Unit- II HTML, XML & Scripting

1. **HTML:** HTML is a subset of SGML. HTML is used to define the composition of a web page, not the appearance.

   **Structure of HTML Document:**

   **On Code:** has only starting tag.
   Ex. `<tagname>`

   **Off Code:** has starting and ending tag.
   Ex. `<tagname>                                      </tagname>`

2. **Skeleton of HTML Document:** `<html>` and `</html>` tags marks the beginning and the end of the document. These tags inform the browser that the document is an HTML file. The HTML document can be divided into two sections:

   **Headed or Head Section:** to identify the heading or title of a document.
   `<title>               </title>` can appear in this section. It should not exceed beyond 60 characters.

   **Body Section:** The body of an HTML document contains the text that will show up on the web page.
   `<body>                     </body>` It gives a description of the document’s layout and structure.

   **Example:**

   ```html
   <html>
   <head>
   <title>Example of a Simple HTML Document</title>
   </head>
   <body>
   <p>HELLO WORLD! Here I am.</p>
   </body>
   </html>
   ```
3. **Tags:**

   **A. Container Tag:** These tags are in which bracket contain text or other tag elements are called container tag. These actually consist of two tags, a start tag and an end tag which enclose the text the effect.

   **B. Empty Tag:** These are stand alone and do not have bracket or contain text or any other tag elements.

**Tags Used in HTML:**

**I. Heading Tag:**
- The heading tags are used to display headings on the web page in various size.
- It can be used for main heading and sub headings to the web page.
- It supports six level of heading `<h1>` (largest) to `<h6>` (smallest).
- Each these headings always begin on a new line.
- Ex.

  ```html
  <body>
  <h1> First Heading </h1>
  <h2> Second Heading </h2>
  </body>
  ```

**II. Italic:** Creates italic text between `<i>` and `</i>`.

  ```html
  <i>...</i>
  ```

**III. Paragraph Formatting tags:**
- `<p>` `</p>` is used for beginning and end of the paragraph. Ending tag is not necessary.
- ALIGN attribute is used.
  ```html
  <p ALIGN = “CENTER”>
  Text here
  </p>
  ```
- `<br>` tags insert a line break.

### Attribute | Value | Description
--- | --- | ---
**align** | left, right, center, justify | Deprecated. Use styles instead. Specifies the alignment of the text within a paragraph.

### IV. Font Setting Tag:

- `<font> </font>`
- FACE attribute tag is used to set the font of the text.

```html
<font size="7" face="Georgia, Arial" color="maroon">TEXT HERE</font>
```

### V. DIV tag:

It is used for applying alignment and style characteristics to only a section of a document.

```
<div ALIGN = "CENTER">
Text here
</div>
```

### VI. Comment tag:

- It will not be interpreted or displayed by the browser
- Syntax `<!-- -->`

### VII. Some other tags are:

- `<STRONG>`-Strong Emphasis
- `<STRONG>` indicates strong emphasis.
<SUB>-Subscript

<SUB> indicates a subscript.

<SUP>-Superscript

<SUP> indicates superscript.

VIII. Preformatted tag:

- The <PRE> tag set creates preformatted text that maintains the spacing of the characters. Text is displayed in a nonproportional font.
- <PRE>...</PRE>

4. Creating lists:

- Ordered List: numbered list

<OL TYPE=A>

<LI>Apples
<LI>Peaches
<LI>Oranges
</OL>

<table>
<thead>
<tr>
<th>TYPE</th>
<th>Numbering Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1, 2,…</td>
</tr>
<tr>
<td>A</td>
<td>a, b,…</td>
</tr>
<tr>
<td>A</td>
<td>A, B,…</td>
</tr>
<tr>
<td>I</td>
<td>i, ii,…</td>
</tr>
<tr>
<td>I</td>
<td>I, II,…</td>
</tr>
</tbody>
</table>
• **Unordered list:** Bulleted list

```html
<UL TYPE=”square”>
<LI> COMPUTER CONCEPTS
<LI> MS-WINDOWS
<LI> MS-EXCEL
</UL>
```

*Type may be square, disc, and circle.*

• **Multilevel Outline:** Multilevel Outline is handled by START and VALUE.

```html
<OL START= “3”>
<LI>numbered as 3
<LI>numbered as 4
<LI VALUE = “4” this should be numbered as 4
<LI> this should be numbered as 5.
</OL>
```

• **Nested List:** You can nest the same or different kinds of lists.

