HTML

HTML :

Html document consists of html tags and some text.

HEML TAGS:

1. Html tags are used to markup html element.
2. Html tags are surrounded by <and> Characters these characters are called as angler brackets.
3. Html tags always come in pairs. That is tags contain a starting tag and ending tags.
4. The text in b/w the starting tag and ending tag is called as element content.
5. Html tags are free defined tags.
6. Html tags are not case sensitive. That the upper and lower document is html document.
7. The html tag <html> in a document represents. That the document is html document.
8. The entire html document must be written starting html tag <html> and ending html </html>.
9. Html document is divided into two sections.

(I) Head section:

This section is used to provide general information about the html doc. This section is representing by <head>.

Ex: Title, Meta etc.

(II) Body section:

This section is used to display text are images on the Browser. This section is representing by <body>.

Html document structure:

```
<html>
        <head>
                <title> </title>
        </head>
        <body>
                Welcome to any web page
        </body>
```

```
                    </html>
```

Step1: Processor to write executes the html document. Open an editor and type html document.

```
<html>
        <head>
                <title>First html</title>
        </head>
        <body>
                Welcome to any web page
        </body>
</html>
```

Step2: Save the html document any name having. Html as the extension.
Step3: To execute the html document to open it in any Browser. To html language is an error free language. That is it will not display any error massage on the Browser. Even it the html document contain any error.

Html comments:
Comments are used to make the code name readable or they are used to explain the code.
HTML comments begins with <!--line one comment-->

```
EX :<!--line one comment -- >
     <!--line one comment
          line two comment
                 .
                 .
                 .
        N no. of line comments -- >
```

Attributes:

1. Attributes are used to provide additional information about the html elements tags.
2. The attributes must be specified in starting tags.
3. The attributes always come in pears.
   attribute name = attribute value
4. The attribute value can be enclosed with in single codes are double codes.

5. Every html tag can contain attribute.

Note: Specifying the attributes for a tag is optional.

Body tag attributes:

1. **Bgcolor:** This attributes specified the background color to displayed for a html document.
2. **Text** : This attributes specified the color of the text to be displayed on the html document. Color can be specified (on the html document) for two formats.
    a. Specified the color have directly.
    b. We can specify the color by combining Red, Green and Blue. The format should be #rrggbb. The color combination is 6 digit Hexadecimal numbers.
3. **Background**: This attribute specify an image as the background to the html document.

```
<body bgcolor = “#ffffoo” text = “#ffoooo”>
                Good Afternoon
</body>
```

Note: Specifying the attributes is optional, we can specify any number of attributes and they can specified in any order.

Formatting tags:

<b>: This tag is use to display a text in Bold face.

    <b>Bold</b>

<strong>: This tag is similar to <b> used to display the text in Bold face.

    <strong>Bold</strong>

<i>: This tag is use to emphasize a text by display in Italics format.

    <i>Italics</i>

<em>: This tag is use to emphasize a text by display in Italics format.

    <em>Italics</em>

<tt>: This tag is use to display a text in Teletype font. That is typewriter font.

    <tt>Teletype</tt>

<s>: This tag is used to display a Strike the text.

    <s>Strike</s>

<strike>: This tag is similar to <s> use to display a Strike out text.

```
<strike>Strike</strike>
```
<del>: This tag is use to Delete a text by Striking it out.
```
<del>Strike</del>
```
<ins>: This tag is use to Insert a line below the text.
```
<ins>underline</ins>
```
<u>: This tag is use to underline a text.
```
<u>underline</u>
```
<sup>: This tag is use to display a text as a Superscripted. That the text will be displayed above the normal text.
```
a<sup>2</sup>+b<sup>2</sup>
```
<sub>: This tag is use to display a text as a Subscripted. That is the text will be displayed below the normal text.
```
H<sub>2</sub>SO<sub>4</sub>
```
<big>: This tag is use display a text solidly Bigger then the normal text.
```
<big>text</big>
```
<small>: This tag is to display a text solidly Smaller then the normal text.
```
<small>text</small>
```
<center>: This tag is to display a text in the Center of the webpage(html document).
```
<center>Welcome</center>
```

Heading tags: These tags are used to perform font changes display the text in boldface and the heading tags by default align the text tag left side. These are six levels of heading tags. H1, H2, H3, H4, H5 and H6. H1 is the biggest in heading and H6 is the biggest in heading and H6 is the smallest heading.

We can change the alignment of the headings by using align attributes.

Ex: align = “left |center |right”
```
<H1 align = “left”>Heading1</H1>
<H2 align = “left”>Heading2</H2>
<H3 align = “center”>Heading3</H3>
<H4 align = “center”>Heading4</H4>
<H5 align = “right”>Heading5</H5>
<H6 align = “right”>Heading6</H6>
```
<pre>:(preformatted text)

This tag is used to display a text in preformatted manner. The browser preserve all the white spaces as it is.

```
        Ex:<pre>
                    To,
                     The manager,
                    HDFC bank,
                    Nellore,
Sub: Regarding loan to buy a house.
        Respected sir/madam,
                              I………………………
                                        …………………………
                                                  ……………………….
                                                                    Thanking you.
</pre>
```

<u>**<br>**</u>: This tag is used to stop displayed in the content at that point and displayed content in a new line. The <br> will started new line where ever it is specified.

Note:<Br> is an empty tag that is <BR> tag doesn't content any element content.

Ex:

```
    <body>
             line one</br>
              line two</br>
              line three
    </body>
```

<u>**<p>**</u>: This tag is used to display a paragraph. The <p> will add paragraph break (inserts an empty line before and after the paragraph). The paragraphs are is by default to left side. We can change the alignment by using the alignment attribute.

Ex: align = "left |center |right"

```
<p align = "left">this is a paragraph</p>
<p align = "center">This is a paragraph</p>
<p align = "right">This is a paragraph</p>
```

<hr>: This tag is used to display a horizontal line in a html document. Using the horizontal rule we can divide the web page into multiple sections.

***Attributes of <hr>**:

1. **Width**: This attribute specifies the length of the Horizontal rule to be displayed in a web page. The width can be specified either in pixels or percentage.

   Ex:<hr width = "300"/> or <hr width = "75%"/>
   The width can be specified either in pixels or percentage.

2. **Color**: This attribute specifies the color of the horizontal rule to be displayed in a web page.

   Ex:<hr color = "red"/>

3. **Size**: This attribute specifies the thickness of the horizontal rule to be displayed.

   Ex:<hr size "10"/>

4. **align**: We can change the alignment of the horizontal rule by using this align attribute. By default the horizontal rules are aligned to center.

   Ex :< hr align = "center"/>
   <hr width = "75%" align = "center" color = "blue" size = "10"/>

Note: hr is an empty tag.

Lists: In html document we can present the data in the form of list. There are 3 types of lists.

