UNIT – I

Introduction:

HTML stands for Hypertext Markup Language. It is used to display the document in the web browsers. HTML pages can be developed to be simple text or to be complex multimedia program containing sound, moving images and java applets. HTML is considered to be the global publishing format for Internet. It is not a programming language. HTML was developed by Tim Berners-Lee. HTML standards are created by a group of interested organizations called W3C (world wide web consortium). In HTML formatting is specified by using tags. A tag is a format name surrounded by angle brackets. End tags which switch a format off also contain a forward slash.

Points to be remembered for HTML tags:

- Tags are delimited by angled brackets.
- They are not case sensitive i.e., <head>, <HEAD> and <Head> is equivalent.
- If a browser not understand a tag it will usually ignore it.
- Some characters have to be replaced in the text by escape sequences.
- White spaces, tabs and newlines are ignored by the browser.

Structure of an HTML document:

All HTML documents follow the same basic structure. They have the root tag as <html>, which contains <head> tag and <body> tag. The head tag is used for control information by the browser and the body tag contains the actual user information that is to be displayed on the screen. The basic document is shown below.

```
<html>
    <head>
        <title> Basic HTML document </title>
    </head>
    <body>
        <h1> Welcome to the world of  Web Technologies</h1>
        <p> A sample HTML program written by Amer </p>
    </body>
</html>
```

Besides head and body tag, there are some other tags like title, which is a sub tag of head, that displays the information in the title bar of the browser. <h1> is used to display the line in its own format i.e., bold with some big font size. <p> is used to write the content in the form of paragraph.

Comments in HTML documents start with <! and end with >. Each comment can contain as many lines of text as you like. If comment is having more lines, then each line must start and end with -- and must not contain -- within its body.

```
<! -- this  is a comment line - -
   -- which can have more lines - ->
```

Basic HTML tags

1. Body tag :
   Body tag contain some attributes such as bgcolor, background etc. bgcolor is used for background color, which takes background color name or hexadecimal number and #FFFFFF and background attribute will take the path of the image which you can place as the background image in the browser.
   <body bgcolor="#F2F3F4" background= "c:\amer\imag1.gif">

2. Paragraph tag:
   Most text is part of a paragraph of information. Each paragraph is aligned to the left, right or center of the page by using an attribute called as align.
   <p align="left" | "right" | "center">

3. Heading tag:
   HTML is having six levels of heading that are commonly used. The largest heading tag is <h1> . The different levels of heading tag besides <h1> are <h2>, <h3>, <h4>, <h5> and <h6>. These heading tags also contain attribute called as align.
   <h1 align="left" | "right" | "center"> . . . . <h2>

4. hr tag:
   This tag places a horizontal line across the system. These lines are used to break the page. This tag also contains attribute i.e., width which draws the horizontal line with the screen size of the browser. This tag does not require an end tag.
   <hr width="50%">.

5. base font:
   This specify format for the basic text but not the headings.
   <basefont size="10">

6. font tag:
   This sets font size, color and relative values for a particular text.
   <font size="10" color="#f1f2f3">

7. bold tag:
   This tag is used for implement bold effect on the text
    …….

8. Italic tag:
   This implements italic effects on the text.
   …….

9. strong tag:
   This tag is used to always emphasized the text
   …….

10. tt tag:
    This tag is used to give typewriting effect on the text
    <tt>……..</tt>

11. sub and sup tag:
    These tags are used for subscript and superscript effects on the text.
    _{…….}
    ^{……..}

12. Break tag:
    This tag is used to the break the line and start from the next line.

13. &amp &lt &gt   &quot
    These are character escape sequence which are required if you want to display characters that HTML uses as control sequences.
        Example: < can be represented as &lt.

14. Anchor tag:
    This tag is used to link two HTML pages, this is represented by <a>
     some text
    href is an attribute which is used for giving the path of a file which you want to link.

Example 1: HTML code to implement common tags.

mypage.html

```
<html>
<head>  <! -- This page implements common html tags -->
<title> My Home page </title>
</head>
<body >
<h1 align="center"> GREEN FORT ENGINEERING COLLEGE</h1>
<h2 align="center"> Bandlaguda, Hyderabad</h2>
<basefont size=4>
<p> This college runs under the <tt>management</tt> of <font size=5> <b><i>&quot
Syed Hashim Education Society&quot &amp it is</i></b></font><br>
affiliated to <strong> JNTU</strong>
<hr size=5 width=80%>
<h3> <u>&ltSome common tags&gt</u> </h3><br>
```

 List

 Table

</body>

</html>

Lists:

One of the most effective ways of structuring a web site is to use lists. Lists provides straight forward index in the web site. HTML provides three types of list i.e., bulleted list, numbered list and a definition list. Lists can be easily embedded easily in another list to provide a complex but readable structures. The different tags used in lists are explained below.

 …..

The ordered(numbered) and unordered(bulleted) lists are each made up of sets of list items. This tag is used to write list items

<ul type=”disc” | “square” | ”circle” > …..

This tag is used for basic unordered list which uses a bullet in front of each tag, every thing between the tag is encapsulated within <li> tags.

<ol type=”1” | ”a” | “I” start=”n”>…..

This tag is used for unordered list which uses a number in front of each list item or it uses any element which is mentioned in the type attribute of the <ol> tag, start attribute is used for indicating the starting number of the list.

<dl>….. </dl>

This tag is used for the third category i.e., definition list, where numbers or bullet is not used in front of the list item, instead it uses definition for the items.

