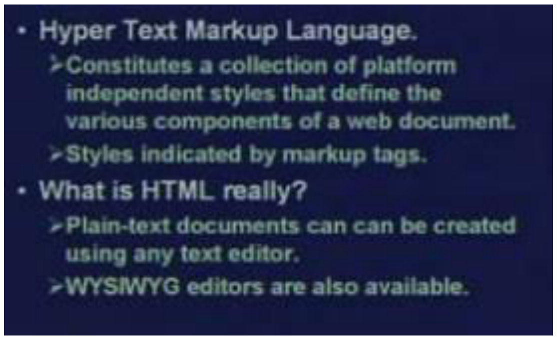
UNIT-02

Hyper Text Markup Language (HTML)

UNIT-02/LECTURE-01

Introduction to Hyper Text Markup Language (HTML)

About HTML: [RGPV/Dec 2013(4)]

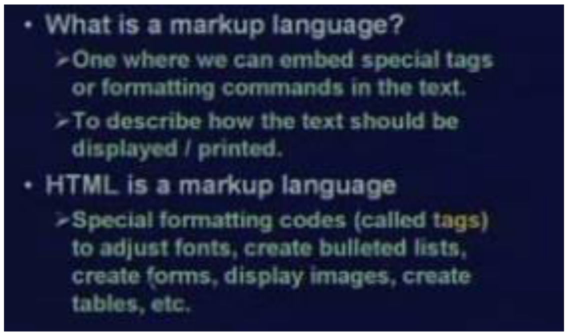


- Hyper Text Markup Language.
  - ➢Constitutes a collection of platform independent styles that define the various components of a web document.
  - ➢Styles indicated by markup tags.
- What is HTML really?
  - ➢Plain-text documents can can be created using any text editor.
  - ➢WYSIWYG editors are also available.

So the first thing is that html the full form is Hyper Text Markup Language. So there are two components to this name. One is hypertext, other is markup. Hypertext is a kind of textual document where you can have links to other documents. In html this kind of links are allowed. So in that sense, html is a hypertext document. Markup, we shall be explaining shortly. This html actually constitutes a collection of platform independent styles. This is called platform independent in the sense that well you can view these files on any browser you want. It can be explorer, it can be konquerer, it can be Mozilla. So in that sense these styles are considered to be platform independent. They are all; they are actually part of some kind of standard. And these styles will define the various components of a web document. And we will actually specify how the document will be displayed on the window of the browser.

Now in html these styles are specified by something called markup tags. This we shall be explaining very shortly. So html essentially is a plain text document. It can be opened and edited using any simple text editor. You can use any simple text editor like notepad, like vi and you can view the doc. You can view the source file of the html you can edit it. But of course there are many more sophisticated editors which are available, these are called what you see is what you get or wysiwyg editors. These are more graphic user friendly interface oriented where whatever you type you see it on the screen in exactly the same form in which it will get displayed on the browser window. In contrast in a text editor you see simple text and those special commands to specify the styles which when you are viewing on the browser window will show up in a particular way depending on what kind of commands you have put in there.

What is markup language?

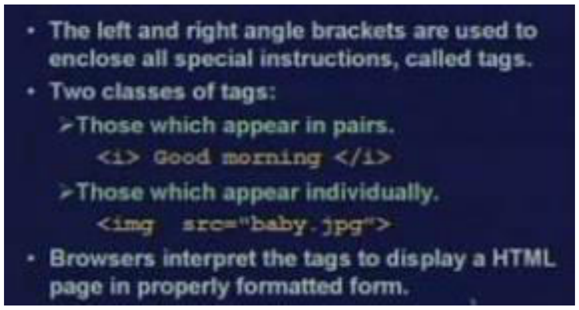


- What is a markup language?
  - One where we can embed special tags or formatting commands in the text.
  - To describe how the text should be displayed / printed.
- HTML is a markup language
  - Special formatting codes (called tags) to adjust fonts, create bulleted lists, create forms, display images, create tables, etc.

Now talking about mark up, well, html is a markup language. Markup language is one where you have the document or the textual information. In addition you can embed some special commands in the document, these are called tags. These tags are nothing but some special formatting commands. Well the idea is that you have some kind of a document with you. You have a document and you can specify some special tags which will tell you, for example, this portion of the document will be displayed in bold face letters and so on. So the mark up tags or the markup language features will allow you to embed these kinds of special commands in the original html document. These tags essentially specify how the text should be displayed when you are seeing it on the browser or printed when you are taking a print out. So as I told you html is a markup language, it provides some special formatting codes.

Again I am repeating, these are called tags in html. Using these tags you can do a number of different things. For instance, you can adjust the fonts in terms of font sizes, font styles, font color; you can create different kinds of lists. You can create font forms, you can include images you can create tables and you can do a lot of other things. So this tags which are there in html provides a very powerful feature that allows you to specify a number of different things with respect to the language. So now there are few things you need to remember. Now in html this set of this tags are fixed or well defined; which means html is a fixed language. Now we shall see later, there is a trend where people are talking about. Well a not a fixed language but rather a language where we can have some extendable or user definable tags. They will be much more powerful or flexible.

HTML tags **: [RGPV/Dec 2013(4)]**

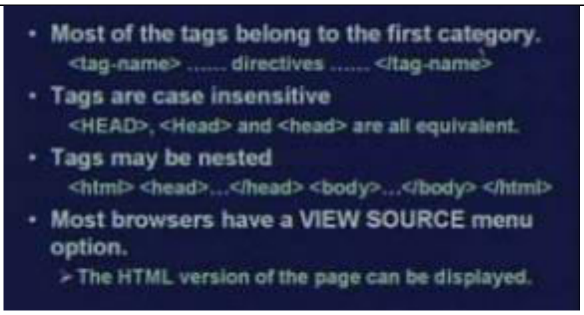


- The left and right angle brackets are used to enclose all special instructions, called tags.
- Two classes of tags:
  - ➢Those which appear in pairs.
    `<i> Good morning </i>`
  - ➢Those which appear individually.
    `<img  src="baby.jpg">`
- Browsers interpret the tags to display a HTML page in properly formatted form.

Now let us see how html tags look like. Now in html, tags are specified by enclosing them within left and right angle brackets. There are two classes of tags in general. One in which tags appear in pairs; for example, this is one example. This specifies the beginning of the tag; this specifies the end of the tag. It is like begin-end structure in a document. In this specific example this angular bracket i means that text which is there in between should be displayed in italics font. And the end tag will contain the same name but it will start with a slash character. So remember the slash character before the name of the tag will specify that the tag definition is ending here. So in text you can have a begin tag and end tag and whatever comes in between will get formatted as part of the definition of that tag.

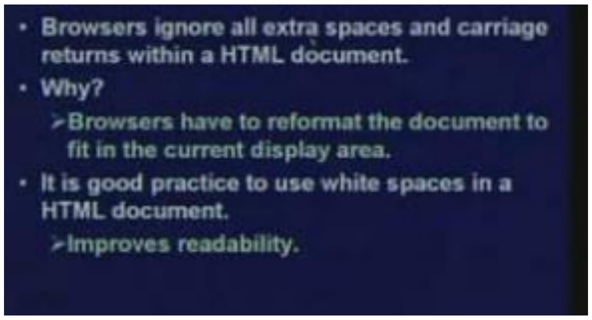
But there are some tags for this end tag is not required for the tag definition appears individually. This is an example. This img is the name of a tag. This is the begin tag symbol. This is the ending angular bracket and between these examples also illustrates that in addition to the tag name img you can have some attributes specified like here. This example tells you that you want to include an image whose, the corresponding file name is baby.jpg. So here src is an attribute which specifies which is the file name for this image. For this you do not need an end tag with which will contain slash img. Browsers have the capability to interpret the tags so as to display them in a properly formatted form on the window of the browser.

Some Points:


- Most of the tags belong to the first category.
  <tag-name> ...... directives ...... </tag-name>
- Tags are case insensitive
  <HEAD>, <Head> and <head> are all equivalent.
- Tags may be nested
  <html> <head>...</head> <body>...</body> </html>
- Most browsers have a VIEW SOURCE menu option.
  - The HTML version of the page can be displayed.

Some points to note is that number one is that most of the tags we use belong to the first category in the sense that it starts with a tag name it finishes with an end tag. But there is a slash character in the beginning and in between there is something. This is can be a text this can be a table whatever which will get formatted according to which tag you are specifying. There is another point. Tags are not case sensitive. They are case insensitive. So it is up to you how do you want to put in your tag. You can put it in all capital you can put only the first letter capital all lower case or any order of lower and upper case letters you can use. Tags can be nested just like you can nest structures in a program. This is one simple example. This is a tag html, this is begin html, this is end html. Now within this begin and end there is a begin head, end head, begin body, end body.

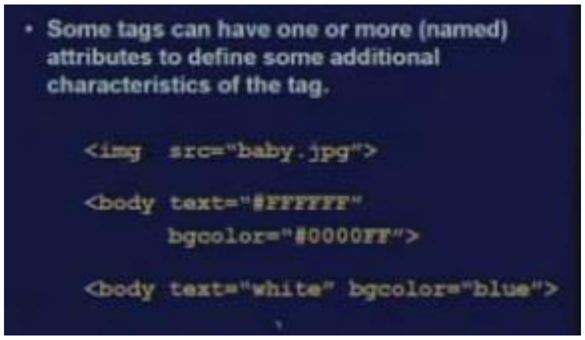
So the rules of nesting are identical to the rules of nesting of loops in a conventional programming language. Nesting should be total there should not be any overlap with respect to two different structures. Because, if you have given a begin command the end tag, command should come after any other begin and end tag command you have put inside it, it cannot overlap. And suppose in a browser you are viewing a page, if you want to look at the corresponding html code, most browsers have a view source option which if you click, then the source or the html version of the page gets displayed on a separate window. You can see the html version of the page which has led to the page which you are seeing in form which is getting displayed.


- Browsers ignore all extra spaces and carriage returns within a HTML document.
- Why?
  - Browsers have to reformat the document to fit in the current display area.
- It is good practice to use white spaces in a HTML document.
  - Improves readability.

Well in html you can add as many extra spaces as you want to make your document readable.

Browsers will ignore all extra spaces it will ignore all carriage returns. You can put as many lines of space in between as you want. But browser will take minimum amount of space. It will ignore all spaces unless you explicitly specify that here is a line break here is a paragraph break and so on. It will ignore all line breaks. This is done primarily because you know that a browser when it is displaying on your computer screen. You can resize the browser; you can make it big; you can make it small. Now depending on the current area of the display window the browser has to reformat the text or whatever as you display within that space.

So if you are hath coding space in your document it becomes difficult for the browser to format it. So spacing is the responsibility of the browser how the spacing in the document will be controlled. So as I have just told spacing is done explicitly done by putting in the paragraph or the line break commands as part of some tags explicitly. So implicitly by putting extra spaces or extra blank lines browser will not give the additional spacing in between whatever is displayed. But as it is said, it is always good practice to use paces to improve the readability of your html source code.



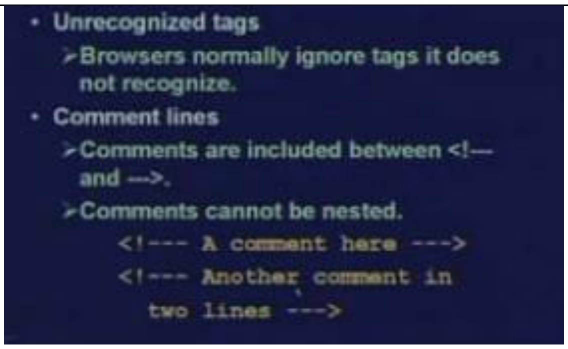
- Some tags can have one or more (named) attributes to define some additional characteristics of the tag.

```
<img  src="baby.jpg">

<body text="#FFFFFF"
      bgcolor="#0000FF">

<body text="white" bgcolor="blue">
```

Some tags as I mentioned, can have one or more attributes. Now each of these attributes have a name. Some of these attributes can also have a value. These attributes specify some additional characteristics of the tag. Some examples are shown here. This first example I have already mentioned earlier. This img is the tag name. This is the optional attribute. This src is the name of the attribute. It means that what is the name of the file equal to baby.jpg is the value of the attribute. Similarly body is another tag where these are the two attributes text and bg color. Text specifies that what should be the color of the text, that will get displayed and bgcolor will tell you what should be the color of the background. These are the two things you can specify. Just instead of specifying color like this. This will just see what this means? This fff and this 0 0 0? You can also specify colors by specifying names like text equal to white bg color equal to blue. These are example of attributes of the tags.



- Unrecognized tags
  - Browsers normally ignore tags it does not recognize.
- Comment lines
  - Comments are included between <!— and —>.
  - Comments cannot be nested.

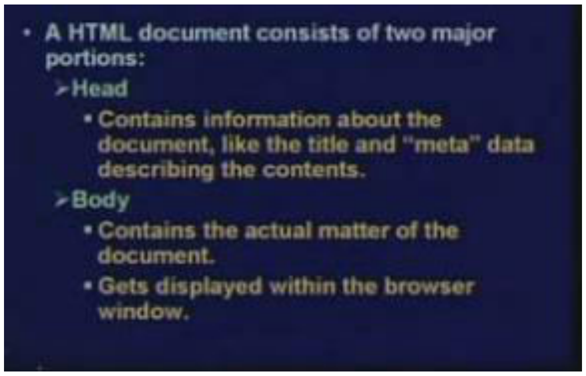
```
<!--- A comment here --->
<!--- Another comment in
  two lines --->
```

Now if some tags are unrecognized, for example if you have misspelled a tag, browsers normally ignore those tags or they will displays just like it were a normal text. It will not do any formatting with respect to that. It depends on the browser exactly what it does. You can include comments lines which will be ignored by the browser. When it gets displayed and comment lines begin with this angular bracket exclamation and three dash. And it will end with three dash followed by closing angular bracket. So comments can lye entirely on a single line or it can be broken across as many lines as you want. So it will be starting with this. It will be finishing with this. So it is good practice to include comments in html source as to increase the readability, what you have done in the different sections of the document

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	Describe the structure of an HTML document.	Dec.2013	4
Q.2	What is the difference between tag and attributes? Explain with an example.	Dec.2013	4

UNIT-02/LECTURE-02

HTML elements



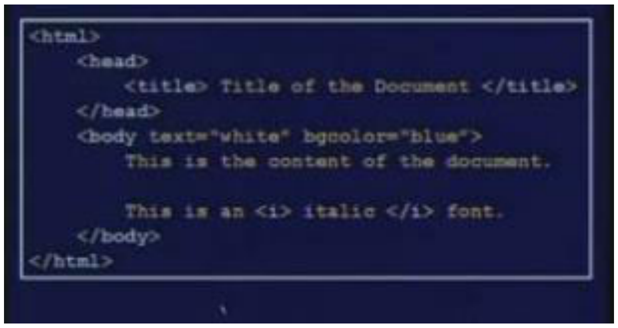
- A HTML document consists of two major portions:
  - Head
    - Contains information about the document, like the title and "meta" data describing the contents.
  - Body
    - Contains the actual matter of the document.
    - Gets displayed within the browser window.