1. Ordered List
2. Unordered List
3. Definition List

1) **<u>Ordered List</u>**: A list is said to be an ordered list if the items are displayed by using either by digits or by Alphabets or Roman numbers. By default digits are used to display order list. We can change the display by using “type” attribute.
   Type = “1 | a | A | i |I|”
   Ex: <ol type “I”>

```
        <li>Hyderabad</li>
        <li>Mumbai</li>
        <li>Chennai</li>
    </ol>
```

2) **<u>Unordered List</u>**: If a list is displayed with the help of some graphical symbols then it is called as unordered list. We can change the display by using “Type” attribute.
   Type = “disc | square | circle”
   Ex:

```
    <Ul>
        <li>Hyderabad</li>
        <li>Mumbai</li>
        <li>Chennai</li>
    </Ul>
```

3) **<u>Definition List</u>**: A list is said to be a definition list if we provide a definition or description to every item.
   Ex:

```
    <dl>
        <dt>Hyderabad</dt>
        <dd>Biriyani</dd>
        <dt>Chennai</dt>
        <dd>somber</dd>
        <dt>Mumbai</dt>
        <dd>bomb</dd>
    </dl>
```

We can place a list inside another list that is nesting of lists is possible.
Ex:

```
    <Ul>
        <li>Hyderabad</li>
        <ol>
            <li>Ameerpet</li>
           <li>Hi-tech</li>
           <li>Birla temple</li>
        </ol>
            <li>Mumbai</li>
            <li>Chennai</li>
    </Ul>
```

<Font>: This tag can be used to apply font changes, change the text color or face or size.

Attributes of <font>:

1. **Color**: This attribute specifies the color of the text to be displayed.
   Ex: <font color = "red"> Hello </font>

2. **Size**: This attribute specifies the font size to be displayed.
   Ex: <font size = "10">Hello</font>

3. **Face**: face will specify the place of the text to be display.
   Ex: <font face = "Arial">Hello </font>

<font color = "red" size = "10" face = "Arial"> Hello</font>

Note: Font tag is dependent tag. That is <font> will not apply any changes without the attributes.

<img> :(image) this tag is used to display and img on the html document.

Attributes of<img>:

1. **Src(source)**: This attribute specifies of the URL(Address of the image).

2. **Border**: This attribute specifies of the thickness of the border around the image. By default thickness is 'zero' pixels.

3. **Width**: This attribute specifies of the width of the an image to be displayed in a html document.

4. **Height**: This attribute specifies of the Height of the image to be displayed in a html document.

5. **Alt(alternate text)**: This attribute specifies a text to be displayed on the html document when an image is not loaded.

6. **Hspace(Horizontal Space)**: This attribute specifies the amount of space empty space to be displayed on the left and right sides of the image.

7. **VSpace(vertical space)**: This attribute specifies the amount of empty space to be displayed on the top and bottom of the image.

   Ex:<img src = “smile.jpg” border = “2” width = “100” height = “100” alt = “smile logo” hspace = “50” vspace = “30”/>

Note: An <img> is an empty tag and (source) “src” is a mandatory attribute to display an image.

<Table>: This tag is used to display a table in a html document
Attributes of <table>:
1. **Border**: This attribute specifies the thickness of the border to be displayed. By default are zero pixels.
2. **Border color**: This attribute specifies the color of the border to be displayed.
3. **Width**: This attribute specifies the width of the table.
4. **Height**: This attribute specifies the height of the table.
5. **Align**: This attribute can be used to align a table in a web page.
6. **Bgcolor**: This attribute specifies a bgcolor for the table.
7. **Background**: This attribute is used to display an image the background to be table.
8. **Cell padding**: This attribute specifies the amount of space between cell content and cell border.
9. **Cell spacing**: This attribute specifies the space between the cells and b/w the cell border and table border.

Sub tags of <table>:

1. **<tr>(table row)**: This tag is used to create a row in a table. The no. of <tr> will indicate the no. of row in a table.
2. **<td>(table data)**: This tag is used to create a column(cell) in a particular row. The no. of <td>'s with indicate no. of columns in a particular row.
3. **<th>(table heading)**:This tag is used to create a column in a particular row similar to <td> but <th> will display the text in bold face and the text will be centered.
4. **<caption>**: This tag is used to display a heading to a table.

Attributes of sub tags:

1. **Row span**: This attribute is used to span (merge) the cells vertically.
2. **Colspan**: This attribute is used to span (merge) the cells horizontally.
3. **Valign**: This attribute is used to align the text vertically. Valign = "top | middle |bottom" by the default it is vertically aligned to middle. In this sub tags we can also use align, bgcolor, and background attributes.

   Ex:
```
<html>
        <head>
              <title>Table<title>
        </head>
        <body>
              <table border = "2" width = "150" height = "150" bordercolor =
              "red" bgcolor = "pink" cellpadding = "10">
        <caption>number table</caption>
                      <tr>
                              <th>one</th>
                              <th>two</th>
                              <td align = "center">three</td>
                      </tr>
                      <tr>
                              <td colspan = "2" align = "center">four</td>
                              <td rowspan = "2" valign = "top" bgcolor = "red">
                              six</td>
                      </tr>
                      <tr>
                              <td>seven</td>
                              <td>eight</td></tr>
```

```
                    </table>
                    </body>
          <html>
```

<Marquee>: This tag use to display a scrolling text on a web page.

Attributes of <Marquee>:

1. **Behavior**: This attribute specifies the behavior of the scrolling text in a web page.

   Ex:

   Behavior = "scroll | slide | alternate"(the default behavior is scroll)
2. **Direction**: This attribute specifies the direction in which the text will be scrolled. By default the text scrolls towards left side.
3. **Bgcolor**: This attribute will display bgcolor to the scrolling text.
4. **Scroll amount**: This attribute specifies the speed of the scrolling text.

   Ex:

   <marquee behavior = "scroll" direction = "right" bgcolor = "red" scrollamount = "45">flash news</marquee>

<a>: This tag is use to create link b/w the html documents are any other web resource.

Attributes of <a>:

1. **Href(hyperlinking reference)**: This attribute specifies the URL(address) of the resource to which the hyperlink as the link reference.
2. **Target**: This attribute will specifies were to open the linked document.

   Target = "_self" | "_blank" (_under scor) the default value _self.
3. **Name**: This attribute is used to create named Anchor. The named anchor can be used to jump from one location to another location with in a html document.

   Ex: <a href = "image.html" target = "_blank">click here</a>

Frame: Frames can be used to display multiple html documents in the same browser at the same time.

<frameset>: This tag is use to divide a frame into multiple frames.

Attributes of <frameset>:

1. **Rows**: This attribute is a list of values separated by a comma (,). The no. of values in a list will decide the no .of horizontal divisions.

2. **<u>Cols</u>**: This attribute is a list of values separated by a comma (,). The no. of values in a list will decide the no. of vertical divisions.
3. **<u>Border</u>**: This attribute specifies the thickness of the border with in a frameset.

<u><frame></u>: This tag represents and individual frame with in a frameset.