5. **Creating Tables:** Table is a collection of rows and columns.

A. Table is created by `<table>……. </table>` having following attributes:

```html
<table ALIGN = “…….”
BORDER =“…….”
CELLPADDING = “……..”
CELLSPACING =“……..”
WIDTH =“……..” >

__________
__________
</table>
```

* Align may be center, left, right.
* Border may be 1,2,3, and so on.
* Space between content and inside of a cell is called **cellpadding.** It may be 1,2,3 and so on.
Space between cells in a table is called **cellspacing**. It may be 1,2,3 and so on.

Width of the table may be in.........

B. `<tr>.....</tr>` is used to add a row in a table.

C. `<td>.....</td>` is used to add table data with following attributes

```
<td ALIGN = "......."
    COLSPAN = "......."
    ROWSPAN = ".......">
```

Align may be center, left, right.

**Colspan:** Width of the cell in terms of numbers of columns when a cell occupies more than one column. It may 1,2,3 and so on.

**Rowspan:** Height of the cell in terms of numbers of rows when a cell occupies more than one row. It may be 1,2,3 and so on.

**Example: HTML code for creating two rows and two columns.**

```
<html>
<body>
<table border = "1" cellpadding = "6" cellspacing ="6">
<tr bgcolor = "cyan">
<td align = "center">INDIA <td>
<td align = "center">SOUTH AFRICA <td>
</tr>
<tr bgcolor = "cyan">
<td align = "center">AMERICA <td>
<td align = "center">BANGAL <td>
</tr>
</table>
</body>
</html>
```
6. **Creating Images:**
   
   A. The `<IMG>` tag is used to insert a GIF or JPEG graphic into the document. The attributes are:

   ```html
   <IMG SRC = "MyImage.gif" BORDER = “3” WIDTH = “100%”>
   ```

   B. **Image links:**

   ```html
   <A HREF="http://www.microsoft.com/">
   <IMG SRC = "MyImage.gif" Microsoft Homepage </A>
   ```

   C. **ALT attributes:** Alternative attribute is used when the required image is not supported by the browser.

   ```html
   <IMG SRC = “ABC.GIF” ALT = “A Simple Image”>
   ```

7. **Forms:** Form is a group of various elements just like examination form. A form film is a data-entry field on a page. A user supplies information into a field either by typing text or by selecting a value from a list of predicted values.

   **Application areas for forms:**
   
   - Education sites
   - Online purchasing of orders
   - Collecting feedback about a website
   - Providing the interface for a chat session etc.

   **Creating a Form:**

   `<form>……</form>` tag is used to create a form. It has three parts.

   A. Form header
   B. Input fields/ elements
   C. Action Buttons
A. Form header: In header we have the following attributes.

I. Method: It specifies how the browser should communicate with the server.
   
   It has two types:
   
   - GET: It is just getting or retrieving of data.
   - POST: It is used for insertion, updation and sending e-mail of data.

II. Action: It specifies what CGI script is used to process the data.

III. NAME: name of the form.

   <form action = “ABC.html” method = “POST” name=“Form1”>
   
   __________
   
   __________
   
   </form>

B. Input fields/ elements:

I. Various Inputs:

   <input TYPE= “……” NAME = “……..” VALUE = “……..” ALIGN= “…….”
   SIZE = “……..” MAXLENGTH = “……..”>

   Type may be text, button, checkbox, image, password, radio, reset, submit.

