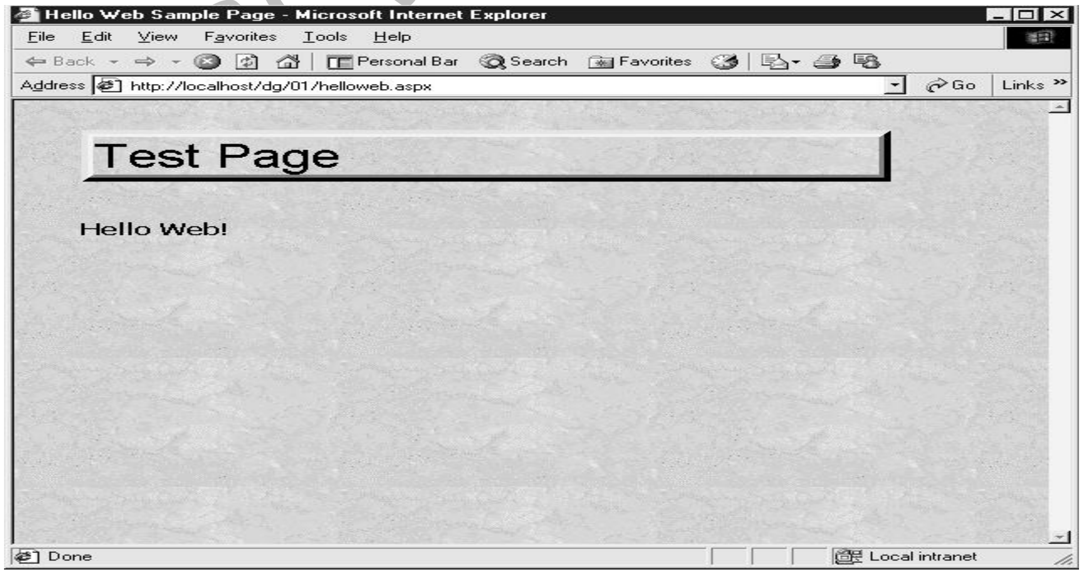
Hello Web Sample Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Personal Bar Search Favorites
Address http://localhost/dg/01/helloweb.aspx Go Links
Test Page
Hello Web!
Done
Local intranet

Figure 1 Simple sample page

Referemces

1. http://www.quackit.com/html/tags/
2. http://en.wikipedia.org/wiki/HTML
3. http://www.antipope.org/charlie/attic/webbook/ch2html/html-ch1b.html
4. http://reocities.com/siliconvalley/board/1250/htxsg/advandis.htm
5. http://www.columbia.edu/~sss31/html/html-tags.html
6. http://ezinearticles.com/?dHTML---Advantages-and-Disadvantages&id=144196
7. http://www.ibm.com/developerworks/xml/tutorials/xmlintro/section2.html
8. http://csharpcomputing.com/XMLTutorial/Lesson7.html
9. http://www.howtocreate.co.uk/tutorials/javascript/introduction
10. http://www.jscripters.com/javascript-advantages-and-disadvantages/
11. http://books.mcgraw-hill.com/downloads/products/0072192887/0072192887_ch01.pdf