<u>Attributes of <frame></u>:

1. **<u>Src(source)</u>**: This attributes specifies the URL(address) of the html document to be displayed with in the frameset.
2. **<u>Scrolling</u>**: This attribute will decide where there frame will contain a scrolling bar of not.

   Scrolling = “auto | yes | no”
   The default scrolling value is also “auto”.
3. **<u>Name</u>**: This attribute is use to assign a name to a frame.
4. **<u>No resize</u>**: It is a flag which indicates that the frame can’t be resized.

   <u>Ex</u>:
```
<html>
      <frameset rows = “30%,*” border = “10”>
      <frameset src = “welcome.html” noresize/>
      <frameset cols = “100,*”>
      <frame src = “sample.html”/>
      <frame src = “image.html”/>
      </frameset>
      </frameset>
```

   <u>Note</u>: The frameset tag must be written outside the <body>.

<u>Forms</u>: Form is an area which contains some input elements where the user can provide the information. Forms are used to submit the data to the server are any other resource. The html document can have any no. of forms. To create a form in html document we use <form>.

<u>Attributes of <form></u>:

1. **<u>Action</u>**: This attribute specifies the URL(address) of the server are any other resource where the form data has to be submit.
2. **<u>Method</u>**: This attribute defines how to submit the form data to the server. The most widely values are ‘get and post’. If the method value is ‘get’ then the

form data will be submitted to the server along with the URL. The form data will be display in the address bar and hence doesn't provide security to the form data.
If the method value is 'post' then the form data will be submit to the server separately form the URL. It will don't display the from data in the address bar. There by provides security to the form data.
Using 'get' we can send limited data only but using 'post' we can sent unlimited data.
The default method value is 'get'.
3. **Name**: This attribute is used to assign a name to a form for identification.

Sub tags of form tag:
<input>: This tag is used to create a input element where the user can either some information.
Attributes of <input>:
1. **Type**: This attribute will be decide the type of input element to be display list.
   Ex: type: text, password, radio, checkbox, submit, reset, hidden (internal all pages displayed), button.
2. **Name**: This attribute is used to assign a name to the input element.
3. **Size**: This attribute specified the length of the input field.
4. **Value**: This attribute is used to specifies the default value are initial value to a input element.
5. **Max length**: This attribute specify the maximum number of characters to be entered the input element.

<Select>: This tag will provide the user a list of options form which the user can select one option.
<Text area>: This tag allows the user to enter the data in multiple lines the size of the text area can be specified by using rows and cols attribute.

Ex:
```
<html>
    <head>
            <title>Registration</title>
    </head>
```

```
        <body bgcolor = “green”>
                <form method = “post” name = “registrationform”>username<input
                type = “text” name = “user” size = “30”/><br/>
                Password<input type = “password” name = “pass” size = “30”/><br/>
                Confirm password<input type = “password” name = “cpass” size =
                “30”/><br/>
                Gender<input type = “radio” name = “sex”/>
                Male<input type = “radio” name = “sex”/>
                Female<input type = “radio” name = “sex”/><br/>
Qualification
                <select>
                        <option>BTech</option>
                        <option>MCA</option>
                        <option>BSc</option>
                <select><br/>
Hobbies
                <input type =”check box”>programming
                <input type =”check box”>browsing
                <input type =”check box”>reading<br/>
Comments
                <textarea rows = “5” cols = “30”>Enter your comments</textarea><br/>
                <input type = “submit” value = “register”/>
                <input type = “reset” value = “clear”/>
                </form>
        </body>
</html>
```

<u>Special character in HTML</u>:

This are those character which have special meaning with the browser all these characters are not available in keyboard.

1. 4 & lt; (4 < 5)
2. 4 & gt; (4 > 5)
3. &reg;
4. &copy;
5. &trade;
6. Parsnd;

7. &amp;
8. &quot; inet solve & quot; “inet solve”
9. &a pos; solutions & apos; “solutions”

Xml: It’s transfer “Extensible Markup Language”. It is a markup language that provide environment to create cross platform compactable file. Cross platform compatibility means it is not specific to any operating system, any H/W, any S/W and etc.
. Xml is a standard given by ‘w3c’. To create tag based data.
. Xml is called as “Mother language” using which we can create other markup languages. Like wml, vml, and mml etc.
. Xml is use to store and describe the data. Data means meaningful and understandable information. Data can be stored in a text file or database or xml file.

Text file: The text file can content formatted and unformatted data.
. It doesn’t show any hierarchy among the data.
. It doesn’t show any relationships b/w the values.
. It doesn’t provide any tools to check or verify the correctness of the data.
. The text files are some time depended on the output.

Database: The database can content formatted data in the form of tables.
. In a database we require a separate of a query language to operate on the data.
. Database has to be used in language scale application.
. That data in a database is specific to the database s/w.

XML

Xml: Xml document contents only formatted data.
. Xml document shows hierarchy among the data.
. Xml document is a cross platform document.
. Xml provides tools like xml engine, xml parser to check verify to correctness of the data in an xml document.
. It doesn't require any query language for manipulating the data.
. Xml can use to store data in small scale application.

HTML	XML
1. Html is use to display and format the data.	1. Xml is use to store and describe the data.
2. Html contains predefined tags.	2. Xml contains user defined tags.
3. Html tags are not case sensitive.	3. Xml tags are case sensitive.
4. Html is error and free language.	4. Xml is not an error free language.
5. Html can be used to design web pages.	5. Xml can be used to transfer data b/w in compatible.
6. Html document can content proper and improper nesting.	6. Xml document can content only proper nesting.
7. Html can be used to apply styles on the data.	7. Xml can be use to structured the data.

Uses of xml:
. Xml can be use to store the data permanently similar to a database.
. Xml can be use to create other "Markup language".
. Xml can be use to create transfer data b/w incompatible types.
. Xml can be use to configure the resources in a web applications.

Ex: web.xml

. Xml document provides some instructions for installing the software.

Xml document preparation rules:

1. Xml document must contain a root tag.
2. Every xml tag must contain a starting tag and closing tag.
3. Every xml tag can (optional) contain either child tags and attributes, child tags or attributes.
4. If a tag contains attribute then its value must be specified enclosed either in single codes (‘’) or the double codes (““).
5. Xml tags are case sensitive (either lower/upper).
6. Xml document must be properly nested.
7. Xml tags can be empty.

Element naming rules:

1. An element naming can be combination of alphabets, digits and special characters.
2. The element name must not begin with a digit or a punctuation characters.

   Ex: (\,*, # ….etc)
3. The element name must be not beginning with xml.
4. There is no restriction on the length of the element name. If it is contains multiple words then separate them by underscore (_).