II. Multiple line text input:

   <textarea rows = “….” Cols = “…..” wrap = “…..”>

   Rows identify height, cols identify width and wrap means word wrapping. The value of
   wrap may be OFF, VIRTUAL (means long lines) and PHYSICAL (means wrap points).

III. Pull down menus:

   <select name = “list name” >
   
   <Option> option1
   
   <Option> option2
   
   </select>

C. Action buttons: We design action buttons using input tags.
Difference between GET and POST method:

<table>
<thead>
<tr>
<th>GET</th>
<th>POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) It used for static resources.</td>
<td>1) It used for dynamic resources.</td>
</tr>
<tr>
<td>2) In this, request parameters are append to the URL and send as part of request header.</td>
<td>2) In this request parameters are sent as part of request body.</td>
</tr>
<tr>
<td>3) All parameters are shown in the address bar of browser.</td>
<td>3) Not shown.</td>
</tr>
<tr>
<td>4) Maximum data that can be sent as request parameter is determined by the size of header which has a fixed size.</td>
<td>4) Unlimited data can be sent as request parameter.</td>
</tr>
</tbody>
</table>

Fixed data sent as request parameter

Unlimited data sent as request parameter.

8. **Frames**: Frames are the division of window either horizontally or vertically. Frames enable the user to divide a page into number of rectangular regions/window of various sizes.

**Example:**

```html
<html>
  <FRAMESET Rows = "30%,*">
    <Frame Src="Header.html" >
    <FRAMESET Cols = "25%,*">
      <Frame Src="Index.html" >
      <Frame Src="Details.html" >
    </FRAMESET>
  </FRAMESET>
</html>
```
 Rows or columns can be set in the form of percentage or in the form of pixels.

Some attributes of frame tag are:

i. Name: name of the document page.
ii. Scrolling: scroll bar setting (YES, NO or AUTO).
iii. MarginWidth: left and right margins in pixel.
iv. MarginHeight: top and bottom margins in the frame.

9. **Style Sheet**: style sheet is a collection of formatting styles which can be applied to a web page.

**Style Rules**: A style rule is a set of HTML tags specifying the formatting elements. A style rule can be applied to selected contents of a web page. A style rule can basically be split into two parts:

A. **Selector**: A selector is a string that identifies what elements the corresponding rule applies to and is the first part of the rule.

B. **Declaration**: This part of the rule is enclosed within curly brackets:

   \[ \text{Selector} \{ \text{Property}: \text{Value} \} \]

- **Selector**: Any HTML tag
- **Property**: Attribute like font, color etc.
- **Value**: Setting for the attribute.

**Types of Style rules:**

I. **Inline Styles**:

   \[ <htmltag STY LE = "property: value" > \]

**Example**:

   \[<html>

   \[<body>

   \[<h1 STY LE = "color: Lime Green" >\]
This is a style applied to an H1 element

</h1>
</body>
</html>

II. **Embedding Style Sheet:** You can group more than one style rule using <style>……..</style> tag pair unlike of applying it individually in inline style.

**Example:**

<html>
<head>
<style>
H1 { color: LimeGreen}
H1 { font-family: Arial}
H2 { color: red}
H2 { font-family: Times New Roman}
</style>
</head>
<body>
<H1> This is the H1 element </H1>
<H2> This is the H2 element </H2>
<H3> default text </H3>
</body>
</html>
III. Grouping Style Rules:

<html>
<head>
<style>
H1, H2 { color: Red, font-family: Arial}
</style>
</head>
<body>
<H1> This is the H1 element </H1>
<H2> This is the H2 element </H2>
<H3> default text </H3>
</body>
</html>

Selectors: There are four type of selectors used in HTML named Simple Selector, HTML Selector, Class Selector, ID selector.