Now talking of the structure of an html document, an html document consists of two major portions: the head, the body. Now in the head section you specify some information about the document. This we shall see some of the information today and some we shall see later. For example, you can specify what will be the title of the document. The title of the document typically gets displayed on the top title bar on the browser when the browser window comes the top title bar displays the title of the document. So whatever you specify as title will come on the top. There is some additional Meta data you can also specify this we shall talk about later. Meta data actually specify some information about the contents.

Like for example some of the key words that is that your document contains. These are sometimes used by the search engines in order to provide indexing services. For example if you are searching Google for a page you give a key word that means how to locate the page. So the periodically, Google look for pages, they look for Meta tags in the header. So if they find, they update their data indexing database. In addition to head there is a body. Now in the body portion the actual matter of the document resides and it is the body which gets displayed within the browser window. The head is something which specifies the title and some other information. Body is something which gets actually displayed in the window.

we dont take any liability for the notes correctness.

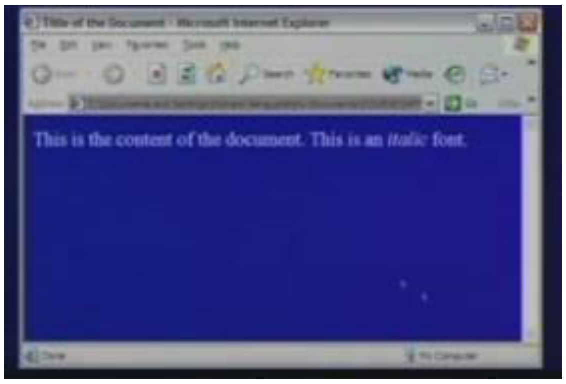
A simple HTML Document



```
<html>
    <head>
        <title> Title of the Document </title>
    </head>
    <body text="white" bgcolor="blue">
        This is the content of the document.

        This is an <i> italic </i> font.
    </body>
</html>
```

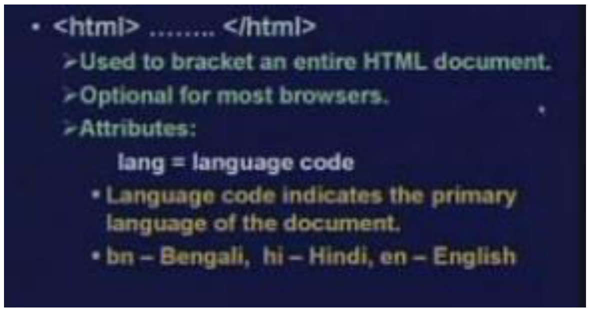
This is a very small example. This is a very rudimentary and a simple html document. You see this starts with a begin html, it ends with an end html. Now in most browsers, this begins and end htmls are optional. This you can also omit. You can see there is a head section, there is a body section. In the head section you have this begin head and end head. And within the head we have nested only one tag the title tag. Now within the title tag we have specified the title of the document. The name I have given as just the title of the document. Now in the body the first you have specified in the begin body tag. We have added some attribute, specifying that our text color will be white and our background color will be blue and the actual body of the document is this. This is the content of the document. This is some text which will come, this is an italic font you see there is a begin italic and an end italic. So the intermediate word is italic, this should come in the italic font. Now this particular html document when it gets displayed on the browser it will look something like this.



This is the content of the document. This is an *italic* font.

This is the content of the document. This is an italic font. You see that the italic is coming in the italic font and the background is blue and the foreground is white.

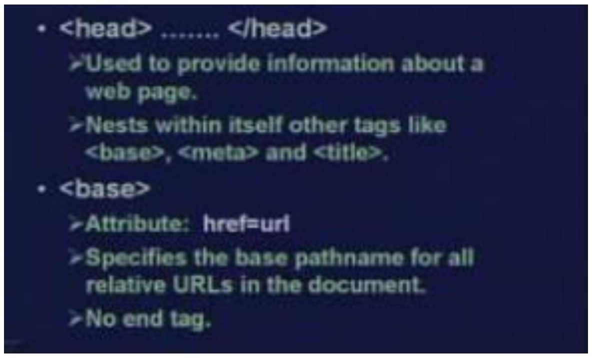
Structural HTML tags



- <html> …….. </html>
  - Used to bracket an entire HTML document.
  - Optional for most browsers.
  - Attributes:
    - lang = language code
      - Language code indicates the primary language of the document.
      - bn – Bengali, hi – Hindi, en – English

So here we first look at the different html tags with respect to their functionality. First we talk about the tags which define the structure of the document. These are the so called structural html tags. So first tell us look at this. Structure html tag, the most important one is I told you html and the html tags they are used to bracket or specify the boundaries of an entire html document. As we have seen in that example, that the first statement in the html file is the begin html. The last statement is the end html. That is the boundary or the beginning or end of the entire document. This as I mentioned, html tags are optional for most browsers. If you do not specify anything, it will be assumed that html begins and end is present at the beginning and the end.

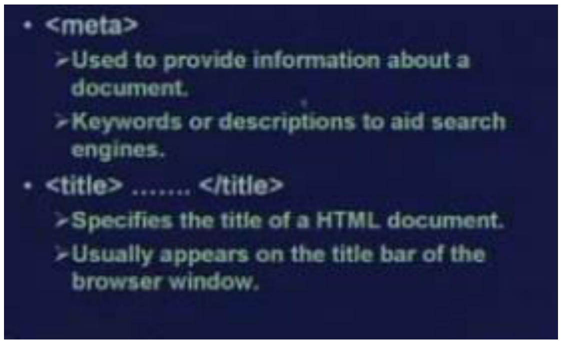
Well in the html begin html command you can have some attributes. For example, if it is a multilingual document like means you are specifying a document in a different language, other than English. Then using this lang l a n g attribute will specify the language code. This language code can be any international language like for instance bn will mean Bengali, hi will mean Hindi and the default en will mean English. So if a language code is specified then the browser will know which font it has to use to display the document in a proper way. So we are not going into the detail of this multilingual document and processing and display right now. But browsers have a facility or this html language have the facility to specify this kind of language.



- <head> ……. </head>
  - Used to provide information about a web page.
  - Nests within itself other tags like <base>, <meta> and <title>.
- <base>
  - Attribute: href=url
  - Specifies the base pathname for all relative URLs in the document.
  - No end tag.

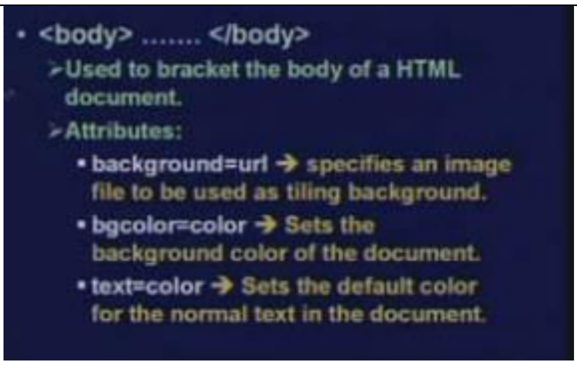
Now within html document as it seem you can have a begin head end head tag. This head is used to provide information about a webpage. There are basically three things which can be nested within head. See head defines a portion of the document begin head and end head. Now the within or between this begin and end heads you can specify a number of other tags. Now in fact it is possible to have three different tags base, meta and title. Let us try to see what these are. Base is a tag where you can specify an attribute, href is equal to you can specify some URL. Like you have something like this, say you can have base, then you can have href equal to, you can have docs slash something like this, which will mean is that you are specifying a base URL out here.

And in the document whatever file name or path name you specify that will with respect to this base URL you have specified. So in all the documents you need not specify doc slash. Base URL allows you to have a shortcut in the path name for all subsequent references. If all the documents you are referencing as part of your html page they can be images, they can be other documents links to other documents. If they are all located under the same directory structure you can specify, the path in the base and when you actually referring you need not have to specify the whole path every time, just specify the file name which is there within the base URL. This base URL does not need any end tag.



- <meta>
  - ➢Used to provide information about a document.
  - ➢Keywords or descriptions to aid search engines.
- <title> ……. </title>
  - ➢Specifies the title of a HTML document.
  - ➢Usually appears on the title bar of the browser window.

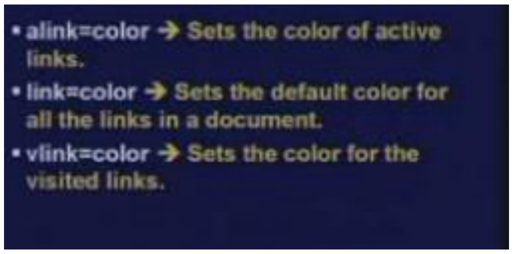
Meta basically provides information about a document. Typically keywords or some description about the document which helps the search engines in indexing the pages so that the pages can be linked with respect to the search engines. Well, use of the title tag we have already seen. The title tag will specify the title of the document. And the title of the document usually appears on the title bar on the top; at the top of the browser window. These are the so called structural tags.



- <body> ……. </body>
  - Used to bracket the body of a HTML document.
  - Attributes:
    - background=url → specifies an image file to be used as tiling background.
    - bgcolor=color → Sets the background color of the document.
    - text=color → Sets the default color for the normal text in the document.

And there is another structural tag which is the actual body of the document. So under this body structure the actual displayable information will be stored or will be specified; the part which will be actually visible in the browser window. Now within the body part you are actually bracketing the so called body of the html document. Now in the body part, there are a number of different attributes you can specify. Well here a couple of this we have already seen. For example we have seen the text attribute, text equal to color. We shall see how we can specify color in different ways. But we can specify a color to the text. If this attribute is present, this will set the default color for the normal text in the document. So if I write text equal to red, then all my text will be displayed in the color red in the window unless I explicitly change the color somewhere else.

Similarly bgcolor attribute this also specifies a color. This will set the background color of the document. Background color means the background of the display window will be set to this color. If I specify bgcolor is equal to blue, then my background will become blue. Sometimes we want to or means we like to have an image to be used as a background and not a fixed color. In that case we use the attribute background. Background equal to some URL or path name, this will specify an image file. Background attribute will specify an image file which will be used as tilling background. So in the window that image will come as the background and the text will be displayed on it. Sometimes we want to have this. Now, in addition to these three attributes, there are other attributes also.



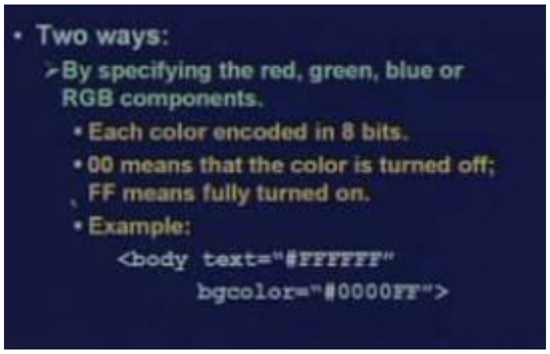
- alink=color → Sets the color of active links.
- link=color → Sets the default color for all the links in a document.
- vlink=color → Sets the color for the visited links.

Well you might be aware that an html document contains links to other documents. There may be some portions which are clickable if you click on the mouse you go to some other document.

These are called hyperlinks. Now you can specify the color of the hyper links the color in which they will appear. There are actually three parts to it. There is something called alink, simple link and vlink. Well alink is equal to color means you are setting the color of the active link. Active link means when you move the mouse on top of a link and click it. What color that link or that document which points to other document that part of the document what color will it become. That is active link.

And simple link is normally which color will be used to show the links. Because in a document normally the links are shown in a different way either underlined or in a different color. So you know that these are portions which you can click using the mouse. This is the link and vlink says the links which are already clicked are visited. Normally they are shown by a different color, so that you remember that these are links that I have already clicked that color also you can specify. So in general you can specify these three colors as different. One is active link, one is the normal displayable link and other is the visited link.

How to specify Color?

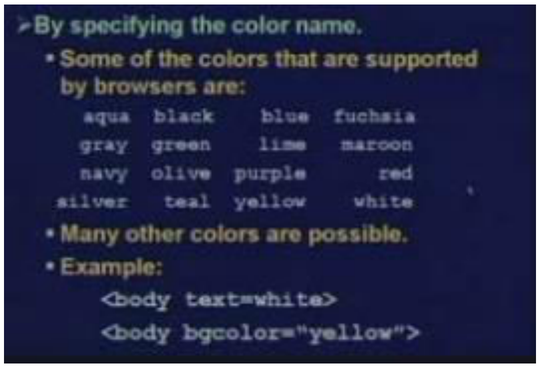


- Two ways:
  - By specifying the red, green, blue or RGB components.
    - Each color encoded in 8 bits.
    - 00 means that the color is turned off; FF means fully turned on.
    - Example:

```
<body text="#FFFFFF"
      bgcolor="#0000FF">
```

The next is how do I specify colors? Well here although I have given some examples. Let me tell you how you can do this. There are two ways the first way and the most flexible way is to specify the red, green and blue components of the color. Now you may be knowing that red, green, blue, these are three fundamental colors and that any color in this universe can be generated by some combination of these three. So in this method we are specifying this red, green and blue components separately and each of this components are encoded in 8 bits. So in hexadecimal it can be minimum 00 maximum FF. 00 means the contribution of the color is minimum it is turned off. FF means the contribution is maximum.

Well, why we specify a color in this way, you do it like this; body text equal to for example this is a color. The hash sign indicates that you are specifying it in this mode color red, green, blue mode. This hash also means hexadecimal. The first two characters means red, next two means green, last two means blue. In the first one red is FF, green is FF, blue is also FF, which means red, green, blue all are maximum combination of which means white. This is how we specify the white color. Similarly for bgcolor red is 0, green is 0, blue is maximum. So we are specifying a deep blue color. Well in this way you can specify any kind of color if you know the combination which combination will result in which color.