   Ex: <NAME_OF_THE_STUDENT>

Well formed xml document:

1. Xml document is said to be well formed if it satisfies both xml document preparation rules and elements naming rules.
2. We can verify weather an xml document is well formed or not by using either xml engine or xml parser.
3. To verify weather a xml document is well formed or not by using xml engine then just open the xml document. In a browser, if the xml document is displayed as it is on the browser then it indicates that the xml document is well formed other wise not well formed.
4. Every browser by default contains xml engine as a inbuilt program.

There are no reasons when to use a child tag or when to use attributes. It is recommended to presser child tags instead of attributes.

Reasons for avoiding attributes:

1. Attributes doesn't show any hierarchy.
2. Attributes doesn't show any relationships.
3. Attributes can't contain multiple values.
4. Attributes can't expand in feature.
5. Reading the attribute values while processing the xml document is a difficult task.

   To create the rules for an xml document we use either xml DTD or XML SCHEMA.

   XML DTD or XML SCHEMA will allow us to create the following set of rules.

1. The tag name to be used.
2. The attribute name to be used.
3. The number of occurrences a tag must be available.
4. The type of data the tag or an attribute can contain.
5. The root tag of the xml document.

Valid xml document:

Xml document is said to be a valid xml document if it satisfies xml document preparation rules, element naming rules and xml DTD or XML SCHEMA based document preparation rules.
Every valid xml document is a well formed xml document.

XML DTD:

It stands for xml document type definition. It is used to create rules for writing the xml document. It is provides legal building blocks using which we can define a structure for the xml document.

The building blocks are:

1. **Element**: This is the most important building block. This building block is used to create tags. Which can contain some text, or it can contain some other element or it can be empty.

2. **Attribute**: This building block is used to create attributes which are used to provide additional information about the element. The attributes must be the specified in the staring tags and they come in name value pairs, the attribute value must be enclosed in single codes (‘’) or double codes (“”).
3. **Entity**: this building block is used to create special characters that are to be used in an xml document.
4. **PcDATA (parsed character data)**: This data will be parsed by the parser and they can expand the entities.
5. **CDATA**: its stands for ‘character data’. This data will not be parsed by the parser and it can’t expand the entities.

Declaring an element:

1. Declaring an element which can be empty.

   Syn :<! ELEMENT element-name EMPTY>

   Ex :<! ELEMENT address EMPTY>
2. Declaring an element which contains PCDATA.

   Syn: <! ELEMENT element-name (# PCDATA)>

   Ex: <! ELEMENT roll no (#PCDATA)>
3. Declaring an element which can contain any data

   Syn: <! ELEMENT element-name ANY>

   EX: <! ELEMENT name ANY>
4. Declaring an element which contains sequence of child elements.

   Syn: <! ELEMENT element-name (child-name1, child-name2….)>

   Ex: <! ELEMENT student (roll no, name, mobile no, add)>
5. Declaring an element which must occur exactly one time.

   Syn: <! ELEMENT element-name (child-name)>

   Ex: <! ELEMENT student (roll no)>
6. Declaring an element which can occur zero or more times.

   Sys: <! ELEMENT element-name (child-name*)>

   Ex: <! ELEMENT student (mobile no*)>
7. Declaring an element which must occur at least one time (one or more times).

   Sys: <! ELEMENT element-name (child-name+)>

   Ex: <! ELEMENT student (mobile no+>
8. Declaring an element which must occur either 0 or 1 time.

Sys: <! ELEMENT element-name (child-name?)>
Ex: <! ELEMENT student (address?)>

Declaring attributes:

1. Declaring n attribute with a default value.
   Sys :<! ATTLIST element-name attribute-name attribute-type “default”>
   Ex :<! ATTLIST student course cDATA “xml”>
2. Declaring an attribute which is mandatory.
   Sys :<! ATTLIST element-name attribute-name attribute-type #REAUIDED>
   Ex :<! ATTLIST student course cDATA #REAUIDED>
3. Declaring an attribute which is optional.
   Sys :<! ATTLIST element-name attribute-name attribute-type #IMPLIED>
   Ex :<! ATTLIST student course cDATA #IMPLIED>
4. Declaring an attribute whose value is fixed.
   Sys :<! ATTLIST element-name attribute-name attribute-type #FIXED “value”>
   Ex :<! ATTLIST student fee cDATA #FIXED “3500”>
5. Declaring an attribute containing an enumerated list.
   Sys :<! ATTLIST element-name attribute-name (en-value1, en-value2….)>
   Ex :<! ATTLIST student branch (cse |it | eee |ese)>

Declaring an entity:

1. Declaring an internal entity.
   Sys :<! ENTITY entity-name entity-value>
2. Declaring an external entity.
   Sys :<! ENTITY entity-name SYSTEM URL>
   Rules to be prepared for writing a xml document which stores student information.

I. Create a tag by the name institute which should be the root tag.
II. Create as student as child tag to institute which must be available at least one time.
III. Create roll no, name, mobile no, address as child tags to student tag.
IV. The roll no and name tags must be present exactly 1 time.
V. The mobile no tag can be available either 0 or 1 or more time.

VI. The address tag must be available either 0 or 1 time.
VII. Create hno, colony and city as child tags to address tag.
VIII. The hno, colony, city must be available exactly 1 time.
IX. Create course as an attribute to student tag as enumerated list of values.
X. Create type as an optional attribute under student tag.
XI. Create country as an attribute under address tag whose value is fixed.

XML PARSER:

To validate a xml document we required a xml parser. Xml parser is a utility tool using which we can check where a xml document well format or not and valid or not. Xml parser can even be used for processing the xml documents.

Ex:

SAX (Simple API XML Process)
DOM (Document Object Model)
Document Object model for Java (JDOM)

IDE:

It's "transfer Integrated Development Environment". It is used to speed of the development of an application. Where reduces the development time.

Ex:

XML SPY
STYLUS STUDIO

Procedure to open XML SPY IDE: Click on start on ⟶ select all program sales Alcove ⟶ Click on xml spy.

Procedure to create DTD rules: Click on file menu ⟶ select new item ⟶ select DTD (Document Type Definition) ⟶ click on OK.

Rules:

1. <!ELEMENT institute (student +)>
2. <! ELEMENT student (rollno, name, mobileno*, address?>
3. <!ELEMENT rollno (#pcDATA)>
4. <!ELEMENT name (#pcDATA)>

5. <!ELEMENT mobileno (#pcDATA)>
6. <!ELEMENT address (hno,colony,city)>
7. <!ELEMENT hno (#pcDATA)>
8. <!ELEMENT colony (#pcDATA)>
9. <!ELEMENT city (#pcDATA)>
10. <!ATTLIST student course (java|.net|oracle|html) “html” >
11. <!ATTLIST student type cDATA #IMPLIED>
12. <!ATTLIST student country cDATA #FIXED “india”>

<u>Procedure to write a XML Document with external DTD rules</u>:

Click on file menu ⟶ select new item ⟶ select XML. (Extensible Markup langue) ⟶ Select DTD, click on OK Browser for DTD file click on OK.