I. Simple Selector: \textit{H1 \{color: blue\}}

II. HTML Selector: These selector use the name of HTML elements without brackets like: \textit{<p> tag becomes p}}

III. Class Selector: The class selector provides the ability to apply styles to specific parts of a document and do not necessarily to the whole document.
Syntax:

```
<html>
  <head>
    <style>
      .note{color: blue}
      .syntax{color: red}
      p.syntax{color:red}
      p{font-size: lage}
    </style>
  </head>
  <body>
    <p class = "syntax"> HELLO
    </p>
    <p class "note"> HI
    </p>
    <H1 class = "note"> Introduction
    </H1>
  </body>
</html>
```
IV. **ID Selector:** Like class selector, ID selector is also used to apply style to the selected parts of text. In this style **ID selector has a unique identifier.** An ID selector is preceded by a **hash mark (#).**

**Syntax:**

```html
<head>
  <style>
    #ID Selector. Class name {Property: value}
  </style>
</head>
<body>
  <p ID = "ID Selector name"> ID attribute 
</body>
```

**Example:**

```html
<html>
  <head>
    <style>
      #control {color: red} HELLO
    </style>
  </head>
  <body>
    <h1 ID = "control"> HELLO </h1>
  </body>
</html>
```
10. **Cascading Style sheet**: If you want to apply similar settings for all web pages in the website, this can be done by putting all the style rules in a style sheet file and then importing or linking it with your HTML document. This method of linking or importing is called cascading style sheet (CSS).

It must be saved with .CSS extension.

**Style Sheet Properties:**

I. **Font Properties**:
   a. Font-family: denotes font
   b. Font-size: denotes the size of the text.
   c. Font-style: style of the text like normal, bold, italic, etc.
   d. Font-weight: denotes the weight or darkness of the font. Ranges from 100 to 900 by increments of 100.

II. **Text Properties**:
   a. Letter-spacing: space between letters.
   b. Word-spacing: space between words.
   c. Vertical-align: denotes the vertical positioning of the text and images with respect to the base line. The possible values include baseline, sub, super, top, text-top, middle percentage values etc.
   d. Text-align: specifies the alignment of the text. The possible values are center, justify etc.
   e. Text-indent: denotes margins.
   f. Text-transform: denotes the transformation of text. The possible values are capitalize, uppercase, lowercase etc.
   g. Text-decorate: denotes the text decoration. Values will be blink, line-through, over line, underline etc.

III. **Color and Background Properties**:
   a. Color
   b. Background-color
   c. Background-image
   d. Background-image.
IV. **Box Properties:**

   a. Margin properties: like 15 pt. etc.
   b. Border properties:
      - Border-style: solid, double, groove, ridge etc.
      - Border-width: in terms of pt.
      - Border-color

V. **Padding Properties:** The space between an element’s border and its content can be specified four padding regions can be set using the padding-top, padding-right, padding-bottom and padding-left properties.

Applying CSS to an HTML document:

I. **In-line style:** One way to apply CSS to HTML is by using the HTML attribute style.

   Ex.
   
   ```html
   <html>
   <head>
   <title>Example</title>
   </head>
   <body style="background-color: #FF0000;">
   <p>This is a red page</p>
   </body>
   </html>
   ```

II. **Internal (Tag) Style:** Another way is to include the CSS codes using the HTML tag `style`.

   Ex.
   
   ```html
   <html>
   <head>
   <title>Example</title>
   </head>
   <style type="text/css">
   body {background-color: #FF0000;}
   </style>
   <body>
   <p>This is a green page</p>
   </body>
   </html>
   ```
III. **External (link to a style sheet) style:** Open Notepad (or whatever text editor you use) and create two files - an HTML file and a CSS file - with the following contents:

**default.html:**

```html
<html>
<head>
  <title>My document</title>
  <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
  <h1>My first stylesheet</h1>
</body>
</html>
```

**style.css:**

```css
body {
  background-color: #FF0000;
}
```

IV. **importing a CSS:** you can also import a style sheet for just a specific media:

```html
<style type="text/css">
  @import url("import4.css");
</style>
```
Ex.