- By specifying the color name.
  - Some of the colors that are supported by browsers are:

```
  aqua  black    blue  fuchsia
  gray  green    lime   maroon
  navy  olive  purple      red
silver   teal  yellow    white
```

  - Many other colors are possible.
  - Example:

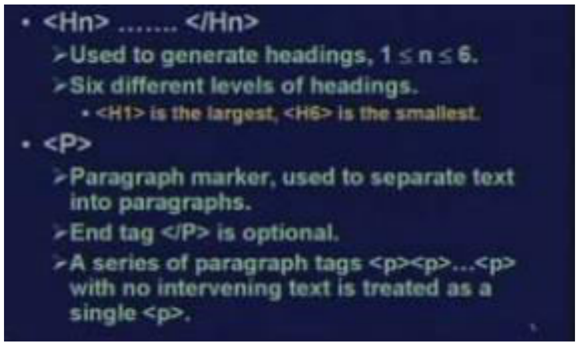
```
<body text=white>
<body bgcolor="yellow">
```

But there is an easier alternative. Of course there are some limitations to this you cannot arbitrarily specify any color. There is a list of color names you can specify the color by the name. Some of the colors that all browsers support are these aqua, black, blue, gray, green, lime, and etcetera. So you can use one of these colors also in this specification. For example you can say body text equal to white, body bgcolor equal to yellow. Well, this example shows you that this begins and end codes are optional. This you can give and you may not give both will be interpreted correctly by the browser.

Text Formatting in HTML**: [RGPV/Dec 2013(6)]**

How we can do text formatting in html? Well in html one of the most important things is to display text. So, let us see how we can display text and how we can do different kinds of formatting using html.

Paragraphs and Headings

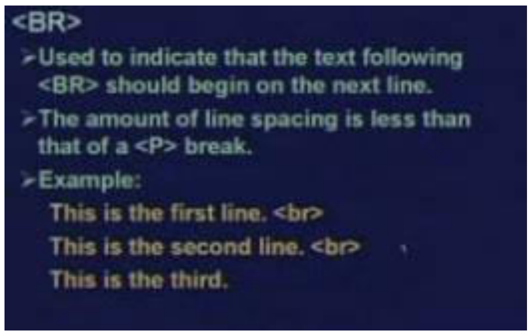


- <Hn> ……. </Hn>
  - Used to generate headings, $1 \le n \le 6$.
  - Six different levels of headings.
    - <H1> is the largest, <H6> is the smallest.
- <P>
  - Paragraph marker, used to separate text into paragraphs.
  - End tag </P> is optional.
  - A series of paragraph tags <p><p>…<p> with no intervening text is treated as a single <p>.

First let us talk about paragraphs and headings. Well html allows you to specify headings by using the h tag; begin h, end h, followed by number n. N is a number which can vary from 1 to 6. So you can have 6 different levels of headings. H1, H2, H3, H4, H5, H6. H1 means the biggest and the boldest headings. H2 means slightly smaller. H3 is even smaller and so on. H6 will be the smallest. Well this among this H1 to H6, all of the text which are enclosed in these headings. They will be displayed in bold or emphasized font. But their sizes will go on progressively reducing.

So if in a document you have several different levels of headings like a section name subsection, name sub, sub section name. You can have different classes or levels of headings to specify which section you are. So as I told you, H1 one is the largest. H6 is the smallest. We shall see examples later and whatever text you include between, this begin heading and end heading will get displayed with respect to that heading size. P, this indicates a paragraph break. This tag just a p within angular brackets is used to separate text into paragraphs.

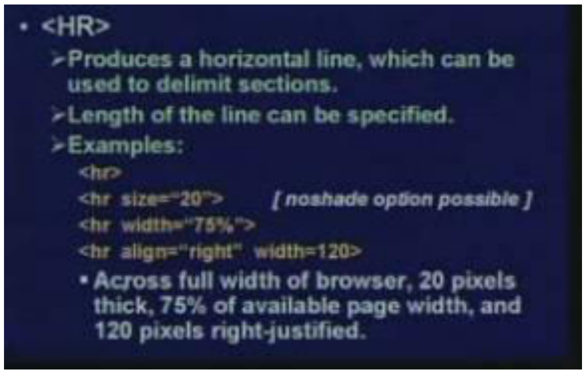
Well sometimes a paragraph you can enclose by giving a begin paragraph and finishing with an end paragraph. But the end paragraph tag is optional. This you need not always give just you can give a bracket p bracket between paragraphs automatically. Paragraph spacing or break will be taken. But one thing you remember that if you put a number of paragraph breaks one after the other with no intervening text. Then the browser will take only one paragraph break and ignore all the rest. So even if you put a number of different paragraph p p p by one after the other, only space equivalent to a single paragraph break will be taken or considered by the browser.


<BR>

- Used to indicate that the text following <BR> should begin on the next line.
- The amount of line spacing is less than that of a <P> break.
- Example:

  This is the first line.
  This is the second line.
  This is the third.

BR is a line break. Means this is used to specify that the text which follows should begin on the next line. Well in a sense, this is similar to paragraph break also but the difference is that, in a paragraph break, the spacing between the two sections of the document is more. But in a BR break, there is no extra line spacing. The only thing is that the text which follows that BR will start from the next line. But there will be no addition spacing between the line. This is the difference between BR and paragraph p. Just an example if you want that these three lines be displayed on three different lines, on the browser then you give a BR at the end of the first two lines. This is not a paragraph. Break the lines will be displaying will be displaying with the normal text spacing. Only the thing is that they will be starting on different lines. Well in many cases we want that some text begin from a new line. So in that case you can give the BR break command.



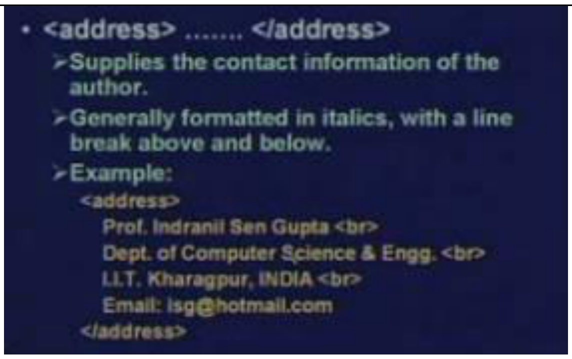
- <HR>
  - Produces a horizontal line, which can be used to delimit sections.
  - Length of the line can be specified.
  - Examples:

```
<hr>
<hr size="20">          [ noshade option possible ]
<hr width="75%">
<hr align="right" width=120>
```

  - Across full width of browser, 20 pixels thick, 75% of available page width, and 120 pixels right-justified.

This HR command produces a horizontal line. Sometimes we need to put a horizontal line like this in the document. Well this can be used to separate a delimit in the different sections. Now HR is flexible in the sense that the length of the line the width can be specified some example follows. If you give a simple HR, well this HR does not contain an end tag. This HR means a rule will be displayed across the full width. If you give HR size equal to 20, this 20 will mean the thickness of the line. That is, how thick the line will be that is the width of the line. Now if you give just like this HR size equal to 20, then the line will be displayed as a hollow thick line like this. What if we give additional attribute called no shade after this 20, just a space and no shade. Then this line will be solid.

This will come as a solid line. HR width is equal to 75 percent. You can also specify width. Width equal to 75 percent means it will have a width equal to 75 percent of current displayed window. If the window is bigger it will be 75 percent of that if we have resized the window to make it smaller. The line will be 75 percent of that. So you specify the relative width of the window with respect to the widow size of this line. And there is another way of specifying the width by specifying just a number. In this case, this 120 will mean that it is how many pixels the width is specified in terms of number of the number of dots. And this right is specified that your line will be right justified. It will be justified right you can just instead of right. You can specify left that case it will be left justified.

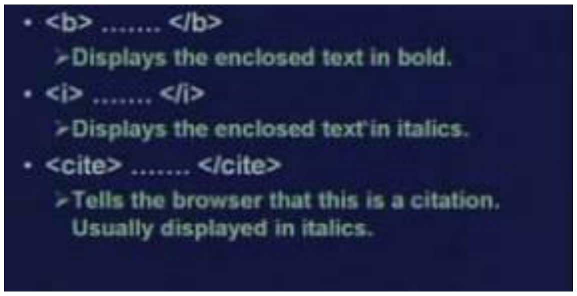


- <address> .... </address>
  - Supplies the contact information of the author.
  - Generally formatted in italics, with a line break above and below.
  - Example:

```
<address>
  Prof. Indranil Sen Gupta <br>
  Dept. of Computer Science & Engg. <br>
  I.I.T. Kharagpur, INDIA <br>
  Email: isg@hotmail.com
</address>
```

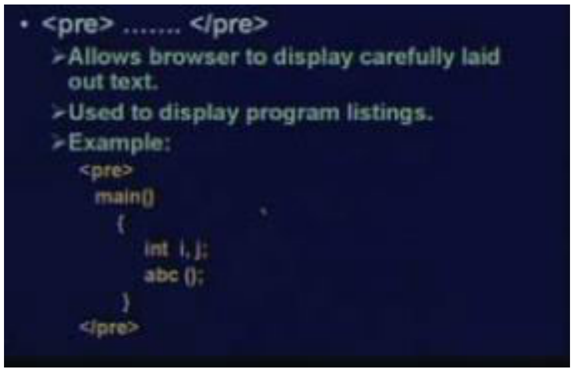
Sometimes you need to display the contact information of the author or the person you can use the address tags for that purpose. One example is shown here between begin address and at the end address. I can give my contact details. Of course with a line break across lines otherwise everything will display on the same line. Now if it is given in address normally some spacing is given at the beginning at the end and they are usually formatted in italics font.

Appearance of Text:



- <b> .... </b>
  - Displays the enclosed text in bold.
- <i> .... </i>
  - Displays the enclosed text in italics.
- <cite> .... </cite>
  - Tells the browser that this is a citation. Usually displayed in italics.

Now with respect to the appearance of text, there are some tags like when you want to display a text in the bold font you can use the b tag; begin b, end b. So the text everything which is there between the begin b and end b that will get displayed in bold font. Similarly you can display in italics. Italics is I; begin i and end i. Sometimes you want to display some kind of a citation. The browser has a facility to display citation in a particular form. There is begin cite, end cite tag in html for this. But most browser display the cite in italics font only, but with proper spacing. So cite and italic may be the same in some browser. But in some other browser the cite may be displayed in a different, may be in a different color. So depends on the browser really.

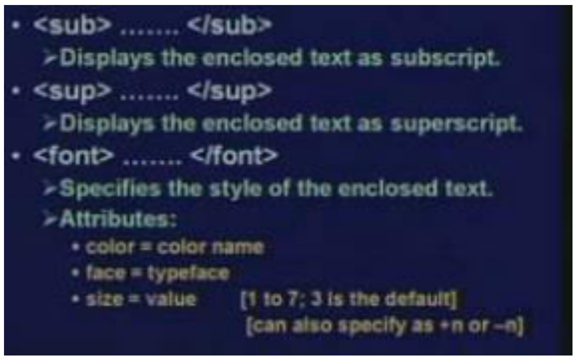


- <pre> .... </pre>
  - Allows browser to display carefully laid out text.
  - Used to display program listings.
  - Example:

```
<pre>
  main()
   {
       int  i, j;
       abc ();
   }
</pre>
```

Sometimes you need to display some text which is already preformatted. Well I am giving you two examples. Suppose I have a program code. But I have already given some comments some indentations so that my program code already looks good or I have an array of numbers. This can be table containing some array of information head expenditure income balance etcetera. So these are available in text form I want that the browser should display this in exactly the same form I type, I give space in between value type. So there is a tag called pre formatted tag called pre; in short begin pre, end pre. So whatever you include between the begin pre and end pre for example in the here we have included a c-program segment.

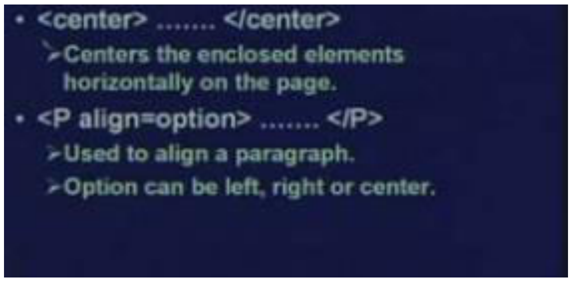
This c-program segment will get displayed exactly in this format. So this is how it will get displayed and normally in the preformatted display when it is displayed using the pre tags. The font size is some kind of uni space or mono space fonts like courier. In the courier font the width of all characters are equal. This is unlike other fonts like times new roman or Arial which we use more commonly where you have something called proportional spacing. Where an i is thinner than w for example. But in pre whenever it is displayed, it displayed in some kind of a font where spacing is equal like courier.



- _{.......}
  - Displays the enclosed text as subscript.
- ^{.......}
  - Displays the enclosed text as superscript.
- <font> .... </font>
  - Specifies the style of the enclosed text.
  - Attributes:
    - color = color name
    - face = typeface
    - size = value [1 to 7; 3 is the default] [can also specify as +n or –n]

Sometimes when you want to use a part of the text to be displayed as subscript, you can use the sub tags. Similarly for super script you can use the sup s u p tag. Well sometimes you can specify the style of the enclosed text using the font tag. Now in the font tag there are a number of attributes. Means after this begin font you can specify the attributes. Now attributes can be color equal to the name of the color. Face equal to type face. Well, here you have to know the name of the faces it can be Arial, it can be Courier, it can be Times New Roman etcetera. You can specify the size of the font also. The size can be specified in two ways. Well the size is a number from 1 up to 7; 3 is the default.

If you do not specify anything text are displayed in a font size of 3. So using this size you can specify any number from 1 to 7, 1 will mean a small font size; 7 mean the biggest. But instead of an absolute number you can also specify the relative number like plus n or minus n. Like you can specify size equal to plus 2 which means it is 2 steps above the current font size. If the current font size is 3, it will be 5; 2 steps above. So value can be a just a number 3 or it can be plus 2 something like this. So you can specify value in either of these two ways. So if you specify plus and minus, it will be with respect to some base font. Well there is a tag again to specify the base font size. So with respect to that you can specify.



- <center> .... </center>
  - Centers the enclosed elements horizontally on the page.
- <P align=option> .... </P>
  - Used to align a paragraph.
  - Option can be left, right or center.

Center is a tag which is used to center some text in a document. Begin center, end center whatever is in between will be displayed centered. Well I have said normally paragraph spacing does not need an end paragraph. But in some cases where you want to do some paragraph alignment you need to have this end paragraph. Well you can have this align operator. This align attribute, align equal to option. Now option can be either left, right or center. So you can have a paragraph which is aligned left, justified right justified or it is centered right.