<dt>…..</dt>

This is a sub tag of the <dl> tag called as definition term, which is used for marking the items whose definition is provided in the next data definition.

<dd> ….</dd>

This is a sub tag of the <dd> tag, definition of the terms are enclosed within these tags. The definition may include any text or block.

Example 2: HTML code showing list tags.

<html>
<head>
<title> list page </title>

</head>

```
<body>
<h3> Course details </h3><br>
<ul type="disc">
<li> Computer Science and Engineering </li>
<li> Information Technology</li>
<li> Electronics and Communication </li>
</ul><br>
<ol type="I" start=4>
<li> Mechanical Engineering</li>
<li> Electronics and Electrical Engineering</li>
</ol><br>
<h3> Subject Details </h3><br>
<dl>
<dt> Web Technologies</dt>
<dd> This subject is related to the technologies used in web applications</dd>
</dl>
<a href="d:\feroz\gfec\wtlab\asgm1.html">back</a>
</body>
</html>
```


Tables:

Table is one of the most useful HTML constructs. Tables are find all over the web application. The main use of table is that they are used to structure the pieces of information and to structure the whole web page. Below are some of the tags used in table.

<table align=”center” | “left” | “right” border=”n” width=”n%” cellpadding=”n” cellspacing=”n”>……</table>

Every thing that we write between these two tags will be within a table. The attributes of the table will control in formatting of the table. Cell padding determines how much space there is between the contents of a cell and its border, cell spacing sets the amount of white space between cells. Width attribute sets the amount of screen that table will use.

<tr> ….. </tr>

This is the sub tag of <table> tag, each row of the table has to be delimited by these tags.

<th>……</th>

This is again a sub tag of the <tr> tag. This tag is used to show the table heading .

<td>…..</td>
This tag is used to give the content of the table.

Example 3: HTML code showing the use of table tag

```
<html>
<head>
<title> table page</title>
</head>
<body>
<table align="center" cellpadding="2" cellspacing="2" border="2">
<caption>Time for III year IT</caption>
<tr><th> I period </th>
<th> II peiord> </th>
</tr>
<tr>
<td> wt </td>
<td> uml</td>
</tr>
</table>
</body>
</html>
```


Color and Image:

Color can be used for background, elements and links. To change the color of links or of the page background hexadecimal values are placed in the <body> tag.

<body bgcolor = "#nnnnnn" text = "#nnnnnn" link= "#nnnnnn" vlink= "#nnnnnn" alink = "#nnnnnn">
The vlink attribute sets the color of links visited recently, alink the color of a currently active link. The six figure hexadecimal values must be enclosed in double quotes and preceded by a hash(#).

Images are one of the aspect of web pages. Loading of images is a slow process, and if too many images are used, then download time becomes intolerable. Browsers display a limited range of image types.

<body background = “URL”>
This tag will set a background image present in the URL.

Another tag that displays the image in the web page, which appears in the body of the text rather than on the whole page is given below

<img src=”URL” height=”n” width=”n” align = “top” | “center” | “bottom” >

Example 4: HTML code that implements color and image

```
<html>
<head>  <! -- This page implements color and image -->
<title> My Home page </title>
</head>
<body bgcolor="gray" text="magneta"  vlink="yellow" alink="brown">
<img src= ”d:\feroz\gfec\wtlab\asgm1.gif”>
</body>
</html>
```

Example 5: HTML code that implements background image

```
<html>
<head>  <! -- This page implements background image -->
<title> My Home page </title>
</head>
<body background=”d:\feroz\gfec\wtlab\asgm1.gif ">
</body>
</html>
```

Frames:

Frames provide a pleasing interface which makes your web site easy to navigate. When we talk about frames actually we are referring to frameset, which is a special type of web page. The frameset contains a set of references to HTML files, each of which is displayed inside a separate frame. There are two tags related to frames i.e., frameset and frame

<frameset cols=” % , %” | rows=” % , %”>……..</frameset>
<frame name=”name” src=”filename” scrolling =” yes” | “no” frameborder =”0”|”1”>

Forms:

Forms are the best way of adding interactivity of element in a web page. They are usually used to let the user to send information back to the server but can also be used to simplify navigation on complex web sites. The tags that use to implement forms are as follows.

<forms action=”URL” method = “post” | “get”>…….</form>

When get is used, the data is included as part of the URL. The post method encodes the data within the body of the message. Post can be used to send large amount of data, and it is more secure than get. The tags used inside the form tag are:

<input type = “text” | “password” | “checkbox” | “radio” | “submit” name=”string” value=”string” size=”n”>

In the above tag, the attribute type is used to implement text, password, checkbox, radio and submit button.

Text: It is used to input the characters of the size n and if the value is given than it is used as a default value. It uses single line of text. Each component can be given a separate name using the name attribute.

Password: It works exactly as text, but the content is not displayed to the screen, instead an * is used.

Radio: This creates a radio button. They are always grouped together with a same name but different values.

Checkbox: It provides a simple checkbox, where all the values can be selected unlike radio button.

Submit: This creates a button which displays the value attribute as its text. It is used to send the data to the server.

<select name=”string”>…..</select>

This tag helps to have a list of item from which a user can choose. The name of the particular select tag and the name of the chosen option are returned.

<option value=”string” selected>……</option>

The select statement will have several options from which the user can choose. The values will be displayed as the user moves through the list and the chosen one returned to the server.