Acss.css

```html
<style>
<!--
.note{color: blue}
.syntax{color: red}
p.syntax{color:red}
p{font-size: large}
-->  
</style>

Mn.html

```html
<html>
<head>
<style type="text/css">
@import url("acss.css");
</style>
</head>
<body>
<p class = "syntax">HELLO</p>
<p class="note">HI</p>
<H1 class = "note">Introduction</H1>
</body>
</html>
```
11. **XML (Extensible Markup Language):**

**Introduction:**

a. XML is the case sensitive markup Language.

b. With XML you can store information about your document and pieces of your document.

c. You can use that information as criteria for displaying page but also for validating digital signatures, sharing data across systems, processing data for other applications and much more.

d. **HTML is limited in:**
   
   - **Intelligence:** How will data knows itself?
   - **Adaption:** How will data changes in response to changing time?
   - **Maintenance:** How easily data is cared for?

e. **Reasons for picking up XML:**
   
   - Easy and simple.
   - Easy to maintain.
   - Contains only data and markup.

f. **Features of XML:**
   
   - XML can be used with existing protocol.
   - Support a wide variety of applications.
   - Compatible with SGML.
   - It is easy to write programs that process XML document.
   - XML documents are reasonable and clear to person.

g. **Extensible means:**
   
   - Reusable
   - Create your own data type derived from standard data types.
   - Reference of your code in same document.
12. **DTD (Document Type Definition):** A set of rules for defining the legal building blocks of a web document like HTML or XML. DTD can be of two types:

**A. Internal DTD:**

```xml
<?xml version="1.0"?>
<!DOCTYPE mail [ 
<!ELEMENT mail (to,from,heading,body)> 
<!ELEMENT to (#PCDATA)> 
<!ELEMENT from (#PCDATA)> 
<!ELEMENT heading (#PCDATA)> 
<!ELEMENT body (#PCDATA)> ]>

<mail>
<to>Ravi</to>
<from>Dolly</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</mail>
```

**B. External DTD:** External DTD is more preferable because it is the standard DTD, It is available and use every where:

```xml
<?xml version="1.0"?>
<!DOCTYPE mail "mail.DTD">

<mail>
<to>Ravi</to>
<from>Dolly</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</mail>
```
Mail.DTD:

```xml
<?xml version="1.0"?>
<!DOCTYPE mail [ 
<!ELEMENT mail (to,from,heading,body)> 
<!ELEMENT to (#PCDATA)> 
<!ELEMENT from (#PCDATA)> 
<!ELEMENT heading (#PCDATA)> 
<!ELEMENT body (#PCDATA)> ]>
```

13. XML Schema:
   a. XML Schema is an XML-based alternative to DTD.
   b. It describes the structure of an XML document.
   c. It is also referred to as XML Schema definition (XSD).
   d. It contains:
      • Defines elements, attributes that can appear in a document.
      • Defines child elements, order and number of child elements.
      • Defines whether an element is empty or can include text.
      • Defines data types for elements and attributes.
      • Defines default and fixed values for elements and attributes.
   e. XML Schemas are successors of DTD:
      • XML schemas are extensible for future purpose.
      • Are richer and more powerful than DTD.
      • Are written in XML.
      • It supports data types, namespaces.
Example:

```xml
<?xml version="1.0"?>
<xs:schema xmlns:xs="URL/XMLSchema"
  targetNamespace="URL"
  xmlns="URL"
  elementFormDefault="qualified">

  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

- `xmlns:xs="URL/XMLSchema"` indicates that the elements and data types used in the schema come from the “URL/XMLSchema”.
- `targetNamespace="URL"` indicates that the elements defined by this schema (note, to, from, heading, body) come from URL namespace.
- `xmlns="URL"` indicates that the default namespace is URL.
- `elementFormDefault="qualified"` indicates that any elements used by the XML instance document which were declared in the schema must be namespace qualified.
14. **Object Models:**

- Viewing XML using the “XML data source object”.
- **Data source objects** are used for what Microsoft calls data binding. **Data Binding** is Microsoft’s way of bringing data manipulation to the browser (Client) away from the server.
- Normally, if you want a new view on the data, you resubmit a query to the server. The server performs the necessary calculations and sends a new HTML page together with the data to the client.
- There are **two types of object models:**
  - **A.** DOM (Dynamic Object Model)
  - **B.** SAX (Simple API for XML)
A. DOM (Dynamic Object Model):

I. DOM is a platform and language independent object model for representing HTML or XML and related formats.

II. DOM supports navigation in any direction (e.g. parent and previous sibling), hence the DOM is likely to be best suited for applications where the document must be accessed repeatedly or out of sequence order.