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	Explain with example various text formatting attributes	Dec.2013	6

UNIT-02/LECTURE-03

XHTML : [RGPV/Jun 2014(7)]

XHTML reformulates the existing HTML technology to be an application of XML. It is already used for mobile phone and PDA Internet sites, and expected to gain widespread use as XML becomes more popular. Dan Wellman explains the differences between XHTML and HTML 4, and shows how easy it is to make the switch.

XHTML was invented to try to clean up the mess left by trying to make HTML a full-fledged presentation tool, and to reduce the fragmentation of HTML caused by the introduction of non-standard elements, mainly by Microsoft and Netscape. Additionally, XML is viewed by many as the future of the Internet, so reformulating existing markup technology to be an application of XML is a step towards embracing that future and letting go of the long standing legacy of the past. XHTML is now the specification of choice for mobile phone and PDA Internet sites, so its cross platform functionality is already being taken advantage of in excellent ways.

XHTML and its predecessor HTML 4 are extremely similar languages to use. Anyone that has worked with HTML 4 (or XML even) will find switching to XHTML extremely easy; it is more often then not simply learning to break those bad coding habits that many of us have found so easy to slip into. Remembering the few rules that come with XHTML, and trying to avoid any deprecated tags where possible, is all it really takes. In HTML, the mark-up surrounding your content is traditionally referred to as a tag, the <a> tag for example. In XML and subsequently, XHTML, these tags and the text enclosed within them are known as elements. For example:

<p>This sentence, including the opening and closing tags is known as a paragraph element</p>

<p>This is a separate paragraph element</p>

Basic XHTML syntax and semantics

XHTML is a separate language that began as a reformulation of HTML 4.01 using XML 1.0. It continues to be developed:

XHTML 1.0, published January 26, 2000 as a W3C Recommendation, later revised and republished August 1, 2002. It offers the same three variations as HTML 4.0 and 4.01, reformulated in XML, with minor restrictions.

XHTML 1.1, published May 31, 2001 as a W3C Recommendation. It is based on XHTML 1.0

Strict, but includes minor changes, can be customized, is reformulated using modules from Modularization of XHTML, which was published April 10, 2001 as a W3C Recommendation.

XHTML 2.0,. There is no XHTML 2.0 standard. XHTML 2.0 is incompatible with XHTML 1.x and, therefore, would be more accurate to characterize as an XHTML-inspired new language than an update to XHTML 1.x.

XHTML 5, which is an update to XHTML 1.x, is being defined alongside HTML 5 in the HTML 5 draft.

Markup:

HTML markup consists of several key components, including elements (and their attributes), character-based data types, character references and entity references. Another important component is the document type declaration, which specifies the

Document Type Definition. As of HTML 5, no Document Type Definition will need to be specified and will only determine the layout mode.

The Hello world program, a common computer program employed for comparing programming languages, scripting languages and markup languages is made of 9 lines of code although in HTML newlines are optional:

```
<!doctype html>
<html>
<head>
<title>Hello HTML</title> </head>
<body>
<p>Hello World!</p> </body>
</html>
```

(The text between <html> and </html> describes the web page, and The text between <body> and </body> is the visible page content.)

This Document Type Declaration is for HTML 5. If the <!doctype html>
declaration is not included, Windows Internet Explorer will render using "quirks mode".

HTML Text Formatting Tags

Tag	**Description**
<b>	Defines bold text

<big>	Defines big text
<em>	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
<strong>	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text

HTML "Computer Output" Tags

Tag	**Description**
<code>	Defines computer code text
<kbd>	Defines keyboard text
<samp>	Defines sample computer code
<tt>	Defines teletype text
<var>	Defines a variable
<pre>	Defines preformatted text

HTML Citations, Quotations, and Definition Tags

Tag	**Description**
<abbr>	Defines an abbreviation
<acronym>	Defines an acronym
<address>	Defines contact information for the author/owner of a document
<bdo>	Defines the text direction

<blockquote>	Defines a long quotation
<q>	Defines a short quotation
<cite>	Defines a citation
<dfn>	Defines a definition term

Your browser does not support inline frames or is currently configured not to display inline frames.

RELATIVE URLS:

A URL is another word for a web address.

A URL can be composed of words, such as "w3schools.com", or an Internet Protocol (IP) address: 192.68.20.50. Most people enter the name of the website when surfing, because names are easier to remember than numbers.

URL - Uniform Resource Locator

When you click on a link in an HTML page, an underlying <a> tag points to an address on the world wide web.A Uniform Resource Locator (URL) is used to address a document (or other data) on the world wide web.

scheme://host.domain:port/path/filename

Explanation:

scheme - defines the type of Internet service. The most common type is http host - defines the domain host (the default host for http is www)

domain - defines the Internet domain name, like w3schools.com

:port - defines the port number at the host (the default port number for http is 80)

path - defines a path at the server (If omitted, the document must be stored at the root directory of the web site

filename - defines the name of a document/resource

Common websites start with http://. Pages starting with http:// are not encrypted, so all information exchanged between your computer and the Internet can be "seen" by hackers.

Secure websites start with https://. The "s" stands for "secure". Here, the information

we dont take any liability for the notes correctness. http://www.rgpvonline.com

exchanged will be encrypted, making it useless to hackers.

Common URL Schemes

The table below lists some common schemes:

Scheme	**Short for....**	**Which pages will the scheme be used for...**
http	HyperText Transfer Protocol	Common web pages starts with http://. Not encrypted. Unwise to enter personal information in http:// pages
https	Secure HyperText Transfer Protocol	Secure web pages. All information exchanged are encrypted, cannot be read by hackers
ftp	File Transfer Protocol	For downloading or uploading files to a website. Useful for domain maintenance
file		A file on your computer
gopher		A Gopher document or menu
news		A newsgroup
WAIS	Wide Area Information Search	A database or document on a WAIS database

HTML TABLES

Tables are defined with the <table> tag.

A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, other tables, etc.

Table Example

```
<table border="1"> <tr>
<td>row 1, cell 1</td> <td>row 1, cell 2</td> </tr>
<tr>
<td>row 2, cell 1</td> <td>row 2, cell 2</td> </tr>
</table>
```

How the HTML code above looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

HTML Tables and the Border Attribute

If you do not specify a border attribute, the table will be displayed without borders. Sometimes this can be useful, but most of the time, we want the borders to show.

To display a table with borders, specify the border attribute:

```
<table border="1"> <tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td> </tr>
</table>
```

HTML Table Headers

Header information in a table are defined with the <th> tag.

The text in a th element will be bold and centered.

```
<table border="1"> <tr>
<th>Header 1</th> <th>Header 2</th> </tr>
<tr>
<td>row 1, cell 1</td> <td>row 1, cell 2</td> </tr>
<tr>
<td>row 2, cell 1</td> <td>row 2, cell 2</td> </tr>
</table>
```

How the HTML code above looks in a browser:

Header 1	**Header 2**
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

HTML LISTS

The most common HTML lists are ordered and unordered lists:

HTML Lists

An ordered list:	An unordered list:
1. The first list item	List item
2. The second list item	List item
3. The third list item	List item

HTML Unordered Lists

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

The list items are marked with bullets (typically small black circles).

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

How the HTML code above looks in a browser:

Coffee

Milk

HTML Ordered Lists

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

The list items are marked with numbers.

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

How the HTML code above looks in a browser:

1. Coffee
2. Milk

HTML Definition Lists

A definition list is a list of items, with a description of each item.

The <dl> tag defines a definition list.

The <dl> tag is used in conjunction with <dt> (defines the item in the list) and <dd> (describes the item in the list):

```
<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd> <dt>Milk</dt>
<dd>- white cold drink</dd> </dl>
```

How the HTML code above looks in a browser:

Coffee

- black hot drink

Milk

- white cold drink

Basic Notes - Useful Tips

Tip: Inside a list item you can put text, line breaks, images, links, other lists, etc.

HTML List Tags

Tag	Description
<ol>	Defines an ordered list
<ul>	Defines an unordered list
<li>	Defines a list item
<dl>	Defines a definition list
<dt>	Defines an item in a definition list
<dd>	Defines a description of an item in a definition list

HTML FORMS

HTML forms are used to pass data to a server.

A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.

The <form> tag is used to create an HTML form:

```
<form>
.
input elements
.
</form>
```

HTML Forms - The Input Element

The most important form element is the input element. The input element is used to select user information.An input element can vary in many ways, depending on the type attribute. An input element can be of type text field, checkbox, password, radio button, submit button, and more.The most used input types are described below.

Text Fields

<input type="text" /> defines a one-line input field that a user can enter text into:

<form>

First name: <input type="text" name="firstname" /><br /> Last name: <input type="text" name="lastname" /> </form>

How the HTML code above looks in a browser:

First name:

Last name:

Note: The form itself is not visible. Also note that the default width of a text field is 20 characters.

Password Field

<input type="password" /> defines a password field:

<form>

Password: <input type="password" name="pwd" /> </form>

How the HTML code above looks in a browser:

Password:

Note: The characters in a password field are masked (shown as asterisks or circles).

Radio Buttons

<input type="radio" /> defines a radio button. Radio buttons let a user select ONLY ONE one of a limited number of choices:

<form>

<input type="radio" name="sex" value="male" /> Male<br /> <input type="radio" name="sex" value="female" /> Female </form>

How the HTML code above looks in a browser:

○ Male

○ Female

Checkboxes

<input type="checkbox" /> defines a checkbox. Checkboxes let a user select ONE or MORE options of a limited number of choices.

<form>

<input type="checkbox" name="vehicle" value="Bike" /> I have a bike<br /> <input type="checkbox" name="vehicle" value="Car" /> I have a car </form>

How the HTML code above looks in a browser:

☐ I have a bike

☐ I have a car

Submit Button

<input type="submit" /> defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

<form name="input" action="html_form_action.asp" method="get"> Username: <input type="text" name="user" />

<input type="submit" value="Submit" /> </form>

How the HTML code above looks in a browser:

Username: ▭ SUBMIT

If you type some characters in the text field above, and click the "Submit" button, the browser will send your input to a page called "html_form_action.asp". The page will show you the received input.

HTML Form Tags

Tag	**Description**
<form>	Defines an HTML form for user input
<input />	Defines an input control
<textarea>	Defines a multi-line text input control
<label>	Defines a label for an input element
<fieldset>	Defines a border around elements in a form

<legend>	Defines a caption for a fieldset element
<select>	Defines a select list (drop-down list)
<optgroup>	Defines a group of related options in a select list
<option>	Defines an option in a select list
<button>	Defines a push button

HTML FRAMES: [RGPV/Jun 2010(8)]

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

The web developer must keep track of more HTML documents

It is difficult to print the entire page

The HTML frameset Element

The frameset element holds two or more frame elements. Each frame element holds a separate document.The frameset element states only HOW MANY columns or rows there will be in the frameset.

The HTML frame Element

The <frame> tag defines one particular window (frame) within a frameset.In the example below we have a frameset with two columns.

The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The document "frame_a.htm" is put into the first column, and the document "frame_b.htm" is put into the second column:

```
<frameset cols="25%,75%"> <frame src="frame_a.htm" />
<frame src="frame_b.htm" /> </frameset>
```

Navigation frame

How to make a navigation frame. The navigation frame contains a list of links with the second frame as the target. The file called "tryhtml_contents.htm" contains three links. The source code of the links:

<a href ="frame_a.htm" target ="showframe">Frame a</a><br> <a href ="frame_b.htm" target ="showframe">Frame b</a><br> <a href ="frame_c.htm" target ="showframe">Frame c</a>

The second frame will show the linked document.

Jump to a specified section with frame navigation

Two frames. The navigation frame (content.htm) to the left contains a list of links with the second frame (link.htm) as a target. The second frame shows the linked document. One of the links in the navigation frame is linked to a specified section in the target file. The HTML code in the file "content.htm" looks like this: <a href ="link.htm" target ="showframe">Link without Anchor</a><br><a href ="link.htm#C10" target ="showframe">Link with Anchor</a>.

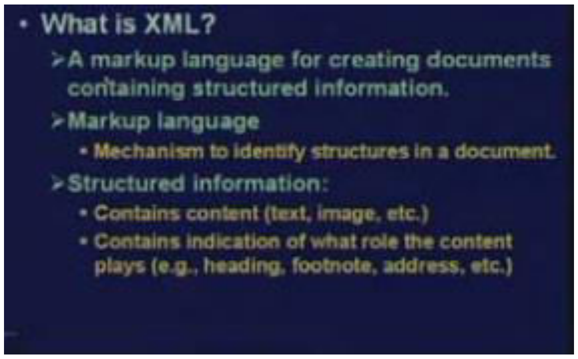
HTML Frame Tags

Tag	**Description**
<frameset>	Defines a set of frames
<frame />	Defines a sub window (a frame)
<noframes>	Defines a noframe section for browsers that do not handle frames
<iframe>	Defines an inline sub window (frame)

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	Discuss the difference between HTML and XHTML with respect to elements.	June.2014	7
Q.2	Write a HTML code to create the following frames. What is a frameset? Rajiv Gandhi Praudyogiki Vishwavidyalaya Bhopal, M.P. Menu • About us • Scheme • Result Welcome	June.2010	8

UNIT-02/LECTURE-04

eXtensible Markup Language (XML) : [RGPV/Jun 2011,12(10)]

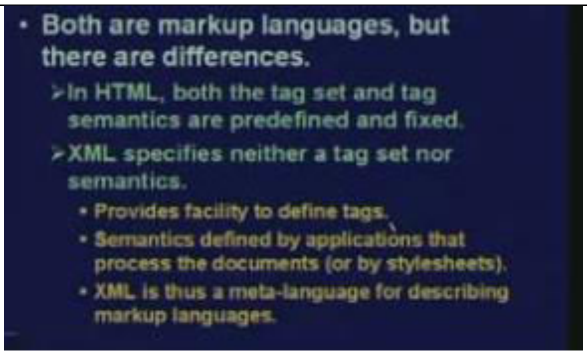


- What is XML?
  - A markup language for creating documents containing structured information.
  - Markup language
    - Mechanism to identify structures in a document.
  - Structured information:
    - Contains content (text, image, etc.)
    - Contains indication of what role the content plays (e.g., heading, footnote, address, etc.)