1 <! DOCTYPE institute SYSTEM “D:\studrules.dtd”>
2 <institute>
3 Student course = “html” Type = “inspector”>
4 <rollno>123</rollno>
5 <name>Rams</name>
6 <mobileno>9949139899</mobileno>
7 <mobileno>8887776663</mobileno>
8 <address country = “india”>
9 <hno>2/202</hno>
10 <colony>nawabpet</colony>
11 <city>bond</city>
12 </address>
13 </student>
14 </institute>

Procedure to create a XML document containing internal DTD rules:

Click on file menu

⟶ Select new item
⟶ Select XML (Extensible Markup Language)
⟶ Click on OK
⟶ Click on cancel

```
<! DOCTYPE institute [
<! ELEMENT institute (student +)>
        .
        .
        .{same as previous rules}
        .
        .
<! ATTLIST address country cDATA #FIXED “india”>
]>
<institute>
        <student course = “html” Type = “inspector”>
                .
                .
                .{same as previous rules}
                .
                .
        </student>
</institute>
```

<u>XML SCHEMA</u>:

1. XML SCHEMA is used to define structure of the XML document.
2. XML SCHEMA is a XML based alternative to DTD.
3. XML SCHEMA is successor to XML DTD.
4. XML SCHEMA is written in XML.
5. XML SCHEMA provides more Richer and powerful functionality.
6. XML SCHEMA supports data types and data types.
7. We can create our own user defined data types based on predefined data types.
8. XML SCHEMA supports name spaces.

In XML schema we have two types of elements.

A. Simple element
B. Complex element

1. **<u>Simple element</u>**:

An element is said to be simple if it contains only text. It should not contain some other element or attributes.

Syn:

```
<xs: element name = “xxx” type = “yyy”/>
```

Here xxx represent the name of the element and yyy represent the type of the data.

Ex:

```
<xs: element name = “mobileno” type = “xs: integer”/>
<xs: element name = “xxx” type = “yyy” default = “value”/>
<xs: element name = “xxx” type = “yyy” fixed = “value”/>
```

2. **Complex element**:

   An element is said to be complex if it contains some child element and attributes or attributes.

   Ex:

```
        <xs: element name = “xxx”>
            <xs: complexType>
                <xs: sequence>
                    <xs: element name = “x1” type = “y1”/>
                        <xs: element name = “x2” type = “y2”/>
               </xs: sequence>
          </xs: complexType>
     </xs: element>
```

Here, xxx represents the name of the complex element name x1 and x2 represent name of the child elements any y1 and y2 represents the type of x1 and x2 respectively. Here, x1 and x2 order can’t be changed x1 must be followed by the x2 only.

Order Indicators:

1. **Sequence**: This indicator will indicate that specifying all the child elements is mandatory and the order is fixed.
2. **All**: This indicator will indicate that specifies the child elements are mandatory but order of the elements can be changed.
3. **Choices**: These indicators will indicate that we can use specifying that we can use any one of the child element.

<u>Occurrence indictors</u>:

1. **<u>Minoccurs</u>**: This indicator will specify the minimum no. of times a tag or element must occur.
2. **<u>Maxoccurs</u>**: This indicator will specify the maximum no. of times an element can (optional) occurs.

<u>Attributes</u>:

Syn:

```
<xs: attribute name = “xxx “ Type = “yyy “/>
<xs: attribute name = “course“Type = “xs: string“default = “xml”/>
<xs: attribute name = “course“Type = “xs: string“default =”xml”/>
<xs: attribute name = “course” Type = “xs: string “fixed=”java”/>
<xs: attribute name = “course” Type = “xs: string“
use=”optional/required”/>
```

<u>Restrictions</u>: These restrictions allow the user to define acceptable values in a xml document. The restrictions on xml elements are called as “facets”.

1. **<u>Restrictions on set of values</u>**:

   <u>Ex</u>:

```
<xs: element name=”Marks”>
    <xs: SimpleType>
        <xs: restriction base=”xs:integer”>
            <xs: minInclusive value=”10”/>
                <xs:maxInclusive value=”100”/>
                    </xs:restriction>
                </xs: SimpleType>
        </xs: element>
```

If we use minInclusive and maxInclusive the extremes values will be included into the range.

Instead of minInclusive and maxExclusive we can use minExclusive and maxExclusive where the Extreme values will not be Included.

2. **Restrictions on List of values**:

   Ex:

```
<xs: element name = “branch”>
    <xs: simpleType>
        <xs: restriction base = “xs: string”>
            <xs: enumeration value = “cse”/>
            <xs: enumeration value = “it”/>
            <xs: enumeration value = “mca”/>
            <xs: enumeration value = “ece”/>
        </xs: restriction>
    </xs: simpleType>
</xs: element>
```

3. **Restrictions on series of values**:

```
<xs: element name = “xxx”>
    <xs: simpleType>
        <xs: restriction base = “yyy”>
            <xs: pattern value = “([a-z]+)”/>
        </xs: restriction>
    </xs: simpleType>
</xs: element>
```

The above pattern allow the user to enter the data of length three characters, where every character lowercase alphabet.

Note: Every pair of [] represents one character.

```
<xs: pattern value = “[abcd]/>
```

The above syntax allow to enter a data of length one character where that character can be either ‘a’ or ‘b’ or ‘c’ and ‘d’.

```
<xs: pattern value = “[0-9][0-9][0-9][0-9]”/>
```

The above pattern will allow the user to enter a data of length of data 4 character where each character should be a digit.

```
<xs: pattern value = “([a-z])*”/>
```

The above syntax will allow the user to enter a data of any length but every character should be a lower case alphabet.

```
<xs: pattern value = "([a-z][A-Z])+"/>
```

The above pattern will allow the user to enter a data containing one or more pairs of lower case and the upper case alphabets respectively.

```
<xs: pattern value ="[0-9]{10}"/>
```

The above pattern will allow the user to enter a data of length exactly 10 character where every should be a digit.

4. **Restrictions on length of the data**:

```
<xs: pattern value =" ([a-zA-Z0-9@#$ %])*/>
<xs: minLength value="6"/>
```

The restriction will allow the user to enter a data whose minimum length should be '6' characters.

```
<xs: pattern value = "([a-z A-Z @ # % $])*"/>
<xs: maxLength value = "20">
```

This restriction will allow the user to enter a data whose maximum length can be up to '20' characters.

```
<xs: length value = "10"/>
```

This restriction will allow the user to enter data whose length is exactly '10' characters.

Procedure Create a Document Containing Schema Based Rules:

Click on file menu ⟶ select new ⟶ select xsd (w3c XML SCHEMA) click on OK.