<textarea name=”string” rows=”n” cols=”n”>…….</textarea>
This creates a free format of plain text into which the user can enter anything they like. The area will be sized at rows by cols but supports automatic scrolling.

Example 6: HTML code that implements forms

```
<html>
<head>
<title>form</title>
</head>
<body>
<p align="left">Name:<input type="text" maxlength=30 size=15>
<p align="left">Password:<input type="password" maxlenght=10 size=15>
<p align="left">Qualification: <br><input type="checkbox" name="q" value="be">B.E
<input type="checkbox" name="q" value="me">M.E
<p align="left">Gender:<br> <input type="radio" name="g" value="m">Male
<input type="radio" name="g" value="f">Female
<p align="left">course:<select name="course" size=1>
<option value=cse selected>CSE
<option value=it>CSIT
</select>
<p align="left">Address:<br><textarea name="addr" rows=4 cols=5 scrolling=yes></textarea>
<p align="center"><input type="submit" name="s" value="Accept">
<p align="center"><input type="reset" name="c" value="Ignore">
</body>
</html>
```

Example 7: HTML code that implements frames

```
<html>
<head>
<title> My page </title>
</head>
```

```
<frameset rows="25%,50%">
<frame name="a" src="f:\feroz\wtlab\asgm1.html">
<frameset cols="25%,50%">
<frame name="b" src="f:\feroz\wtlab\index.html">
<frame name="abc" src="f:\feroz\wtlab\welcome.html">
</frameset>
</frameset>
</html>
```

Cascading Stylesheets:

One of the most important aspects of HTML is the capability to separate presentation and content. A style is simply a set of formatting instructions that can be applied to a piece of text. There are three mechanisms by which we can apply styles to our HTML documents.

- Style can be defined within the basic HTML tag.
- Style can be defined in the <head> tag
- Styles can be defined in external files called stylesheets which can then be used in any document by including the stylesheet via a URL.

A style has two parts: a selector and a set of declarations. The selector is used to create a link between the rule and the HTML tag. The declaration has two parts: a property and a value. Declarations must be separated using colons and terminated using semicolons.

Selector{property: value; property: value …….}

Properties and values in styles:

Fonts:

Font-family: <family name.
Font-style: normal | italic | oblique
Font-weight: normal | bold | bolder | lighter
Font-size: small | medium | large | smaller | larger

Backgrounds and Colors

Color: value
Background-color: value
Background-image: URL

Text

Text-decoration: none | underline | overline
Text-transformation: none | uppercase | lowercase
Text-align: left | right | center | justify

Text-indentation : length | percentage

Example 7: HTML code representing cascading style sheet

```
<html>
<head>
<title>My Web Page</title>
<style type="text/css">
h1{font-family:mssanserif;font-size:30;font-style:italic;font-
weight:bold;color:red;background-color:blue;border:thin groove}
.m{border-width:thick;border-color:red;border-style:dashed}
.mid{font-family:BankGothicLtBT;text-decoration:link;texttransformation:uppercase;
text-indentation:60%}
</style>
</head>
<body class="m">
<h1> Green Fort Engineering College</h1>
<p class="mid">Jawaharlal Technological University Hyderabad</p>
</div>
</body>
</html>
```


UNIT II

Introduction to JavaScript

A number of technologies are present that develops the static web page, but we require a language that is dynamic in nature to develop web pages a client side. Dynamic HTML is a combination of content formatted using HTML, cascading stylesheets, a scripting language and DOM.

JavaScript originates from a language called LiveScript. The idea was to find a language which can be used at client side, but not complicated as Java. JavaScript is a simple language which is only suitable for simple tasks.

Benefits of JavaScript

Following are some of the benefits that JavaScript language possess to make the web site dynamic.

- It is widely supported in browser
- It gives easy access to document object and can manipulate most of them.
- JavaScript can give interesting animations with many multimedia data types.
- Special plug-in are not required to use JavaScript
- JavaScript is secure language

JavaScript code resembles the code of C language, The syntax of both the language is very close to each other. The set of tokens and constructs are same in both the language.

Example 8: A Sample JavaScript program

```
<html>
<head><title>java script program</title>
<script languague="javascript">
function popup()
{
var major=parseInt(navigator.appVersion);
var minor=parseInt(navigator.appVersion);
var agent=navigator.userAgent.toLowerCase();
document.write(agent+" "+major);
window.alert(agent+" "+major);
}
function farewell()
{
window.alert("Farewell and thanks for visiting");
```

```
}
</script>
</head>
<body onLoad="popup()" onUnload="farewell()">
</body>
</html>
```

- JavaScript program contains variables, objects and functions.
- Each line is terminated by a semicolon. Blocks of code must be surrounded by curly brackets.
- Functions have parameters which are passed inside parenthesis
- Variables are declared using the keyword var.
- Script does not require main function and exit condition.