,

Html is a language using which you can create the web pages. It is a markup language. Markup language means, we specify the textual content of the document. As well as we specify how the textual content should be structured or should be viewed on the screen. The concept of the tags and attributes come from there. So this is what a markup language is. Now XML like html is also a markup language. So here lies a similarity. Let us see. So XML is also a markup language. Well although here we are talking about creating documents. We shall see later that XML has been used not only for creating document but for many other applications as well. There is some structured information that we can capture through tags like html. Markup language is that, markup language use of tags is a mechanism to identify structure. For example a begin paragraph and end paragraph tag will tell you that, well this is a paragraph in the document. Begin h1 and end h1 will tell you that this is a heading in the document. So these are structure information which you can extract. So structure information will contain content which can be text, which can be other form of information also, image. And also it should contain of what role the content plays. For example I gave you some examples, heading, paragraph, address. But one problem with html is that the set of tag is limited by the definition of html. If for example I want to define a new tag for country, for city, I do not have it. But in XML we can do it. We can define our own tag we shall see it later.

XML vs. HTML



- Both are markup languages, but there are differences.
  - In HTML, both the tag set and tag semantics are predefined and fixed.
  - XML specifies neither a tag set nor semantics.
    - Provides facility to define tags.
    - Semantics defined by applications that process the documents (or by stylesheets).
    - XML is thus a meta-language for describing markup languages.

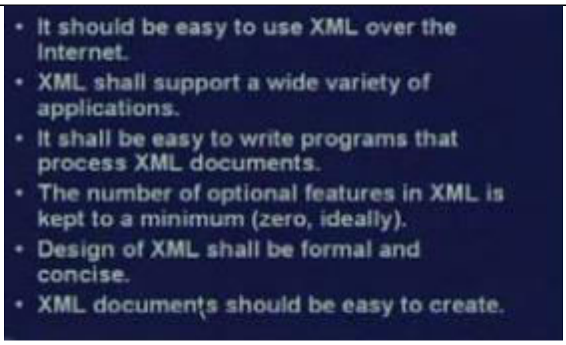
XML and html first thing is that both are markup languages, there are differences as I had mentioned. This html is a fixed language the tag set and the meanings of the tags are predefined. They are fixed and you cannot change it. The browsers have been designed following these rules. Even you cannot change it in the browser you cannot tell the browser that h2 tag you should display it in italics in color red that we cannot tell the browser. You will have to modify the document itself in order to make this into effect but in contrast XML specifies neither a tag set nor semantics.

XML is a totally open language and totally flexible language it does not specify anything by default, depending on your application domain your context you will have to define everything and it is entirely up to you can customize XML as you want it to be. So XML provides you the facility to define tags. It can allow you to define semantics which can be defined by applications for example this html means I repeat. Html is meant for creating documents XML is meant for creating some content. This content is not only meant for viewing on the browser.

There are a much wider range of applications of xml, as you can see some typical application later. The idea is like this. Whatever you define in XML you can write a program later may be in C, C++, Java, whatever that program can read that document and can process it. So the application there will be of course some application programming interfaces to extract the structure information from the XML specification. And the semantics can be defined by the application that what a particular tag means that the application can define or can understand.

So XML is therefore more like a meta language it is not a fixed language which defines everything it is a language using which you can describe markup language. For example using html you can define html because html has a fixed set of tags you can define all of them in html, in that means you can define al of them in html and whatever you get is an html implementation using xml. Similarly using XML you can do a host of other things host lot of other things.

XML Development Goals

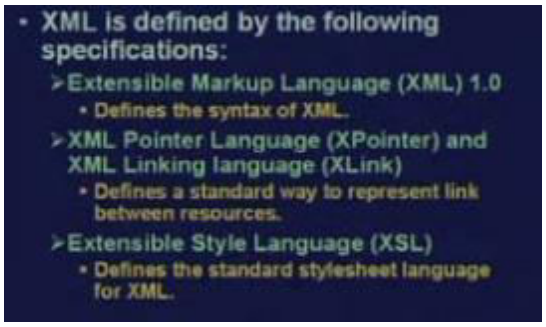


- It should be easy to use XML over the Internet.
- XML shall support a wide variety of applications.
- It shall be easy to write programs that process XML documents.
- The number of optional features in XML is kept to a minimum (zero, ideally).
- Design of XML shall be formal and concise.
- XML documents should be easy to create.

But the primary goals of XML if you look at it, you will understand a few things. The first was that initially it was meant to be used over the internet. So the first goal was it should be easy to use over the internet. This also means that the browsers you see around us they should understand xml. There should be some complaints in deed most of the modern browsers can understand xml. XML shall support a wide variety of application, this is a secondary objective. This application does not necessarily mean that only viewing the documents on the browser. It shall be easy to write programs that process XML documents. These are these applications which are running using XML specifications. The number of optional features in XML is kept to a minimum ideally it is zero.

See optional feature means if you look at the at the html specification, you will find the some of the specifications were referred to has been mandatory. But some were set to be optional and there was a problem with the optional specifications. Some browsers implemented some optional features, some other browsers implemented some other set of optional features. So if you create a document using the optional features there is no guarantee that can interpret it in a correct way. So if there were no optional features at all there would be no ambiguity. So XML does or aims to just that. Design of XML shall be formal and concise it is so and it should be easy to create. There are many XML document editors available now a days using which you can create XML documents.

How is XML Defined?

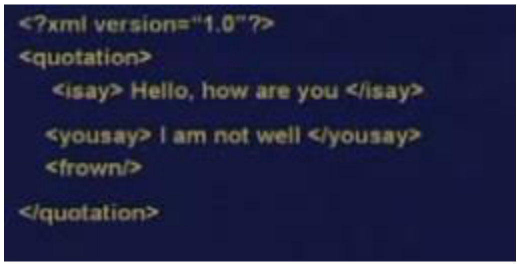


- XML is defined by the following specifications:
  - ➢Extensible Markup Language (XML) 1.0
    - Defines the syntax of XML.
  - ➢XML Pointer Language (XPointer) and XML Linking language (XLink)
    - Defines a standard way to represent link between resources.
  - ➢Extensible Style Language (XSL)
    - Defines the standard stylesheet language for XML.

So let us see how XML is defined. XML is defined broadly using three different specifications or specification languages whatever you call. The first is the basic XML specification. This is the extensible markup language version “1.0”. This defines the syntax of xml. The language constructs how we have to how you need to write the constructs of the language the syntax. The second thing is there is something called XML pointer language in short x pointer; XML linking language in short x link. These are still undergoing some refinements. Now this xpointer and xlink, these are some specifications or languages using which you can create some thing similar to hyperlinks as you create in html. The standard XML specification does not support hyperlink.

You have to use xlink specification for that. So this xlink or xpointer this defines standard ways to represent link between resources and the third one is extensible style language or xsl. This is something similar to the html style sheet that we were discussing in the last class. Here you can define the standard style sheets and this language specifies how means the exact format and syntax using which you can do that. The advantage of having xsl is obvious. Suppose your main application is to use the browser for viewing information. Then you will be creating a style sheet xsl specifically for browser related definitions and syntax semantics whatever you call. So how to compose paragraphs headings center different kinds of alignments, fonts, etcetera.

An Example of XML Document

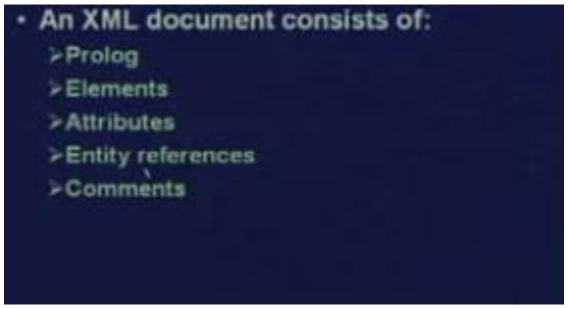


```
<?xml version="1.0"?>
<quotation>
   <isay> Hello, how are you </isay>
   <yousay> I am not well </yousay>
   <frown/>
</quotation>
```

A very small example an XML document looks like this. It starts with a beginning line. This is sometimes called as an XML prolog. This tells you that this is an XML document it begins with less than question mark followed by the key word xml. Version specifies which version of XML it is followed by end and inside the XML there can be several tags just like html like quotation, begin quotation, end quotation, Isay, end isay, yousay, end yousay. See these tags look very peculiar it is indeed peculiar. They were not part of the definition I have defined it according to my need. This particular example I have taken may be it is a portion on except from a dialogue that is going on between two persons myself and you. See whenever I am enclosing some text between the tag isay. This is means I am telling you something between yousay means you are telling something. There can be a large number of alternates of isay and yousay in the total document that will constitute the whole conversation

But I am calling quotation, you can call it something else also. In xml in some cases you may have a pair of tags with nothing in between it is called an empty tag. There is a way to specify an empty tag it is like this the name of the tag followed by slash. This is of course equivalent to begin frown end frown. So instead of writing this frown twice. I can simply write frown followed by slash. This will mean that it is the frown tag but it is presently empty nothing is there in between. So this gives an idea about what or how means html document will look like there are some tags. Well I had used some tags according to my need and I can understand what the meaning of this tags are but may be you will not understand. So the tags will be defined depending on the application which will be processing it or the person who will be viewing it or looking at it. This is the basic idea.

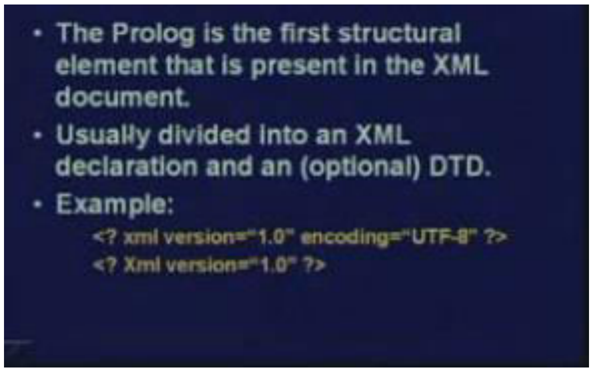
Structure of XML Document: [RGPV/Dec 2013(5)]



- An XML document consists of:
  - ➢Prolog
  - ➢Elements
  - ➢Attributes
  - ➢Entity references
  - ➢Comments

So talking about the structure of a XML document there are 5 different components of the document. Prologs, elements, attribute, entity references and comments. Let us see what these are.

XML Prolog



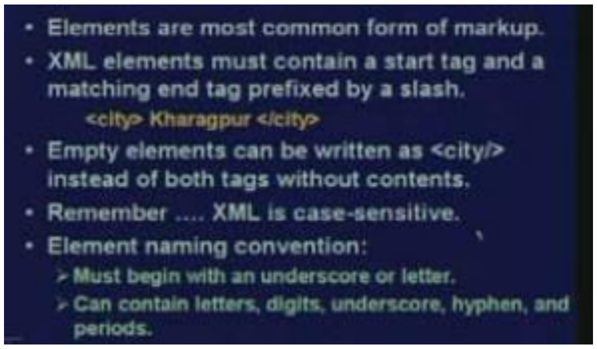
- The Prolog is the first structural element that is present in the XML document.
- Usually divided into an XML declaration and an (optional) DTD.
- Example:

```
<? xml version="1.0" encoding="UTF-8" ?>
<? Xml version="1.0" ?>
```

First is prolog. Prolog as I said with the help of that example. That is the beginning you need to specify that this an XML document that is the prolog. It is the beginning or starting of the document that tells you the prolog has to be the first structural element that is present. First structural element means before that you can have comments. Comments are just like html, so there is no confusion. There you can have blank lines. But the firststructural element will be the prolog. Prolog is sometimes divided into an XML declaration and an optional so called data type declaration that we shall see later.

Some typical examples of prolog definition are this. The second version we have already seen earlier the first version has an optional tag. This is the DTD I have said. This specifies some encoding rules. This is an XML file version “1.0” it uses some encoding rules UTF8. UTF8 is a standard encoding rule where some special character can be encoded by some special escape sequences and the like. So that when you are processing it, just looking at the prolog you will understand that what kind of special characters and their encodings you are likely to encounter in the document

XML Elements: : [RGPV/Dec 2010(10)]



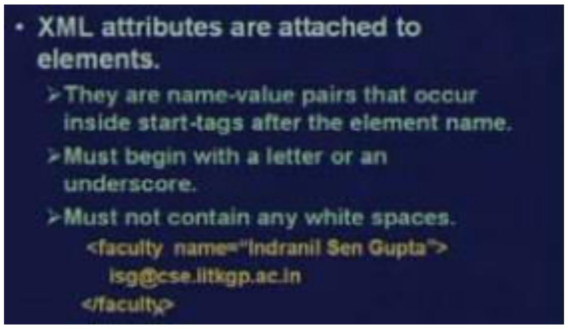
- Elements are most common form of markup.
- XML elements must contain a start tag and a matching end tag prefixed by a slash.
  <city> Kharagpur </city>
- Empty elements can be written as <city/> instead of both tags without contents.
- Remember .... XML is case-sensitive.
- Element naming convention:
  - Must begin with an underscore or letter.
  - Can contain letters, digits, underscore, hyphen, and periods.

These XML elements are like tags. So XML elements they are the most important form of markup or the way of specifying some sections of the document or text whatever you are correct, you are doing or writing. Now XML elements when you are using, they must contain a start tag and a matching end tag just like html. You can say that in html the elements were predefined and you are calling them tags so begin tag end tag there are ways of doing it. End tag starts with a slash with the same name. So here also that syntax remains the same. A small example an element we are calling it city end element slash city in between some text.

Well this text can be arbitrary, but in the context of XML we understand that this Kharagpur is a name of a city. Some application which is reading the data will understand that this Kharagpur is part of an element called city. So the application understands that this is a name of a city. So semantic is also captured here very nicely. And as I mentioned in that example, that if you have empty elements you can write them by putting a slash after the name of the element. Well here alternatively you can also use both the tags without any content in between blank and one thing to note is that unlike html XML is case sensitive. So when you are defining an element city lower case c i t y and capital C I T Y are not the same.

You must follow the same case convention when you are using xml. When you are just using XML for some other applications. There is a naming convention for the element. That means when you define for example I have given the name city. So how you can define names? There are some simple rules. Rule says, that the name must begin with an underscore or a letter. Letter means a correct a to z. But within the name you can have any number of letters, digits, underscore, hyphen and dots or periods. So any combination of this will constitute the name of the element. So it is an element you can construct using any combination of these.

XML Attributes



- XML attributes are attached to elements.
  - They are name-value pairs that occur inside start-tags after the element name.
  - Must begin with a letter or an underscore.
  - Must not contain any white spaces.