1. <xs: schema xmlns: xs="http://www.w3.org/2001/XML Schema">
2. <xs: element name = "institute">
3. <xs: complexType>
4. <xs: sequence>
5. <xs: element name="student" minoccurs = "1" maxoccurs = "60">

```
<xs: complexType>
<xs: sequence>
<xs: element name=“rollno” maxoccurs=“1” minoccurs=”1”>
<xs: simpleType>
<xs: restriction base = “xs: integer”>
<xs: pattern value = “[0-9][0-9][0-9]”>
</xs: restriction>
</xs: simpleType>
</xs: element>
<xs: element name = “name” maxoccurs = “1” minoccurs = “1”>
<xs: simpleType>
<xs: restriction base = “xs: string”/>
<xs: pattern value = “([a-zA-Z])*”/>
<xs: minLength value = “4”/>
<xs: maxlength value = “10”/>
</xs: restriction>
</xs: simpleType>
</xs: element>
<xs: element name = “mobileno” minoccurs = “2” maxoccurs = “5”>
<xs: simpleType>
<xs: restriction base = “xs:integer”>
<xs: pattern value = “[789][0-9]{9}”/>
</xs: restriction>
</xs: simpleType>
</xs: element>
<xs: element name = “address”>
<xs: complexType>
<xs: all>
<xs: element name = “hno” type = “xs: string”/>
<xs: element name = “colony” type = “xs: string”/>
<xs: element name = “city” type = “xs: string”/>
</xs: all>
<xs: attribute name = “country” fixed = “india”/>
</xs: complexType>
</xs: element>
```

```
</xs: sequence>
<xs: attribute name =”course” use = “required”>
<xs: simpleType>
<xs: restriction base = “xs: string”>
<xs: enumeration value = “html”/>
<xs: enumeration value = “xml”/>
<xs: enumeration value = “java”/>
<xs: enumeration value = “oracle”/>
</xs: restriction>
</xs: simpleType>
</xs: attribute>
<xs: attribute name = “type” use =”optional” type = “xs: string” default =
  “regular”/>
</xs: complexType>
</xs: element>
</xs: sequence>
</xs: complexType>
</xs: element>
</xs: schema>
```

Save as rules.xsd

<u>Procedure to write a xml document following schema based rules</u>:

1. Click on file menu → select new item → select xml → click on OK.
2. Select schema → click on OK → browse for.xsd file & click on OK.

```
<institute xmlns: xsi = “http/www.w3.org/2000/xml schema-instance”
        Xsi : noName space schemaLocation = “D:\rules.xsd”>
        .
        .
        .
</institute>
```

JAVA SCRIPT

Java script:

1. Java script is the most popular scripting language used one the internet. It's works all the browsers. Like internet Explorer, Firefox, Chrome etc.
2. Java script is designed to interact with the html document.
3. Java script code can be embedded into the html documents.
4. Java script is an interpreted language.

Note:

Java and java script are two different language designed for two different purpose.

Uses of java script code:

1. Java script can be used to place dynamic content in the html document.
2. Java script can be used to read and modify the content of an html element.
3. Java script can be used to validate the form data before it is submitted to the server. This is called as "client side validations".
4. Java script can be used to detect the user's browser at runtime.
5. Java script can be used to store and retrieve the information from the clients machine.
6. Java script can react to events.
7. To write the java script code in the html document. We have to use <script>. The java script code has to place in b/w <script> only.
8. To specify the <script> contains java script code. We take help of type attribute must be <script type = "text/JavaScript">
9. The <script> can be placed either in head section or body section or both sections.

Ex:

```
<html>
    <head>
        <title>java script</title>
    </head>
    <body>
        <script type = "text/java script">
```

```
                    document.write(“welcome to my web site”);
                    </script>
                </body>
            </html>
```

The document.write is the standard JavaScript command to display a message on the browser. The data which is specified in side the write function will be displayed as it is on the browser.

Java script statements:
The statements are the commands to the browser. The purpose of the statement is to tell a browser what to do. The java script statements can be a terminated by a (;) semi colon.

Java script comments:
The comments are used to explain the code or make the code more readable the comments are non-executable statements ignored by the browser. They are two types of commands.

1. Single line comment	//this is a single line comment
2. Multi line comments	/*……….*/

Java script functions:
When a java script code is written inside a <script> it will be executed immediately. When the html document is loaded on to the browser but some times we want to do this we need to take this support of a function.

Syn:

```
Function function-name (Value1, Value2…) {
        Statements to be executed
}
```

Ex:

```
<html>
    <head>
        <script type = “text/JavaScript”>
            Function display ()
                {
                    document.writen (“good afternoon”);
               }
          </script>
       </head>
       <body onload = “didplay ()”>
       </body>
   </html>
```

Java script variables:
The variables can be used to hold a value or and expression. The variable name must begin with either an alphabet or an underscore symbol.
Java script doesn’t contain any data types. To declare a variable in the java script we are use ‘Var’ (variable).
A variable of java script can contain any kind of data and the string data must be enclose in (“”) double codes.

Ex:

```
<head>
     <script type = “text/java script”>
        var x = 10;
        var y = 20.50;
        var z = x + y;
        var name = “java”;
     document.write (“sum of =” + w);
   </script>
</head>
```

Java script operators:

1. **Arithmetic operators**:
   These operators are used to perform then mathematical calculations. The various Arithmetic operators are Addition (+), Subtraction (-), Multiplication (*), Division (/), Modules (%), Increment (++), and Decrement (--)

2. **Relational operators**:
   These operators are used for comparing the values. These

operators can also be called as comparison operators. The various relational operators are less than, less than are equals, grater than, grater than are equal, equals, not equals (<, <=, >, >=, ==, !=).

3. **Logical operators**:
   These operators are used to combine the conditions or used to compliment the result. The various logical operators are AND (&), OR (|), and NOT (!).

4. **Conditional operators**:
   This operator is also called as ternary operator and it is used to perform some operators based on a condition. The conditional operator is? and ; (question mark, semicolon).

   Ex:

   ```
   first = (x>y)? x:y;
   if (x>y)
       first = x;
   else
       first = y;
   ```

5. **Assignment operators**:
   This operator is used to assign a value to a variable. The various assignment operator are assignment (=), compound assignment operators (+=, -=, *=, /=, %=).

Popup boxes:

1. **Alert box**:
   This box is used to display a message on the browser which must be compulsory read by user.

   Syn:

   ```
   alert (message);
   ```

   The alert box will not allow the user provide for the until the user read the message click on OK button.

2. **Prompt box**:

This box allows the user to enter some of the information.

Syn:

prompt (message);

Syn:

Prompt (message, default-value)

The prompt be contains two buttons OK and CANCEL if the user can click on OK button then it will read the available in the prompt box. If the user can click on CANCEL button then it will display null value for example.

```
<html>
    <head>
        <script type = “text/JavaScript”>
            var n = prompt(“enter a number”, “4567”);
            document.write(“n=” + n);
        </script>
    </head>
    <body>
    </body>
</html>
```

3. **Confirm box**:

This box is used to take the confirmation from the user.

Syn:

confirm(message);

The confirm box contains two buttons OK and CANCEL. When the user click on OK button it return is true. When the use click on CANCEL button it returns false.