Example 9: JavaScript program that shows the use of variables, datatypes

```
<html>
<head>
<title> My Sample JavaScript program</title>
<script languate="javascript">
function disp()
{
var rno,sname,br,pr;
rno=prompt("Enter your registration number");
sname=prompt("Enter your Name");
br=prompt("Enter your branch Name");
pr=prompt("Enter the percentage");
document.writeln("<h2> Your Registration No. is :</h2>"+ rno.toUpperCase());
document.writeln("<h2> Your Name is :</h2>"+sname.toUpperCase());
document.writeln("<h2> Your Branch Name is :</h2>"+br.toUpperCase());
document.writeln("<h2> Your Overall Percentage is :</h2>"+pr);
document.close();
}
</script>
```

```
</head>
<body onLoad="disp()">
</body>
</html>
```

Example 10: JavaScript program showing the using of constructs

```
<html>
<head> <title> Factorial</title> </head.
<body>
<script language="javascript">
function fact(n)
{
var i,f=1;
for(i=1;i<=n;i++)
{
f=f*i;
}
return(f);
}
var x,n,f;
x=prompt("Enter the number");
f=fact(x);
document.writeln("Factorial of "+x+" is "+f);
document.close();
</script>
</body>
</html>
```


Objects in JavaScript

JavaScript is not a pure object oriented programming language, but uses the concept of objects. The new keyword used here is to create an object, it allocates memory and storage. Objects can have functions and variables. To differentiate between global

variables and those which are part of an object but may have the same name, JavaScript uses this keyword. When referring to a property of an object, whether a method or a variable, a dot is placed between the object name and the property.

Example 11: JavaScript program using objects

```
<html>
<head>
<script language=”javascript”>
function demo1()
{
Popup(“Hello”);
Obj= new sample (2, 4);
alert(obj.x + obj.y);
}
function sample(x,y)
{
this.x=x;
this.y=y;
}
</script>
</head.
<body onLoad=”demo1( )”>
</body>
</html>
```


Regular Expression

A script language may take name data from a user and have to search through the string one character at a time. The usual approach in scripting language is to create a pattern called a regular expression which describes a set of characters that may be present in a string.

```
var pattern = “target”;
var string = “can you find the target”;
```

string.match(pattern);

But the above code can also be written using regular expression as a parameter, as shown below.

```
var pattern = new RegExp("target");
var string = "can you find the target";
pattern.exec(string);
```

Regular expression is a javascript object. Dynamic patterns are created using the keyword new.

```
regex =  new RegExp("feroz | amer");
```

Example 12: JavaScript code to implement RegExp

```
<html>
<head>
<body>
<script language="javascript">
var re = new RegExp("[A | a]mer");
var msg=" Have you met Amer recently";
var res= re.exec(msg);
if(res)
{
alert( " I found " + res[0]);
}
else
{
alert(" I didn't find it");
}
</script>
</body>
</html>
```


Functions:

Regular Expressions are manipulated using the functions which belong to either the RegExp or String class.

Class String functions

match(pattern)

This function searches a matching pattern. Returns array holding the results.

replace(pattern1, pattern2)

Searches for pattern1. If the search is successful pattern1 is replaced with pattern2.

search(pattern)

Searches for a pattern in the string. If the match is successful, the index of the start of the match is returned. If the search fails, the function returns -1.

Class RegExp functions

exec(string)

Executes a search for a matching pattern in its parameter string. Returns an array holding the results of the operation.

test(string)

Searches for a match in its parameter string. Returns true if a match is found, otherwise returns false.

Built in objects:

The document object

A document is a web page that is being either displayed or created. The document has a number of properties that can be accessed by JavaScript programs and used to manipulate the content of the page.

Write or writeln

Html pages can be created using JavaScript. This is done by using the write or writeln methods of the document object.

Document.write("<body>");
Document.write("<h1> Hello </h1>");

The form object

Two aspects of the form can be manipulated through JavaScript. First, most commonly and probably most usefully, the data that is entered onto your form can be checked at submission. Second you can actually build forms through JavaScript.

Example 13:

Validate.js

```
function validate()
{
var t1=document.forms[0].elements;
var t2=parent.frames['f4'].document;
var bg1=t1.bg.value;
var c1=t1.c.value;
t2.open();
t2.write("<body bgcolor="+bg1+">");
t2.write("Candidate name is : "+c1);
t2.write("</body>");
t2.close();
}
```

Mypage.html

```
<html>
<head>
<script language = "javascript src= "D:\Documents and Settings \ Amer\ Desktop \ amer\
p6 \  validate.js">
</script>
</head>
<body>
<form>
Background Color: <input type="text" size=16 name="bg" value="white">
Candidate's name:<input type="text" size=16 name="c">
<input type="button" value="showit" onClick="validate()">
</form>
</body>
</html>
```

The browser object

Some of the properties of the browser object is as follows

Navigator.appCodeName

The internal name for the browser.

Navigator.appVersion

This is the public name of the browser.

Navigator.appVersion

The version number, platform on which the browser is running.

Navigator.userAgent

The strings appCodeName and appVersion concatenated together.

The Date object

JavaScript provides functions to perform many different date manipulation. Some of the functions are mentioned below.

Date( )

Construct an empty date object.

Date(year, month, day [,hour, minute, second])

Create a new Date object based upon numerical values for the year, month and day. Optional time values may also be supplied.

getDate( )

Return the day of the month

getDay( )

Return an integer representing the day of the week.

getFullYear( )

Return the year as a four digit number.

getHours( )
Return the hour field of the Date object.

getMinutes( )
Return the minutes field of the Date object.

getSeconds( )
Return the second field of the Date object.

setDate(day)
Set the day value of the object. Accepts values in the range 1 to 31.

setFullYear(year [,month, day])
Set the year value of the object. Optionally also sets month and day values.

toString( )
Returns the Date as a string.

Events:

JavaScript is a event-driven system. Nothing happens unless it is initiated by an event outside the script. The table below shows event, event handler and the description about the event handler.