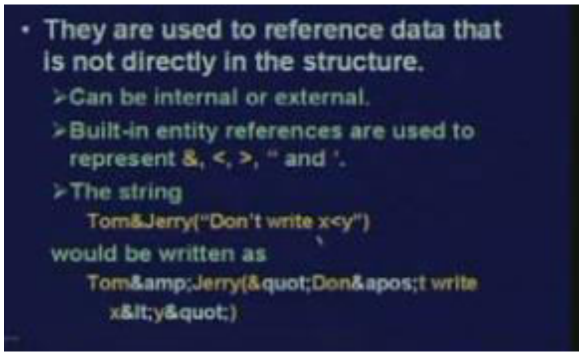
```
<faculty  name="Indranil Sen Gupta">
   isg@cse.iitkgp.ac.in
</faculty>
```

There is something called attributes, this attributes are also quite similar to what is there in html. In html we had tags; along with tags we had attributes. Whenever we have the begin tag along with the begin tag we specify the attributes. Here also we do something very similar the syntax is very similar here also. Here the tags are called elements. So we say that attributes are associated with elements. So XML attributes are attached to elements they are mostly name and value pairs that occur inside start tags like html after the element name. What I mean is that something like this. Suppose in this example we have defined an element whose name is faculty.

And in this start faculty tag element you can say I have defined an attribute. The name of the attribute is n a m e name and indranil sen gupta is the value of that attribute. And within this I can have anything. For example here the body contains the email address but it can contain anything. So name value pairs will constitute attributes. So this simple example shows that we have associated one attribute with this element faculty. Must begin with a letter or underscore. The attribute name must not begin with anything else.

This is the constraint and must not contain any white spaces. You cannot give any blank in between. This equal to before and after should not be any blank

XML Entity Reference



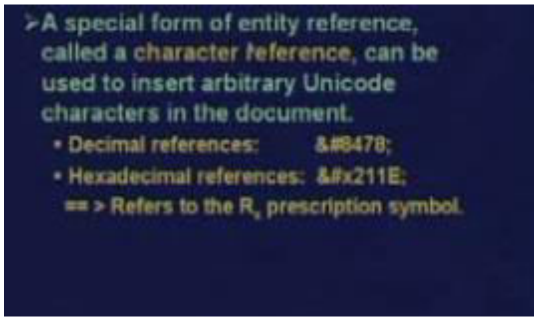
- They are used to reference data that is not directly in the structure.
  - Can be internal or external.
  - Built-in entity references are used to represent &, <, >, " and '.
  - The string
    Tom&Jerry("Don't write x<y")
    would be written as
    Tom&amp;Jerry(&quot;Don&apos;t write x&lt;y&quot;)

Well entity references, see entity references actually mean that when you are creating a document you define some tags. Tags normally use less than greater than for defining tags. So use some of the symbols that are available in your alphabet that are available in the key board to define something special. Now if in the document those same symbols are appearing you normally use some kind of escape sequence to indicate that well this is that particular character not the starting of a tag less than for example. So the entity references as it is mentioned under XML actually talks about exactly that. They are used to reference data that are not directly in the struct.

Not directly in the structure means you cannot normally use those data or those symbols in the structure because they mean something else. But in order to use that you will have to do something else. So there are two things. This can be internal, this can be external. There are two ways of doing it. So internal means the entity references whatever you do; that will define as part of the same file and external as the name implies this will be in some other fact. But in addition to internal external there are other ways of mentioning. We will see it through examples later. Built in entity reference something internal external there is something builtin. This is already there.

Built-in means these are some special symbols which we use in XML ampersand, less than, greater than, quotation, double quote and apostrophe, single quote. So a small example follows. Suppose I have a string like this Tom and Jerry bracket quotation do not. There is an apostrophe in between. Write x less than y quote bracket closed. Now if you write a string like this, this ampersand double quote single quote less than, this will totally confuse the XML parser or processor because they some other meaning to xml. So you will have to escape out this using special escape sequences and specify it in a slightly different way. The same text you will be writing like this.

Instead of this ampersand symbol you will be writing ampersand amp semi colon. This ampersand is the escape sequence here all special symbols will start with ampersand then some name small short name followed by a semicolon. So ampersand amp semi colon is this jerry then this quotation comes double quote. This is ampersand quote ampersand semicolon. Don, d o n then the single quote comes this is ampersand apostrophe semi colon. Write x less than y less than is again lt double quote at the end this is quote. So in this way where ever there is a special symbol that you replace by an escape sequence which starts with ampersand followed by some short name for that particular symbol followed by a semi colon. This is the XML builtin entity reference.



- A special form of entity reference, called a character reference, can be used to insert arbitrary Unicode characters in the document.
  - Decimal references: &#8478;
  - Hexadecimal references: &#x211E;

  == > Refers to the R_x prescription symbol.

Now there is a special form of entity reference also which you can use. This is called a so called character reference. Now using character reference, you can insert arbitrary characters in the documents. See you know this Unicode is a 16 bit character code which can be used to virtually represent any character in any language. So this allows you to insert arbitrary Unicode characters in the document. Well I am giving a small example. How to do that? You can specify the character code in either decimal or in hexadecimal. In decimal you do it in this way. It as usual starts with ampersand.

This hash symbol indicates that this is a character reference followed by 8478 means the corresponding decimal equivalent of the Unicode 16 bit number; followed by semicolon. Hexadecimal is similar, only you use a x before the hexadecimal number. Ampersand hash x 211 E. These two actually means the same number and this refers to the prescription symbol which the doctors give on the prescription that r a small x. That r x is a special symbol that has Unicode number 8478 in decimal. So this is just an example which shows that if you give this r this means 8478, actually this prescription symbol will appear on the browser if you are viewing it on the browser screen.

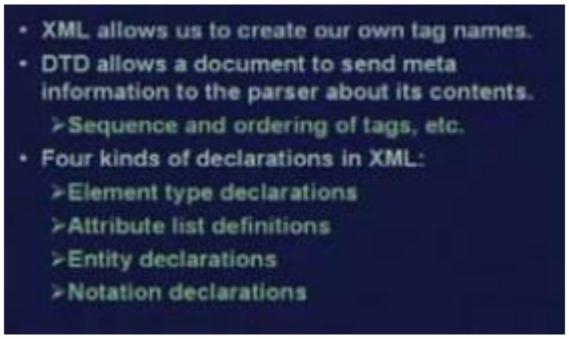
S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	What is XML? What are the differences between XML and HTML	June.2012	10
Q.2	Explain Extensible Markup Language(XML).	June.2011	10
Q.3	What are the components of an XML file?	Dec.2013	5
Q.4	Explain XML elements and attributes	Dec.2010	10

we dont take any liability for the notes correctness. http://www.rgpvonline.com

UNIT-02/LECTURE-05

DTD files : [RGPV/Jun 2012(10),2014(7)]

Document Type Declaration

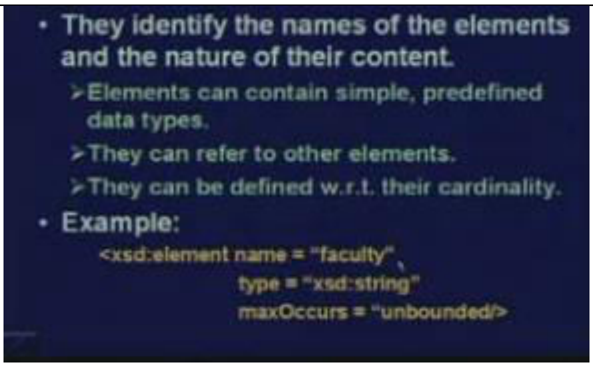


- XML allows us to create our own tag names.
- DTD allows a document to send meta information to the parser about its contents.
  - ➢Sequence and ordering of tags, etc.
- Four kinds of declarations in XML:
  - ➢Element type declarations
  - ➢Attribute list definitions
  - ➢Entity declarations
  - ➢Notation declarations

document type declarations or DTD. Say XML as though as though it may seem that through element declarations it allows us to create our own tag names. Document type declarations allow a document to send Meta information to the parser about it contents. Meta information means that you send some semantic information to the parser that well these are the tags and this is the meaning of the tag. This how you should interpret the tag this kind of Meta information can also be sent. Like I am giving a simple example, you think of a browser. Browser is a one very classic example of an XML application an application that is using XML specification. A web page written in XML that is been downloaded the browser tries to display it browser is now an application.

So browser is using an XML parser to extract some important information from the XML specification and try to display them in a proper way. Now DTD document type declarations are very useful without the document type declaration the browser will not know that. How to display this particular item on the screen bold centered underlined or how DTDs are important due to this. It also tells you that if there is any important sequencing and ordering of the tags. This also may be mentioned that tags must appear in this particular sequence. You cannot have any arbitrary usage of tags. Now broadly four kinds of declarations are there in XML right you can define elements, you can define attributes as a set of elements, you define something called entities and notations. These are the four different kinds of definitions that we encounter in XML specifications.

Element Type Declaration:



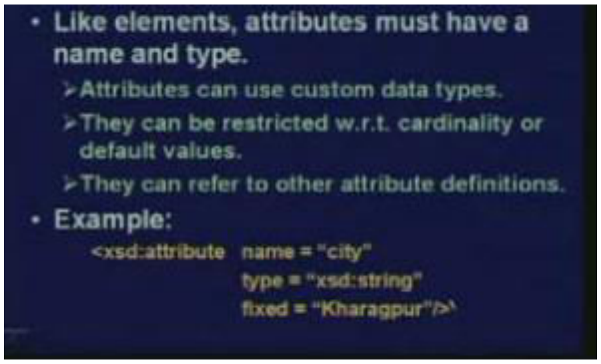
- They identify the names of the elements and the nature of their content.
  - ➢Elements can contain simple, predefined data types.
  - ➢They can refer to other elements.
  - ➢They can be defined w.r.t. their cardinality.
- Example:

```
<xsd:element name = "faculty"
             type = "xsd:string"
             maxOccurs = "unbounded/>
```

Let us see first is element type declaration. They identify the names of the elements and the nature of the content earlier we had seen what an element is and how you can use an element in a XML document. Let that city Kharagpur end city that was one example. Now here you see how you can define an element and XML document. Elements identified in names of the elements and the nature of the content. So I told you in XML there is no predefined element or tag everything I need. I will have to define myself. So this is a way of defining, so elements can contain simple predefined data types they can refer to other elements. They can also be defined with respect to their cardinality. Cardinality means how many times the element appears in the XML document.

Because I may say that a particular document city should not appear in the document more than twenty times or it is unbounded. It can appear any number of times. These kinds of specifications I can also mention through a small example. This defines an element whose name is faculty. Xsd colon element is the keyword used to define elements. Name equal to the name of the element you are trying to define, then type there are several predefined types under xsd again. Xsd colon string means, it is a string. Max occurs is another attribute which says that it is an unbounded that means that there are no limit to the number of times. The faculty element can appear in my XML document. So this way you can define the elements.

Attribute List Declaration

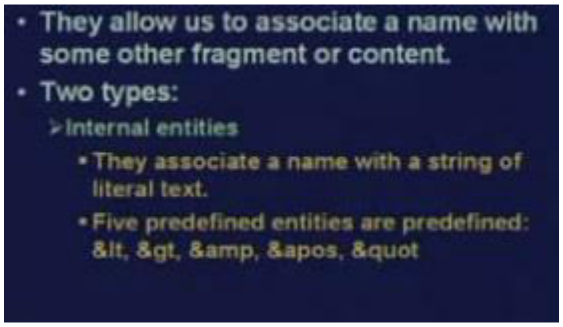


- Like elements, attributes must have a name and type.
  - ➢Attributes can use custom data types.
  - ➢They can be restricted w.r.t. cardinality or default values.
  - ➢They can refer to other attribute definitions.
- Example:

```
<xsd:attribute  name = "city"
                type = "xsd:string"
                fixed = "Kharagpur"/>
```

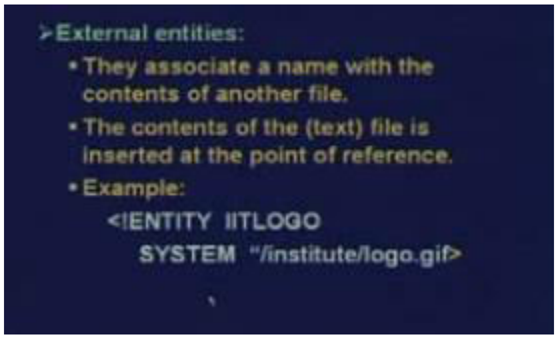
Next come attributes. How to define attributes associated with the elements. Like elements attributes must have a name and type. Attributes can use custom data type also they can also be restricted with respect to cardinality. As I had mentioned before and also some default values you can specify they can also refer to other attribute definitions. Like one small example follows here. Xsd attribute this keyword tells you that you are trying to define an attribute the name of the attribute is city. Type is again string fixed Kharagpur means this is a default value and these are restricted value cannot change it. But if you do not give this then anyone can use any kind of city by default. For example you are you are creating an XML document that is keeping track of all the bank branches in Kharagpur. So for them the city will always be Kharagpur you can define them as fixed type, fixed equal to Kharagpur for them this kind of declarations are relevant.

Entity Declaration



- They allow us to associate a name with some other fragment or content.
- Two types:
  - ➢Internal entities
    - They associate a name with a string of literal text.
    - Five predefined entities are predefined: &lt, &gt, &amp, &apos, &quot

Entity. Entity will allow us to associate a name with some other fragment or content. Like we have already seen internal entities. Like you are replacing some special characters by their escape sequences. This is one example of entity. That means it is like a macro substitution you have something to replace it with to something else. And later on when you again need it back you replace it back. It is like a macro substitution and d substitution. This is sometimes called an entity with which some string or something else is associated you replace the entity by that string. Essentially it means the same thing, it is like a macro substitution. You have a symbol or a string you are replacing it with by some other string. These are internal entities which you can define as part of the xml.

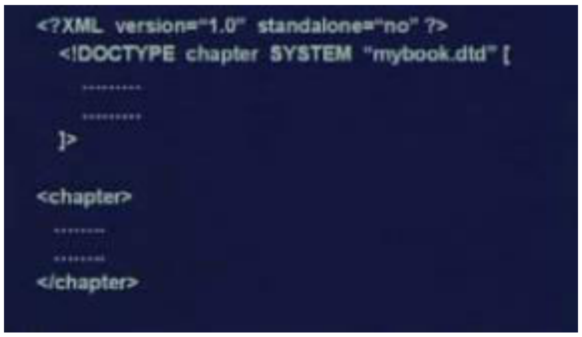


- ➢External entities:
  - They associate a name with the contents of another file.
  - The contents of the (text) file is inserted at the point of reference.
  - Example:

```
<!ENTITY IITLOGO
    SYSTEM "/institute/logo.gif>
```

External entities are those which associates a name with the contents of another file. This is also like a macro. But you replace it with something which is not there in your XML file that has to be brought from somewhere else. So that somewhere else, once you fetch it the contents of that will be inserted at the point of reference. There is one simple example I am giving. Here I am defining an entity. Entity IIT logo, this is the name of the entity I am giving. I am saying system is a keyword followed by a link which tells you where this logo is found this is the URL. This is actually a URL institute slash logo.gif. So now in my XML wherever I use this IIT logo entity that will get replaced by that gif file gif image. These are very convenient way of replacement and this external entity will allow you to do it from other file not from the same document.