Ex:

```
<html>
    <head>
        <title>pop up boxes</title>
    </head>
```

```
                <body>
              <Script type = “text/JavaScript”>
                   var status = confirm(“do you want to sleep”);
                   document.write(status);
              </script>
           </body>
    </html>
```

<u>Conditional statements</u>:

These statements are used to execute a group of statements based on a condition.

1. **<u>If statement</u>**:

   This statement will execute a group of statements when a condition is true.

   Syn:

```
      If (condition)
          {
               Statements to be executed
          }
   <html>
         <head>
              <title>conditional statement</title>
         </head>
         <body>
              <script type = “text/JavaScript”>
              if(1 < 2)
                   {
                        document.write(“Hello”);
                   }
              </script>
           </body>
   </html>
```

2. **<u>if-else statement</u>**:

If the condition is true than if block is executed and when condition is false else block is executed.

Syntax:

```
            if (condition)
                {
                    Statement1
                }
    else
            {
                    Statement2
            }
```

Ex:

```
<html>
    <head>
        <title>conditional statement</title>
    </head>
    <body>
        <script type = “text/JavaScript”>
            if(1 – 2)
              {
                  document.write(“1 is smaller”);
              }
          else
              {
                  document.write(“1 is bigger”);
              }
        </script>
    </body>
</head>
```

3. **switch statement**:
   This statement is used to execute an option from a group of option from a group of options that are available.

Syn:

```
Switch (expression)
        {
                case value1 : statement1;
                                break;
                case value2 : statement2;
                                break;
                case value3 : statement3;
                                break;
                case value4 : statement4;
                                break;
                default : default statement;
        }
```

Ex:

```
<html>
     <head>
          <title>conditional statement</title>
     </head>
           <body>
                <script type = “text/JavaScript”>
                var x = parseInt (prompt(“Enter a number”));
        switch(x)
                {
                     case1 : document.write(“one”);
                                break;
                     case2 : document.write(“two”);
                                break;
                     case3 : document.write(“three”);
                                break;
                     case4 : document.write(“four”);
                                break;
                     default: document.write(“wrong choice”);
                }
                </script>
        </body>
```

```
</html>
```

Iterating statements: These statements are used to execute a group of statements multiple times.

1. **For loop**:
   This statement has to be used when we known the exact number of iterations.

   Syn:

```
for(initialization; condition; increment/decrement)
        {
             Statements to be executed
        }
```

   Ex:

```
<html>
     <head>
          <title>Iterating statements</title>
     </head>
     <body>
          <script type = “text/JavaScript”>
          var fact = 1;
          for(x=1; x<=10; x++)
              {
                  fact = fact * x;
                  document.write(x+”!=” +fact+ “</br>”);
              }
          </script>
       </body>
   </html>
```

2. **While loop**:
   This loop must be used when we do not know the exact no. of iterations.

   Syn:

```
while(condition)
   {
        Statements to be executed
   }
```

Ex:

```
<html>
      <head>
            <title>iterating statement</title>
      </head>
      <body>
            <script type = “text/JavaScript”>
            var n = parseInt(prompt(“Enter a table number”));
            var x = parseInt(prompt(“Enter an other number”));
            var i = 1;
            while(i<=x)
               {
                     document.write(n + “*” +i + “=” +(n*i) +”<br/>”);
                     i++;
               }
            </script>
       </body>
  </html>
```

3. **<u>Do…..while loop</u>**:
   This loop has to be used when we do not know the exact number of iterations.

Syn:

```
do
     {
          Statements to be executed
     } while (condition);
```

<u>Difference b/w while and do….while loop?</u>:
In a while loop if the condition is false for the 1st time then statements will be execute “zero times”.
In a do….while loop if the condition is false for the 1st time statement will be execute for “one time”.

Ex:

```
<html>
```

```
            <head>
                  <title>iteration statements</title>
            </head>
            <body>
                 <script type = “text/JavaScript”>
                       n = praseInt(prompt(“Enter a table number”));
                       x = parseInt(prompt(“Enter an other number”));
                       var i = 1;
                          do
                               {
                                  document.write(n + “*” +i+ “=” +(n*i) + “<br/>”);
                                        i++;
                               }while(i<=x);
                 </script>
             </body>
        </html>
```

<u>JavaScript objects</u>:

JavaScript can be considered as an object oriented language, it allows the user to use predefined objects or create user defined objects.
An object is a kind of data which contains properties and methods. Properties are used to hold the values of an object and the methods represent the Actions that are performed by the object.

1. **<u>String object</u>**:
   If the user encloses any data in “double codes” than it is considered as string object.

   Ex:

```
<html>
    <head>
          <title>JavaScript objects</title>
    </head>
    <body>
          <script type = “text/JavaScript”>
                   var type = “javascript”;
```

```
                    document.write(text +”<br/>”);
                    document.write(text.lenght +”<br/>”);
                    document.write(text.toUpperCase() +”<br/>”);
                    document.write(text.toLowerCase() +”<br/>”);
                    document.write(text.indexOf(“a”) +”<br/>”);
                    document.write(text.lastIndexOf(“a”) +”<br/>”);
                    document.write(text.replace(“java”,”VB”) +”<br/>”);
                    document.write(text.fontcolor(“red”) +”<br/>”);
            </script>
        </body>
    </html>
```

2. **Date object**:

   This object is used to perform operations on data and time.

   Ex:

```
        <html>
            <head>
                <title>date object</title>
            </head>
            <body>
                <script type = “text/JavaScript”>
                var date = new Date();
                    document.write(date + “<br/>”);
                    document.write(date.getDate() + “<br/>”);
                    document.write(date.getMonth() + “<br/>”);
                    document.write(date.getfullYear() + “<br/>”);
                    document.write(date.getHouse() + “<br/>”);
                    document.write(date.getMinutes() + “<br/>”);
                    document.write(date.getSeconds() + “<br/>”);
                </script>
            </body>
        </html>
```

3. **Boolean object**: This object represents either true or false. Based on the value we pass to the boolean object.

   Ex:

```
        <html>
        <head>
            <title>Boolean object</title>
         </head>
         </body>
            <script type = “text/javascript”>
              var b1 = new Boolean();
              var b2 = new Boolean(0);
              var b3 = new Boolean(1);
              var b4 = new Boolean(true);
              var b5 = new Boolean(false);
              var b6 = new Boolean(“ “);
              var b7 = new Boolean(“html”);
              var b8 = new Boolean(null);
              document.write(b1 + “<br/>”);
              document.write(b2 + “<br/>”);
              document.write(b3 + “<br/>”);
              document.write(b4+ “<br/>”);
              document.write(b5+ “<br/>”);
              document.write(b6+ “<br/>”);
              document.write(b7+ “<br/>”);
              document.write(b8+ “<br/>”);
           </script>
       </body>
    </html>
```

4. **Array object**:
   This object is used to value store multiple values into a single variable.
   Ex:

```
<html>
    <head>
        <title>Array Object</title>
    </head>
    <body>
        <script type = “text/javascript”>
            var names = new Array();
```

```
                    names[0] = “john”;
                    names[1] = “Khan”;
                    names[2] = “Zhan”;
                    names[3] = “Mallikarjuna”;
                    names[4]=  “Suneel”;
                document.write(names + “<br/>”);
                for(i=0; i<names.length;i++);
                        {
                            Document.write(names[i] +”<br/>”);
                        }
                document.write(names.reverse() +”<br/>”);
                document.write(names.sort() +”<br/>”);
            </script>
        </body>
    </html>
```

5. **Math object**: This object is used to perform mathematical operations.