Event	Handler	Description
Blur	onBlur	The input focus is moved from the object, usually when moving from a field of a form or from the form itself.
Change	onChange	The value of a field in a form has been changed by the entering or deleting data.
Click	onClick	The mouse is clicked over an element of a page.
Double click	onDblClick	A form elementor a link is clicked twice in rapid succession
Focus	onFocus	Input focus is given to an element. The reverse of blur
Keydown	onKeyDown	A key is pressed but not released

Keypress	onKeyPress	A key is pressed.
Keyup	onKeyUp	A pressed key is released
Load	onLoad	The page is loaded by the browser
Mousedown	onMouseDown	A mouse button is pressed
Mousemove	onMouseMove	The mouse is moved
Mouseout	onMouseOut	The mouse pointer moves off an element
Mouseover	onMouseOver	The mouser pointer moved over an element
Move	onMove	A window is moved
Resize	onResize	A window is resized
Submit	onSubmit	A form is submitted
Unload	onUnload	The user leaves the web page.

Dynamic HTML with JavaScript

Data Validation

Data validation is the common process that takes place in the web sites. One common request is for a way of validating the username and password. Following program shows the validation of data which uses two frames, in one frame user is going to enter the data and in the other frame equivalent result is going to be displayed.

Example 14. JavaScript code for data validation

Mypage.html

```
<html>
<head>
<title>frame page </title>
</head>
<frameset rows="20%,*">
<frame name="f1" src="">
<frameset cols="20%,*">
<frame name="f2" src="">
<frameset cols="50%,*">
<frame name="f3" src="D:\Documents and Settings\Amer\Desktop\amer\p6\reg.html">
<frame name="f4" src="D:\Documents and Settings\Amer\Desktop\amer\p6\profile.html">
</frameset>
</frameset>
</frameset>
</html>
```

Myform.html

```
<html>
<head>
<script language = "javascript" src = "D:\ Documents and Settings \ Amer\ Desktop\ amer\ p6\ validate.js">
</script>
</head>
<body>
<form>
Background Color: <input type="text" size=16 name="bg" value="white">
Candidate's name:<input type="text" size=16 name="c">
<input type="button" value="showit" onClick="validate()">
</form>
</body>
</html>
```

Validate.js

```
function validate()
{
var t1=document.forms[0].elements;
var t2=parent.frames['f4'].document;
var bg1=t1.bg.value;
var c1=t1.c.value;
t2.open();
t2.write("<body bgcolor="+bg1+">");
t2.write("Candidate name is : "+c1);
t2.write("</body>");
t2.close();
}
```

UNIT III

XML: Defining Data for Web Applications

The markup language developed to add structural and formatting information to data and which was designed to be simple enough to be included in any application that language is Standard Generalized Markup Language and was adopted as standard by International Organization for Standardization(ISO).

Markup is nothing but instructions, which are often called as tags. There are many languages which shows how the data is displayed but no one describes what the data is. This is the point at which XML enters. XML is a subset of SGML. XML is used to describe the structure of a document not the way that is presented. XML is the recommendation of World Wide Consortium (W3C). The structure of basic XML is shown below which resembles HTML. The first line is the processing instruction which tells applications how to handle the XML. It is also serves as version declaration and says that the file is XML.

Example 15. Sample XML program

```
<?xml version=”1.0”?>
<college>
<studdetail>
<regno>05j0a1260</regno>
<name>
<firstname>feroz></firstname>
<lastname>amer</lastname>
</name>
<country name="india"/>
<branch>csit</branch>
</studdetail>
</college>
```


Valid an Well Formed XML

XML documents may be either valid or well formed. A well formed document is one which follows all of the rules of XML. Tags are matched and do not overlap, empty elements are ended properly, and the document contains an XML declaration. A valid XML document has its own DTD. XML should also conforms the rules set out in the DTD. There are many XML parsers that checks the document and its DTD

XML elements

XML documents are composed of three things i.e., elements, control information, and entities. Most of the markup in an XML document is element markup. Elements are surrounded by tags much as they are in HTML. Each document has a single root element which contains al of the other markup.

Nesting tags: Even the simplest XML document has nested tags. Unlike HTML these must be properly nested and closed in the reverse of the order in which they were opened. Each XML tag has to have a closing tag, again unlike HTML.

Case Sensitive: XML is case sensitive and you must use lower case for your markup.

Empty tags: Some tags are empty, they don't have content. Where the content of the tag is missing, it appears as
<content/>

Attributes: Sometimes it is important that elements have information associated with them without that information becoming a separate element.

Control Information
There are three types of control information i.e., comments, processing instructions, and document type declaration.

Comments: XML comments are exactly same as HTML. They take the form as
<!- -comment text - ->

Processing Instructions: Processing Instructions are used to control applications. One of the processing instructions is
<?xml version="1.0">

Document Type Declarations: Each XML document has an associated DTD. The DTD is usually present in separate file, so that it can be used by many files. The statement that includes DTD in XML file is
<?DOCTYPE cust SYSTEM "customer.dtd">

Entities
Entities are used to create small pieces of data which you want to use repeatedly throughout your schema.