III. Levels of DOM:
   a) Core DOM: standard model for any structured document.
   b) XML DOM: standard model for XML document.
   c) HTML DOM: standard model for HTML documents.

IV. DOM Nodes: According to the DOM, everything in an XML document is a node.
   a) The entire document is a document node.
   b) Every XML element is an element node.
   c) The text in the XML elements are text node.
   d) Every attribute is an attribute node.
   e) Comments are comment nodes.

V. Example:

```xml
<?xml version = "1.0" ?>
<bookstore>
  <book category = "cooking">
    <title lang = "en"> Sweet Disses </title>
    <author> kshitij </author>
    <year> 2005 </year>
    <price> 30.00 </price>
  </book>
<bookstore>
```
VI. History of DOM: It is a W3C (world wide web construction) standard. W3C DOM was introduced in the mid 1990. DOM1 was introduced in October 1998 and DOM2 was introduced in November 2000. Dom2 was specific on the style sheet object model.

Dom has various levels according to requirement and optional modules:

- **Level 0:** It includes DHTML objects. It is basically not published by W3C but rather a shorthand that refers to what existed before the standardization process.
- **Level 1:** Navigation of DOM (HTML and XML), document (tree structure), and content manipulation.
- **Level 2:** XML namespace support, filtered views and events.
- **Level 3:** It has six specifications: Core, Load and Save, Views and Formatting, Requirements & Validation.
B. SAX (Simple API for XML):

SAX is a serial access parser API for XML. SAX provides a mechanism for reading data from an XML document. It is a popular alternative to the DOM.

SAX parser for XML processing: A parser which implements SAX (SAX parser) functions as a stream parser, with an event-driven API. The user defines a number of call back methods that will be called when event occur during parsing. The SAX events include:

a) XML text nodes.
b) XML element nodes.
c) XML processing instructions.
d) XML comments.

SAX parsing is unidirectional. Previously data cannot be re-read without starting the parsing operation again.

Example:

```xml
<?xml version="1.0"?>

<RootElement param="value">
    <FirstElement>Some Text</FirstElement>
    <SecondElement param2="something">
        Pre-Text <Inline>Inlined text</Inline> Post-text.
    </SecondElement>
</RootElement>
```
THE XML document, when passed through a SAX parser, will generate a sequence of events like the following:

i. XML processing instruction, named XML, with attribute version = “1.0”.
ii. XML Element start, named RootElement, with an attribute param equal to "value"
iii. XML Element start, named FirstElement
iv. XML Text node, with data equal to "Some Text" (note: text processing, with regard to spaces, can be changed)
v. XML Element end, named FirstElement
vi. XML Element start, named SecondElement, with an attribute param2 equal to "something"
vii. XML Text node, with data equal to "Pre-Text"
viii. XML Element start, named Inline
ix. XML Text node, with data equal to "Inlined text"
x. XML Element end, named Inline
xi. XML Text node, with data equal to "Post-text."
xii. XML Element end, named SecondElement
xiii. XML Element end, named RootElement

Advantages of SAX:

i. The quantity of memory that a SAX parser must use in order to function is typically much smaller than that of a DOM parser because Tree takes a lots of memory in DOM.
ii. SAX parser takes less time than DOM parser because memory allocation takes time for tree generation.
iii. Streamed reading from disk is impossible in DOM.

Disadvantages of SAX:

i. SAX is unidirectional. (Suppose next generation of ID need previous ID records).
ii. Some XML processing needs tree like XSLT, XPath. DOM already constructed tree for all.