    Ex:

```
<html>
    <head>
        <title>Math Object</title>
    </head>
    <body>
        <script type = “text/javascript”>
                document.write(Math.min(4,6) + “<br/>”);
                document.write(Math.max(4,6) + “<br/>”);
                document.write(Math.floor(4.5) + “<br/>”);
                document.write(Math.ceil(4.5) + “<br/>”);
                document.write(Math.round(4.5) + “<br/>”);
                document.write(Math.sqrt(a) + “<br/>”);
                document.write(Math.random() + “<br/>”);
                document.write(Math.PI  + “<br/>”);
        </script>
    </body>
</html>
```

Javascript events:

Events are the actions that can be identified by javascript the events will be generated automatically based on the user operations to perform any operation when an event is generated. We generally take the help of functions.

1. **onclick**: This event will be generated when the user performs simple click.
2. **ondblclick**: This event will be generated when the user performs double click.
3. **onkeydown**: This event will be generated when the user clicks on a key and not at released.
4. **onkeyup**: This event will be generated when the key is released.
5. **onkeypress**: This event will be generated when the key is pressed and released.
6. **onmouse down**: This event will be generated when the user click on the mouse left button and not at released.
7. **onmouseup**: This event will be generated when the mouse click is released.
8. **onmousemove**: This event will be generated when there is a change in the move location.
9. **onmouseover**: This event will be generated when the mouse pointer is placed on top of a text or image and etc.
10. **onmouseout**: This event will be generated when the mouse pointer is removed from the top of a text or image and etc.
11. **onload**: This event will be generated when the html document is loaded on to the browser.
12. **onunload**: The event will be generated when the html document is removed from the browser.
13. **onfocus**: This event will be generated when the html element has the control.
14. **onsubmit**: This event will be generated when the user clicks on the submit button.
15. **onreset**: This event will be generated when the user clicks on the reset button.
16. **onchange**: This event will be generated when there is a change in the value of the HTML element an change.

17. **onerror**: This event will be generated when an error occurs in the document.
18. **onresize**: This event will be generated when there is a change in the size of the frame.

```
//javascript
        <html>
             <head>
                  <title>Events</title>
                        <script type = “text/javascript”>
                              Function change()
                                     {
                                         var name = myform.user.value;
                                         var col = myform.color.value;
                                         document.bgcolor = col;
                                         alert(name + “has changed the background
                                         color to” + col);
                                     }
                          </script>
              </head>
              <body>
                 <form name = “myform”>
                 UserName<input type = “text” size = “25” name = “user”/> <br/>
                 color<input type = “text” size = “25” name = “color”/><br/>
                <input type = “button” value = “change” onclick = “change()”/>
                </form>
             </body>
        </html>
```

```
//client side validations
        <html>
             <head>
                <title>Events</title>
                    <script type = “text/javascript”>
                       function validate(loginform)
```

```
          {
                var name = loginform.user.value;
                var pwd = loginform.pass.value;
                var data = /\w/;
                if(name.length==0)
                    {
                        alert(“ Enter your user name”);
                        loginform.user.focus();
                        return false;
                    }
         else if((name.length<6) || (name.length>15))
                {
                    alert(“user name length is not valid”);
                    loginform.user.focus();
                    return false;
                }
         else if(data.test(name))
                {
                    alert(“user name character are not valid”);
                    loginform.user.focus();
                    return false;
                }
            if(pwd.length==0)
                {
                    alert(“Enter your password”);
                    loginform.pass.focus();
                    return false;
                }
             }
     </script>
 </head>
<body bgcolor = “cyan”>
         <center>
<h1>Loginform</h1>
  <form name = “loginform” onsubmit = “return validate(this)”>
```

```
            UserName<input type = “text” name = “user” size = “30”/><br/>
            password<input type = “password” name = “pass” size = “30”/>  <br/>
            <input type = “submit” value = “Login”/>
            </from>
          </center>
       </body>
     </html>
```

We can write the javascript code internal to the html document or external to the html document.

```
     function validate(loginform)
            {
               …….
               …………
               ……………..  //
               …………………
               ………………………
            }
```

Save as validation.js

To include the external javascript into a html document. We take help of “src” attribute of <script>

```
     <script type = “text/javascript” src = “validation.js>
     </script>
```

CSS

CSS: its stands for “cascading style sheets”. CSS is used to define styles how to display the html element. The styles how to display the html element.

1. Inline styles
2. Internal styles
3. External styles

1. **Inline styles**: If the styles are specified inside the tag then styles are called as inline styles. These styles are applied to only that tag in which they are specified.

   Ex:

   ```
   <h1 style = “color:blue; text-align:center;”> XYZ ltd </h1>
   <h1 style = “color:red; font-family: arial;”> ABC LTD </h1>
   <p style = “color:blue; font-style: italic; margin-left: 45t;”> this is a para
   </p>
   <p style = “font-size: 22pt;”> second para</p>
   <hr style = “color:green;”/>
   ```

2. **Internal styles**: If the styles are specified with in the html document by using <style> then those styles are called as internal styles. The internal styles will be applied to all the tags available in that html document.

   Ex:

   ```
   <html>
       <head>
           <title>CSS</title>
       <style type = “text/css”>
           a:hover
             {
                color:red;
                font-size:200%;
             }
   ```

```
            P
              {
                 color:green;
                 font-family:arial;
              }
            h1
               {
                 color:brown;
                 font-size:22pt;
               }
            body
                 {
                     backgroung-color:cyan;
                 }
        </style>
    </head>
    <body>
    </body>
</html>
```

3. **External Style**:
                              If the styles are specified out side the html document then those styles are called as external styles. The styles available externally can be applied to all the tags available in multiple html documents.
   The external styles stored out side the html document. Must be saved with any name (.) dot having.css as the extension.

   Ex:

```
      P
        {
          color:green;
          font-family:arial;
        }
      h1
        {
```

Styles.css

```
        color:brown;
        font-size:22pt;
    }
```

To use external style sheets within the html document we take help of <link>. The <link> must be specified in the need section of the html document.

Syn:

```
<link type = “text/css” rel = “stylesheet” href = “styles.css”/>
```

In the html document we can write all the types of styles. But the preference followed by the browser in displaying the html elements is

1. Inline styles
2. Internal styles
3. External styles
4. Browser Default styles