Example 16: A Complete XML program

<?xml version="1.0"?>

<!DOCTYPE stud S?YSTEM "student.dtd">

<college>

<studdetail>

<regno>05j0a1260</regno>

<name>

<firstname>feroz></firstname>

<lastname>amer</lastname>

</name>

<country name="india"/>

<branch>csit</branch>

</studdetail>

</college>

Document Type Definition

Document type definition have been successfully used in SGML applications for many year. DTD are document centric. They are well understood. There are plenty of tools that support DTD

DTD for the XML document shown in the Example 15 is as follows

<!ELEMENT college(studetail+)>

<!ELEMENT studetail(regno, name+, country, branch)>

<!ELEMENT regno(#PCDATA)>

<!ELEMENT name(firstname, lastname)>

<!ELEMENT firstname(#PCDATA)>

<!ELEMENT lastname(#PCDATA)>

<!ELEMENT country(#PCDATA)>

<!ATTLIST country name CDATA #REQUIRED>

<!ELEMENT branch(#PCDATA)>

XML Schema

W3C developed a technology called XML schema which they accepted as a recommendation. XML schema is itself an XML application which means when you use it your only need a single grammar and can use your normal XML editor to create it.

Example 17: XML Schema for XML document shown in Example 14

```
<?xml version =”1.0” ?>
<xsd:schema xmlns =” http://.....”>
   <xsd:element name = “college”>
     <xsd:complexType>
       <xsd:sequence>
```

```
<xsd:element name = “studetail”>
  <xsd:complexType>
    <xsd:sequence>

      <xsd:element name = “regno” type = “xsd:string”/>

        <xsd:element name = “name”>
          <xsd:complexType>
            <xsd:sequence>

              <xsd:element name = “firstname” type=”xsd:string”/>
                <xsd:element name= “lastname” type = “xsd:string”/>

                  <xsd:element name = “country”>
                    <xsd:complexType>
                      <xsd:attribute name = “India” type= “xsd:string”/>
                    </xsd:complexType>
                  </xsd:element>

                      xsd:element name = “branch” type = “xsd:string”/>

            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>

      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```


Document Object Model

XML parsers can handle documents in any way that their developers choose. There are two models commonly used for parsers i.e., SAX and DOM. SAX parsers are used when dealing with streams of data. This type of parsers are usually used with java. SAX-based parsers run quickly.

DOM is and application program interface (API) for XML documents. The DOM API specifies the logical structure of XML documents and the ways in which they can be accessed and manipulated. The DOM API is just a specification. DOM-complaint applications include all of the functionality needed to handle XML documents. They can build static documents, navigate and search through them, add new elements, delete elements, and modify the content of existing elements. The views XML document as trees. The DOM exposes the whole of the document to applications. It is also scriptable so applications can manipulate the individual nodes.

Presenting XML

XML documents are presented using Extensible Stylesheet which expresses stylesheets. XSL stylesheet are not the same as HTML cascading stylesheets. They create a style for a specific XML element, with XSL a template is created. XSL basically transforms one data structure to another i.e., XML to HTML.

Example 18: Here is the XSL file for the XML document of Example 14

This line must be included in the XML document which reference stylesheet
<?xml:stylesheet type = "text/xsl" href = "student.xsl"?.

Here goes the XSL file

```
<xsl:stylesheet smlns:xsl ="uri:xsl".
<xsl:template match="/">
<html>
<body>
<h1> Student Database </h1.
<xsl:for-each select = "college">
  <xsl:for-each select = "studetail">
     <xsl:value-of select = "regno"/>
   <xsl:for-each select = "name">
       <xsl:value-of select = "firstname"/>
      <xsl:value-of select = "lastname"/>
</xsl:for-each>
   <xsl:value-of select="country/@name" />
  <xsl:value-of select = "branch"/>
</xsl:for-each>
</xsl:for-each>
</body>
</xsl:template>
</xsl:stylesheet>
```

UNIT IV

JAVA BEANS

Introduction to Java Beans

A Java Beans is software component that has been designed to be reusable in a variety of different environments. There is no restriction on the capability of a Bean. It may perform simple function, such as checking the spelling of a document, or complex function, such as forecasting the performance of a stock portfolio. A bean may be visible to an end user. One example of this is a button on a graphical user interface. A bean may be designed to work autonomously on a user's workstation or to work in cooperation with a set of other distributed components.

Advantages of Java Beans

- A bean obtains all the benefits of Java's "write once, run-anywhere" paradigm.
- The properties, events and methods of a bean that are exposed to an application builder tool can be controlled.
- A bean may be designed to operate correctly in different locales, which makes it useful in global markets.
- Auxiliary software can be provided to help a person configure a bean.
- The configuration settings of a bean can be saved in persistent storage and restored at a later time.
- A bean may register to receive events from other objects and can generate events that are sent to other objects.

BDK Introspection

Introspection is the process of analyzing a bean to determine its capabilities. This is a very important feature of Java Bean API, because it allows an application builder tool to present information about a component to a software designer. Without introspection, the java beans technology could not operate. One way exposed the properties, events and methods of bean to application builder tool is using simple naming conventions.

Design pattern for properties

Property is a subset of a bean's state. The values that are assigned to the properties determine the behavior and appearance of that component.

Simple properties:

A simple property has a single value. It can be identified by the following design patterns, where N is the name of the property and T is its type.

Public T getN( );
Public void setN( );

Boolean properties:

A Boolean property has a value of true or false. It can be identified by the following design patterns, where N is name of the property.

Public Boolean isN ( );
Public Boolean getN( );
Public void setN(Boolean value);

Indexed properties

An indexed property consists of multiple values. It can be identified by the following design patterns, where N is the name of the property and T is its type.

Public T getN(int index);
Public void setN(int index, T value);
Public T[ ] getN( );
Public void setN(T values[ ]);

Using Bound Properties

A bean that has a bound property generates an event when the property is changed. The event is of type PropertyChangeEvent and is sent to objects that previously registered an interest in receiving such notifications.

Example 17: Application that uses TickTock bean to automatically control the Color bean

Steps:

1. Go to menu bar of the bean box and select Edit | Events | propetyChange. We can now see a line extending from the button to the cursor
2. Move the cursor so that it is inside the Colors bean display area , and click the left mouse button. See the Event Target Dialog dialog box.
3. the dialog box allows your to choose a method that should be invoked when this event occurs. Select the entry labeled “change” and click the Ok button.

Using BeanInfo Interface

This interface defines several methods, including these:

PropertyDescription[ ] getPropertyDescriptors( )
EventSetDescriptor[ ] getEventSetDescriptors( )
MethodDescriptor[ ] getMethodDescriptors( )

The above methods will return array of objects that provide information about the properties, events, and methods of bean. SimpleBeanInfo is a class that provides default implementations of the BeanInfo interface, including the three methods just shown. We can extend this class and override on or more of them.

Constrained Properties

A bean that has a constrained property generates an event when an attempt is made to change its value. The event is of type PropertyChangeEvent. It is sent to objects that previously registered an interest in receiving such notifications. This capability allows a Bean to operate differently according to its run-time environment. A

Persistence

Persistence is the ability to save a Bean to nonvolatile storage and retrieve it at a later time. The information that is particularly important are the configuration settings.

Customizers

A bean developer can provide a customizer that helps another developer configure this software. A customizer can provide a step-by-step guide through the process that must be followed to use the component in a specific context.

Java Beans API

Interface	Description
AppletInitializer	Methods present in this interface are used to initialize Beans that are also applets
BeanInfo	This interface allows a designer to specify information about the properties, events and methods of a Bean.
Customizer	This interface allows a designer to provide a graphical user interface through which a Bean may be configured.
DesignMode	Methods in this interface determine if a Bean is executing in design mode.
PropertyChangeListener	A method in this interface is invoked when a bound property is changed.
Visibility	Methods in this interface allow a bean to execute in environments where graphical user interface is not available.

Class	Description
BeanDescriptor	This class provides information about a Bean.
Beans	This class is used to obtain information about a Bean
IntrospectionException	An exception of this type is generated if a problem occurs when analyzing a bean.
PropertyChangeEvent	This event is generated when bound or constrained properties are changed.
PropertyDescriptor	Instances of this class describe a property of a Bean

UNIT V

Introduction to Servlets

Servlets are used at server side. When a user request for a web page by entering the URL in the browser. The browser generate HTTP request to the appropriate web server. The web server maps this request with a specific file. The file is returned in the form of HTTP response. To handle these request at server side we require servlet.
There are several advantages of servlet:

- Performance is significantly better. Servlets execute within the address space of a web server.
- Servlets are platform independent, they are written in java . Several web servers, from different vendors such as Sun, Microsoft offer servlet API.
- Java security manager on the server enforces a set of restrictions to protect the resources on a server machine.
- The full functionality of java class libraries is available to a servlet.

Life cycle of a servlet

There are three methods related to the life cycle of a servlet i.e., init( ), service( ) and destroy( ). These methods are called at different times by the servlet. When a user enters a URL to a web browser, the browser generates an HTTP request for the URL and sends it to the appropriate server.

This HTTP request is received by the web server. The web server maps this request with a particular servlet. The servlet is dynamically retrieved and loaded into the address space of the web server.

Then server invokes init( ) method of the servlet. This method is invoked when the servlet is first loaded into memory. Here the initialization parameters are passed to the servlet.

The server then invokes the service method of the servlet to process the HTTP request. Servlet will read the data provided to it in the form of HTTP request and also formulate it. The servlet remains in the server's address space and is available for any HTTP request.

When the servlet is unloaded from the server, then destroy( ) method is called by the server to relinquish resources

Java Servlet Development Kit

The Java Servlet Development Kit contains the class libraries that you will need to create servlet. JSDK is available from the Sun Microsystems web site at java.sun.com.

A Simple Servlet
Example 19 Sample servlet program

HelloServlet.java

```
import java.io.*;
import javax.servlet.*;
```

```
public class HelloServlet extends GenericServlet
{
public void service(ServletRequest req, ServletRespone res) throws ServletException, IOException
{
res.setCcontentType("text/html");
printerWriter pw = res.getWriter( );
pw.println("Hello");
pw.close( );
}
}
```

Servlet API

There are two packages that are required to build servlets i.e., javax.servlet and javax.servlet.http. These packages constitute servlet API.

javax.servlet package:

There are number of interfaces and classes present in this package, they are described below.

Interface	**Description**
Servlet	Declares life cycle methods for a servlet
ServletConfig	Allows servlets to get initialization parameters
ServletContext	Enables servlets to log events
ServletRequest	Used to read data from a client request
ServletResponse	Used to read data to a client response

Class	**Description**
GenericServlet	Implements the Servlet and ServletConfig
ServletInputStream	Provides an input stream for reading requests from a client
ServletOutputStream	Provides an output stream for writing responses to a client.
ServletException	Indicates that a servlet error occurred.

Following are the interfaces and their methods

Servlet Interface

void destroy	Called when the servlet is unloaded
ServletConfig getServletConfig()	Returns a ServletConfig object that contains any initialization parameters
String getServletInfo	Returns a string describing the servlet
void init()	Called when the servlet is initialized
void service	Called to process a request from a client

ServletConfig Interface

ServletContext getServletContext	Returns the context for this servlet
String getInitParmeter(String param)	Returns the value of the initialization parameter name param
getInitParameterNames()	Returns all initialization parameter names

ServletContext interface

getAtrribute(String attr)	Returns the value of the server attribute named attr.
String getServiceInfo()	Returns information about the server.
Servlet getServlet(String sname)	Returns the servlet named sname.
getServletNames()	Returns the names of servlets in the server

ServletRequest Interface

String getParameter(String pname)	Returns the value of the parameter named pname
getParameterNames()	Returns the parameter names for this request
String[ ] getParameterValues()	Returns the parameter values for this request
String getProtocol()	Returns a description of the protocol
String getServerName()	Returns the name of the server
Int getServerPort()	Returns the port number.

ServletResponse Interface

PrinterWriter getWriter()	Returns a PrintWriter that can be used to write character data to the response
ServletOutputStream getOutputStream()	Returns a ServletOutputStream that can be used to write binary data to the response

Following are the classes and their methods

GenericServlet class

This class implements Servlet and ServletConfig interfaces

ServletInputStream class

The ServletInputStream class extends InputStream. It is implemented by the server and provides an input stream that a servlet developer can use to read the data from a client request. In addition to this, one more method is added which returns the actual number of bytes read

Int readLine(byte[ ] buffer, int offset, int size)

ServletOutputStream class

ServletOutputStream class extends OutputStream. It defines the print() and println() methods, which output data to the stream.

ServletException class

This class indicates that a servlet problem has occurred. The class has the following contructor

ServletException( )

ServletException(String s)

Reading Servlet Parameters

Example 20: Servlet program showing reading servlet parameters

```
Import java.io.*;
Import java.servlet.*;
Public class ParameterServlet extends GenericServlet
{
Public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
{
PrintWriter pw=res.getWriter;
Enumeration e = req.getParameterNames();
While(e.hasMoreElements())
{
String pname = (String)e.nextElement();
Pw.print(pname + " =");
String pvalue=req.getParameter(pname);
Pw.println(pvalue);
}
Pw.close();
}
}
```

Reading Initialization parameters

Example 21: Servlet program showing reading initialization parameters

```
import java.io.*;
import javax.servlet.*;
public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
{
ServletConfig sc= getServletConfig();
res.setContentType("text/html");
PrintWriter pw=res.getWriter();
pw.println(" Name : "+ sc.getInitParameter("name"));
```

```
pw.close();
}
}
```

javax.servlet.http package

There are number of classes and interfaces present in this package, they are as follows:

Interface	Description
HttpServletRequest	Enables servlets to read data from an HTTP request
HttpServletResponse	Enables servlets to write data to an HTTP response.
HttpSession	Allows session data to be read and written
HttpSessionContext	Allows sessions to be managed

Class	Description
Cookie	Allows state information to be stored on a client machine
HttpServlet	Provides methods to handle HTTP requests and responses

Following are the interfaces and their methods description

HttpServletRequest Interface

Cookie[ ] getCookies	Returns an array of the cookies in this request
String getMethod()	Returns the HTTP method for this request
String getQueryString()	Returns any query string in the URL
String getRemoteUser()	Returns the name of the user who issued this request.
String getRequestedSessionId()	Returns the ID of the session
String getServletPath()	Returns the part of the URL that identifies the servlet

HttpServletResponse Interface

Void addCookie(Cookie cookie)	Adds cookie to the HTTP response.
Void sendError(int c)	Send the error code c to the client
Void sendError(int c , String s)	Send the error code c and the message s
Void sendRedirect(String url)	Redirects the client to url

Cookie class

The Cookie class encapsulates a cookie. A cookie is stored on a client and contains state information. Cookies are valuable for tracking user activities. A servlet can write a

cookie to a user's machine via the addCookie() method of the HttpServletResponse interface. The names and values of cookies are stored on the user's machine. Some of the information that is used saved includes the cookie's

- Name
- Value
- Expiration date
- Domain and path

Following are the methods that are used by the Cookie class

String getComment()	Returns the comment
String getDomain()	Returns the domain
Int getMaxAge()	Returns the age
String getName()	Returns the name
String getPath()	Returns the path
Boolean getSecure()	Returns true if the cookie is secure
Int getVersion()	Returns the version
Void setComment(String c)	Sets the comment to c
Void setDomain(String d)	Sets the domain to d
Void setPath(String p)	Sets the path to p
Void setSecure(boolean secure)	Sets the security flag to secure

HttpServlet Class

The HttpServlet class extends GenericServlet. It is commonly used when developing servlets that receive and process HTTP requests. Following are the methods used by HttpServlet class

Void doGet(HttpServletRequest req, HttpServletResponse res)	Performs an HTTP GET
Void doPost(HSR req, HSR res)	Performs and HTTP POST
Void service(HSR req, HSR res)	Called by the server when and HTTP request arrives for this servlet.

Handling HTTP Requests and Responses

Example 22: Servlet program handling HTTP GET requests

```
import java.io.*;
import javx.servlet.*;
import javax.servlet.http.*;
 public class GetServlet extends HttpServlet
{
public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException
{
```

```
String name=req,getParameter(“name”);
res.setContentType(“text/html”);
PrintWriter pw=res.getWriter();
pw.println(“The Name is “);
pw.println(name);
pw.close();
}
```

Note: Same program you can use for Handling HTTP POST requests instead of using doGet we can use doPost