Including a DTD

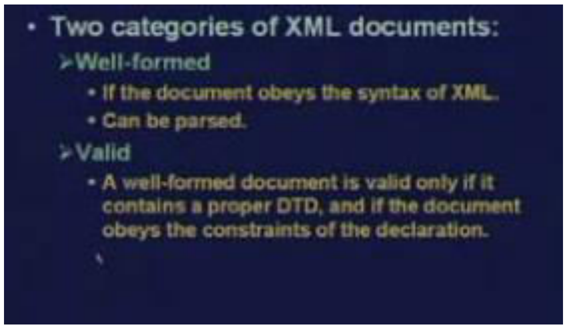


```
<?XML version="1.0" standalone="no" ?>
  <!DOCTYPE chapter SYSTEM "mybook.dtd" [
      .........
      .........
  ]>

<chapter>
  ........
  ........
</chapter>
```

Sometimes we may want to include a document type description definition along with an XML document. There is a way to do this first is that in the prolog you specify that this is not a standalone document. Stand-alone equal to yes is the default. You say that this document depends on some other document for proper interpretation. So this is not a standalone document. So standalone you say no and here you specify the corresponding document type definition DTD. This is the way to specify this doctype chapter system this mybook.dtd is the file where all the description of elements and attributes are present. So all the definition you have put there in one place in a separate file and in this XML file you are including them instead of repeating the definition every time in every XML you can simply include them as a DTD. So this is the syntax and after this your chapter type gets defined just like a document type. This is a document type of chapter. So now we start with begin chapter, end chapter, in between whatever tags you have defined you can use them in between. This is just an indication of how you can use a DTD.

Validity of XML Document



- Two categories of XML documents:
  - ➢Well-formed
    - If the document obeys the syntax of XML.
    - Can be parsed.
  - ➢Valid
    - A well-formed document is valid only if it contains a proper DTD, and if the document obeys the constraints of the declaration.

Now XML documents can be categorized as well formed. If it obeys all the syntax of html of XML and if it obeys the syntax it can be parsed. You can have another level of you can say well form ness you can say valid. Valid says it is it will be formed no doubt. But it will also contain a proper DTD and the document follows all the constraints of the DTD. So it is something more just the syntax. Some semantic is also included then you call it a valid document.

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	What is DTD? Why do we use it with XML documents? Explain the different components of DTD with example.	June.2014	7
Q.2	What is XML DTD (Document Type Definition)? What is the advantage of having a DTD for an XML document?	June.2012	10

UNIT-02/LECTURE-06

Basics of Cascading Style Sheet (CSS) : [RGPV/Jun 2012(10), Jun 2013(20)]

Introduction to Cascading Style Sheets

Cascading Style Sheets (CSS) is a slightly misleading term, since a website might have only one CSS file (style sheet), or the CSS might be embedded within an HTML file. It is better to think of CSS as a technology (in the singular). CSS is comprised of statements that control the styling of HTML documents. Simply put, an HTML document should convey content. A CSS document should control the styling of that content.

<div align="center"></div> <img src="this.gif" border="0" alt="" /> <table height="200">... <td width="30"></td>

All these examples can easily be replaced with CSS. Don't worry if you don't understand these declarations yet.

div {text-align: center;} img {border: 0 none;} table {height: 200px;} td {width: 30px;}

An HTML file points to one or more external style sheets (or in some cases a list of declarations embedded within the head of the HTML file) which then controls the style of the HTML document. These style declarations are called CSS rules.

Features

The latest version of Cascade Style Sheets, CSS 3, was developed to make Web design easier but it became a hot topic for a while because not all browsers supported it. However, trends change quickly in technology and all browser makers currently are implementing complete CSS 3 support. Making that process easier for the browser manufacturers is CSS 3's modularized specification, which allows them to provide support for modules incrementally without having to perform major refactoring of the browsers' codebases. The modularization concept not only makes the process of approving individual CSS 3 modules easier and faster, but it also makes documenting the spec easier.

Eventually, CSS 3 -- along with HTML5 -- are going to be the future of the web. You should begin making your Web pages compatible with these latest specifications. In this article, I explore 10 of the exciting new features in CSS 3, which is going to change the way developers who used CSS2 build websites.Some of the features are:

- CSS Text Shadow
- CSS Selectors
- CSS Rounded Corners

we dont take any liability for the notes correctness. http://www.rgpvonline.com

o CSS Border Image

Core Syntax

1 At-Rules

As we learned when we studied CSS statements, there are two types of statements. The most common is the rule-sets statement, and the other is the at-rules statement. As opposed to rule sets, at-rules statements consist of three things: the at-keyword, @, an identifier, and a declaration. This declaration is defined as all content contained within a set of curly braces, or by all content up until the next semicolon.

@import

Perhaps the most commonly used of the at-rules, @import, is used to import an external style sheet into a document. It can be used to replace the LINK element, and serves the same function, except that imported style sheets have a lower weight (due to having less proximity) than linked style sheets.

```
<style type="text/css" media="screen"> @import url(imported.css); </style>
@import url(addonstyles.css); @import "addonstyles.css";
```

Relative and absolute URLs are allowed, but only one is allowed per instance of @import. One or more comma-separated target media may be used here.

@charset

@charset is used to specify the character encoding of a document, and must appear no more than once. It must be the very first declaration in the external style sheet, and cannot appear in embedded style sheets. @charset is used by XML documents to define a character set.

```
@charset "utf-8";
```

@namespace

The @namespace rule allows the declaration of a namespace prefix to be used by selectors in a style sheet. If the optional namespace prefix is omitted, then the URL provided becomes the default namespace. Any @namespace rules in a style sheet must come after all @import and @charset at-rules, and come before all CSS rule-sets.

```
@namespace foo url("http://www.example.com/");
```

@namespace can be used together with the new CSS3 selectors (see below). It defines which XML namespace to use in the CSS. If the XML document doesn't have matching XML namespace information, the CSS is ignored.

@font-face

This was removed from the CSS2.1 specification, but is still used to describe a font face for a document..

@font-face { font-family: "Scarborough Light"; src: url("http://www.font.com/scarborough-lt"); }

@font-face { font-family: Santiago; src: local ("Santiago"), url("http://www.font.com/santiago.tt"), format("truetype"); unicode-range: U+??,U+100-220; font-size: all; font-family: sans-serif; }

@media

This at-rule is used within a style sheet to target specific media. For example, after defining how an element is to be displayed (in this example for the screen), the declaration can be overwritten for print, in which case we often want to hide navigation.

p {font-size: 0.8em;} /* for the screen */ @media print { p {font-size: 10pt;} #nav, #footer {display: none;} } @media screen, handheld { p {font-size: 14px; text-align: justify;} }

@page

This at-rules declaration is used to define rules for page sizing and orientation rules for printing.

@page {size: 15cm 20cm; margin: 3cm; marks: cross;}

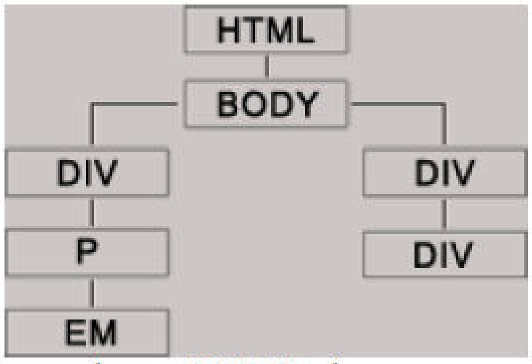
You may specify how pages will format if they are first, on the left-hand side, or on the right.

@page :first {margin-top: 12cm;} @page :left {margin-left: 4.5cm;} @page :right {margin-right: 7cm;}

@fontdef

This is an old Netscape-specific at-rule which we should ignore.

CSS1 Selectors



Selectors refer to elements in an HTML document tree. Using CSS, they are pattern-matched in order to apply styles to those elements. A selector consists of one or more elements, classes, or IDs, and may also contain pseudo-elements and/or pseudo-classes.

Type Selector

The type selector is the simplest selector of all, and matches all occurrences of an element. In this example, all <p> tags throughout the document will have the following style applied, unless overridden.

p {color: #666;}

Universal Selector

The universal selector, used alone, matches all elements in the document tree, and thus will apply styles to all elements. It is in effect a wildcard.

* {margin: 0; padding: 0;}

In this example, all tags are reset to have no padding or margin. This, by the way, is a practice to gain control over all the default padding and margin inherent in the way User Agents (UAs) display HTML.

Class Selector

The class selector matches a classname.

.largeFont {font-size: 1.5em;} h3.cartHeader {text-align: center;}

The "largeFont" class will apply to all elements into which it is called. The "cartHeader" class will only function as styled if called into an H3 element. This is useful if you have another "cartHeader" declaration that you wish to override in the context of an H3 element, or if you wish to enforce the placement of this class.

ID Selector

The ID selector matches an ID. IDs are identifiers unique to a page. They bear a resemblance to classes, but are used a bit differently. IDs will be treated more fully below. The first two ID examples below refer to sections of a web page, while the last refers to a specific occurrence of an item, say, an image in a DHTML menu. IDs have a higher specificity than classes.

#header {height: 100px;} #footer {color: #F00;} #xyz123 {font-size: 9px;}

Descendant Selector

A selector can itself be a chain of one or more selectors, and is thus sometimes called a compound selector. The descendant selector is the only compound selector in CSS1, and consists of two or more selectors and one or more white space combinators. In the example below, the white space between the H1 and EM elements is the descendant combinator. In other words, white space conveys a hierarchy. (If a comma were to have intervened instead, it would mean that we were styling H1 and EM elements alike.) Selectors using combinators are used for more precise drill-down to specific points within the document tree. In this example <em> tags will have the color red applied to them if they are within an <h1> tag.

h1 em {color: #F00;}

Note that EM elements do not have to be immediately inside an H1 heading, that is, they do not have to be children, but merely descendants of their ancestor. The previous style would apply to an EM element in either of the following statements.

<h1>This is a <em>main</em> heading</h1> <h1>This is <strong>another <em>main</em> heading</strong></h1>

In the next example, the color black will be applied to all <span> tags that are descendants (whether directly or not) of <div> tags which are in turn descendants (whether directly or not) of <p> tags, no matter how deep the <p> tags are in the document tree.

div p span {color: #000;}

That is to say, this style would apply to SPAN elements inside a P element, even if they are many levels below (that is, within) the DIV element, as long as there is an intervening P element.

The universal selector can be part of a compound selector in tandem with a combinator.

p * span {font-size: 0.6em;}

This would style any SPAN element that is at least a grandchild of a P element. The SPAN element could in fact be much deeper, but it will not be styled by this declaration if it is the child (direct descendant) of a P element.

Other Selectors

Other combinators convey greater precision. They include the direct adjacent sibling combinator (+), the indirect adjacent sibling (or general) combinator (~), and the child combinator (>). These combinators will be treated below because they are part of the CSS2.1 specification, and are not supported in IE6.

EXAMPLE:

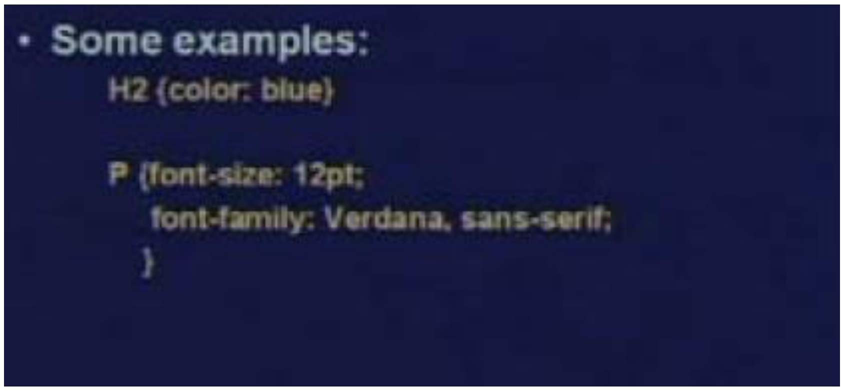
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html>
<head>
<style type="text/css"> div.ex
{
```

```
width:220px;
padding:10px; border:5px solid gray;
margin:0px;
}
</style>
</head>
```

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	What do you understand by ‘style’? What are the benefits of using styles compared with placing formatting directly into the text of webpage? State the different styles which can be applied to HTML document.	June.2012	10

UNIT-02/LECTURE-07

How to specify rules of Cascading Style Sheet: [RGPV/Dec 2010(10),Jun 2011(10)]



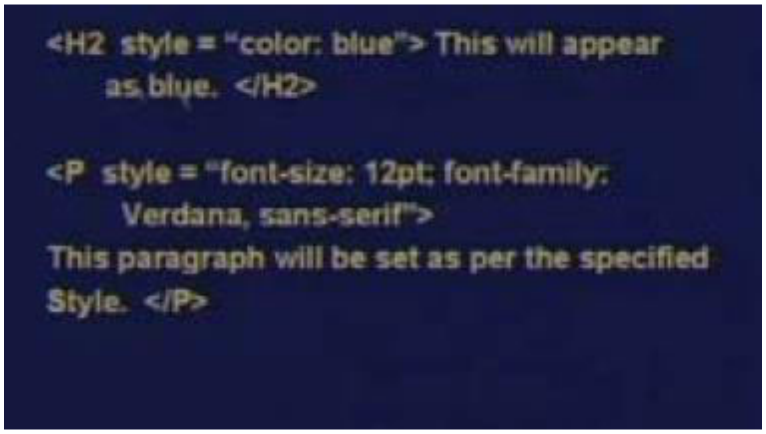
• Some examples:

```
H2 {color: blue}

P {font-size: 12pt;
   font-family: Verdana, sans-serif;
   }
```

Some examples see h2 is a standard tag for heading; second level heading. But you can within curly bracket you can specify like this with in bracket color colon blue. This means I am specifying some rule for the h2 tag. Similarly for paragraph p, I can specify font size 12, font family verdana sans serif, I can specify. So subsequently if I use the p tag; paragraph tag and text in the paragraph tag would be composed according to these rules. So this is how we can specify rules in html which is beyond the definition of you can say the default definition of html.

Inline Style Sheet

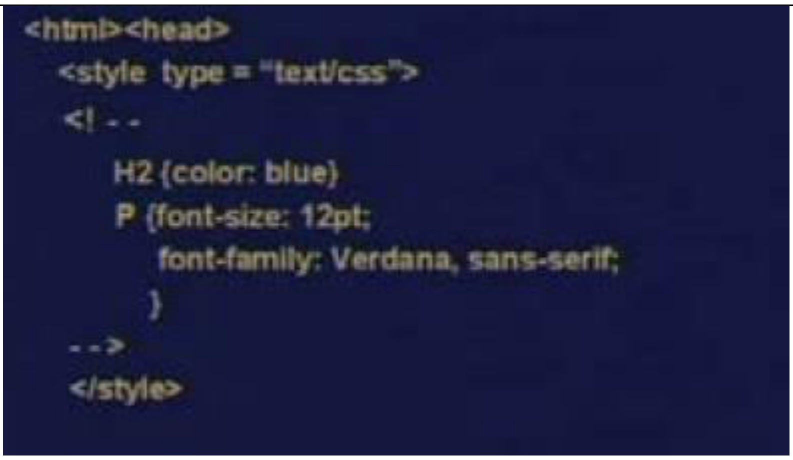


```
<H2 style = "color: blue"> This will appear
    as blue. </H2>

<P style = "font-size: 12pt; font-family:
     Verdana, sans-serif">
This paragraph will be set as per the specified
Style. </P>
```

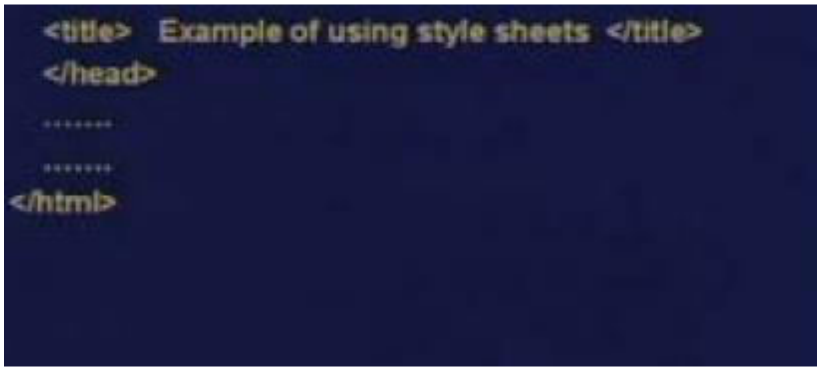
There is some called inline styles which you can define as part of the html document as this example shows. This is the header. In the begin header using the style attribute you can specify color colon blue. So this heading will be appearing blue. Similarly for paragraph you can specify by style. Again the exact font in which you want this paragraph to appear. So it will appear like this. Since this is part of the same document same form you call it inline.

Embedded Style Sheet



```
<html><head>
  <style type = "text/css">
  <! - -
      H2 {color: blue}
      P {font-size: 12pt;
          font-family: Verdana, sans-serif;
         }
  - - >
  </style>
```

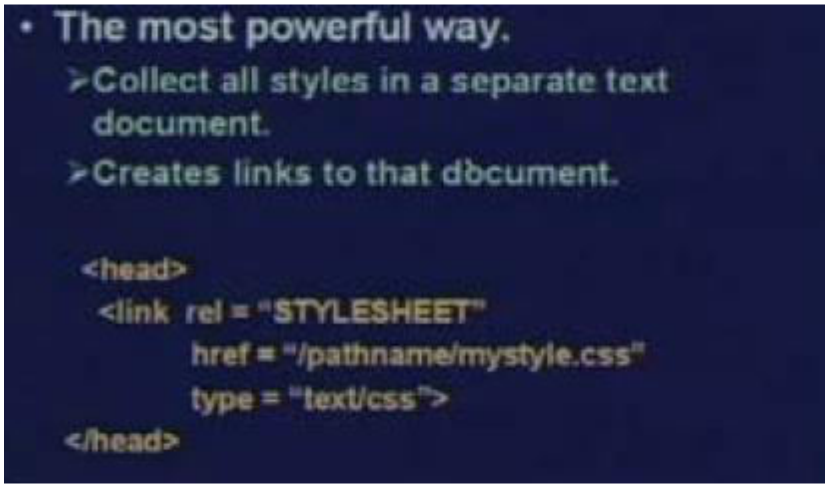
The embedded style sheets are also possible. Embedded style sheet means you have a begin style end style tag. So you give some name or some type to this style. This is a text cascaded style sheet type and this is a special marker. This is the beginning and this is the ending and within this you can define all the styles you want to define. The advantage of this style of definition is that all the definitions you can put in one place. Not only in one place; you can even store in some other file. You can import that file from a main html file.



```
  <title>   Example of using style sheets  </title>
  </head>
  ........
  ........
</html>
```

So after this when you have the main file. You can use those h2; those p, this tag, as part of your document.

External Style Sheet



- The most powerful way.
  - ➢Collect all styles in a separate text document.
  - ➢Creates links to that document.

```
  <head>
    <link rel = "STYLESHEET"
          href = "/pathname/mystyle.css"
          type = "text/css">
  </head>
```

An external style sheet as I said, this is exactly what I told you that we can define all this style together and store it somewhere else in some other file. This is external style sheet. You collect all the styles in a separate document and you create links to that document. There is a link tag using which you can link a style sheet. So you can specify in which file this style sheet is mentioned of course type is text CSS. These are some optional parameters you can give but this is how you can specify. Once you give at the beginning so after that you can give your html document and description after that.

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	What is the use of CSS in DHTML? Explain the methods of using CSS with HTML. Write a suitable program to demonstrate.	June.2013	20
Q.2	Define Cascading Style Sheet(CSS)	Dec.2010	10
Q.3	Explain Extensible Markup Language (XML). Also explain cascading style sheet(CSS)	June.2011	10

UNIT-02/LECTURE-08

Document object Model (DOM):[RGPV/Jun 2014(14)]

INTRODUCTION TO THE DOCUMENT OBJECT MODEL

The Document Object Model (DOM) is the model that describes how all elements in an HTML page, like input fields, images, paragraphs etc., are related to the topmost structure: the document itself. By calling the element by its proper DOM name, we can influence it.The Document Object Model, or DOM, is the interface that allows you to programmatically access and manipulate the contents of a web page (or document). It provides a structured, object-oriented representation of the individual elements and content in a page with methods for retrieving and setting the properties of those objects. It also provides methods for adding and removing such objects, allowing you to create dynamic content.The DOM also provides an interface for dealing with events, allowing you to capture and respond to user or browser actions. This feature is briefly covered here but the details are saved for another article. For this one, the discussion will be on the DOM representation of a document and the methods it provides to access those objects.

Nodes

The DOM is a Document Object Model, a model of how the various objects of a document are related to each other. In the Level 1 DOM, each object, whatever it may be exactly, is a Node. So if you do

<P>This is a paragraph</P>

you have created two nodes: an element P and a text node with content 'This is a paragraph'. The text node is inside the element, so it is considered a child node of the element. Conversely, the element is considered the parent node of the text node.

```
<P> <-- element node
|
|
This is a paragraph <-- text node
```

If you do

<P>This is a <B>paragraph</B></P>

the element node P has two children, one of which has a child of its own:

```
          <P>
```

```
            |
      ------------------
      |                |
   This is a       <B>
                    |
                    |
            paragraph
```

Finally there are attribute nodes. (Confusingly, they are not counted as children of an element node. In fact, while writing this pages I've done a few tests that seem to indicate that Explorer 5 on Windows doesn't see attributes as nodes at all.) So

<P ALIGN="right">This is a <B>paragraph</B></P>

would give something like

```
   <P> ----------------
      |                |
--------------       ALIGN
|         |          |
This is a    <B>          |
              |          right
              |
       paragraph
```

So these are element nodes, text nodes and attribute nodes. They constitute about 99% of the content of an HTML page and you'll usually busy yourself with manipulating them. There are more kinds of nodes, but I skip them for the moment.
As you'll understand, the element node P also has its own parent, this is usually the document, sometimes another element like a DIV. So the whole HTML document can be seen as a tree consisting of a lot of nodes, most of them having child nodes (and these, too, can have children).

```
<BODY>
|
|-------------------------------------
|                                    |
<P> ----------------          lots more nodes
|                   |
--------------      ALIGN
|         |         |
This is a    <B>    |
          |        right
          |
paragraph
```

Walking through the DOM tree:[RGPV/Jun 2010(4)]

Knowing the exact structure of the DOM tree, you can walk through it in search of the element you want to influence. For instance, assume the element node P has been stored in the variable x (later on I'll explain how you do this). Then if we want to access the BODY we do

x.parentNode

We take the parent node of x and do something with it. To reach the B node:

x.childNodes[1]

childNodes is an array that contains all children of the node x. Of course numbering starts at zero, so childNodes[0] is the text node 'This is a' and childNodes[1] is the element node B.

Two special cases: x.firstChild accesses the first child of x (the text node), while x.lastChild accesses the last child of x (the element node B).

So supposing the P is the first child of the body, which in turn is the first child of the document, you can reach the element node B by either of these commands:

```
document.firstChild.firstChild.lastChild;
document.firstChild.childNodes[0].lastChild;
document.firstChild.childNodes[0].childNodes[1]; etc.
document.firstChild.childNodes[0].parentNode.firstChild.childNodes[1];
```

DOM Levels and history

A short history lesson in order. When JavaScript was first introduced in browsers, some sort of interface was required to allow elements on the page to be accessed via scripting. Each vendor had their own implementation but de facto standards emerged to produce a fairly simple model common to most.For example, most browsers used an array of Image objects to represent all IMG tags on a page. These could then be accessed and manipulated via JavaScript. A simple image rollover might use code like this:

```
document.images[3].src = "graphics/button2.gif"
```

These early models were limited. They only provided access to a few types of element and attributes, like images, links or form elements.As vendors released new versions of browsers, they expanded on the model. But they also were often at odds with one another, leading to compatibility problems among different browsers as vendors tried to outdo each other by adding their own new features.Fortunately, most vendors started adopting the DOM standard set by the World Wide Web Consortium, notably Internet Explorer, Netscape and Opera.

In order to provide some backward compatibility, different levels of the DOM standard are defined. You might find references to DOM Level 0 (or "DOM0") which corresponds to the model used by the first, scriptable browsers - mainly Internet Explorer and Netscape prior to version 4 of each. Then there is DOM1 which was adopted in 1998 and covers some of the features introduced in version 4 browsers.Most of the current browsers (version 5 and up) support the DOM2 standard, or at least a good part of it. They may also continue to support some features of the earlier DOM levels, or their own proprietary extensions, so that they are compatible with older web pages.

This article will focus just the current, DOM2 standard. While this standard applies to XML (extended markup language) documents in general, it discusses how the standard applies to HTML in particular (it being a subset of XML).The good news is that, given the current industry trend, you can expect future versions of browsers to follow this standard. The bad news is that for now, you may find it difficult to code pages that work with both old and new browsers.

One such example is Netscape 6.0, which drops support for many of the proprietary features of Netscape 4, such as the LAYER tag and its corresponding Layer object. They are not recognized in version 6 as they were never adopted as part of the standard.

Also note that Internet Explorer's document.all construct is a proprietary feature, not part of any standard. While it may be supported in many versions of IE - even the latest version - it's generally not supported by other browsers. You should keep in mind that the DOM coding is also indirectly dependent on the standards for HTML and CSS, since the DOM reflects the tags and attributes defined by those standards. It also depends on a standard for JavaScript since the DOM is essentially and API for client-side scripting.

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	Explain in detail DOM event handling. Also explain with an example of creating a context menu	June.2014	14
Q.2	How can you target an element in your HTML using the DOM?	June.2010	4

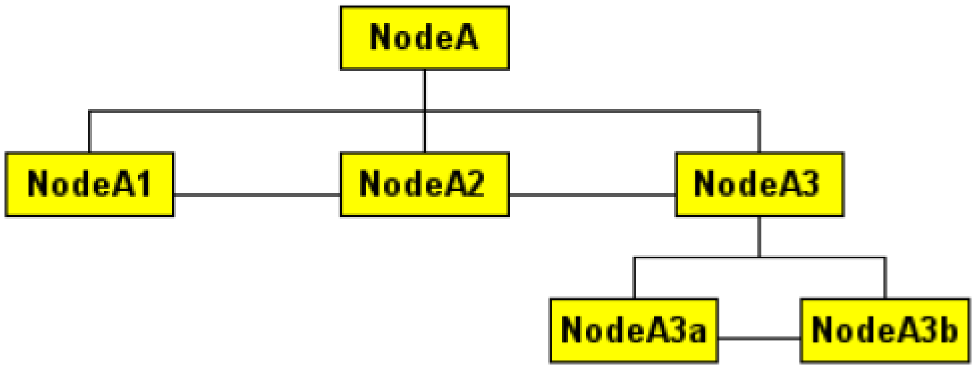
UNIT-02/LECTURE-09

Document tree:[RGPV/Jun 2011(10)]

When a browser loads a page, it creates a hierarchical representation of its contents which closely resembles its HTML structure. This results in a tree-like organization of nodes, each representing an element, an attribute, content or some other object

Nodes

Each of these different object types will have their own unique methods and properties. But each also implements the Node interface. This is a common set of methods and properties related to the document tree structure. To understand this interface better, take a look a the diagram below which represents a simple node tree.



The Node object provides several properties to reflect this structure and allow you to traverse the tree. Here are some relationships using the example above:

- NodeA.firstChild = NodeA1
- NodeA.lastChild = NodeA3
- NodeA.childNodes.length = 3
- NodeA.childNodes[0] = NodeA1
- NodeA.childNodes[1] = NodeA2
- NodeA.childNodes[2] = NodeA3

The Node interface also provides methods for dynamically adding, updating and removing nodes such as:

- insertBefore()
- replaceChild()
- removeChild()
- appendChild()
- cloneNode()

These will be covered later. But for now let's look how the document tree reflects the contents of a web page.

The Document Root

The document object serves as the root of this node tree. It too implements the Node interface. It will have child nodes but no parent node or sibling nodes, as it is the starting node. In addition to being a Node, it also implements the Document interface.

This interface provides methods for accessing and creating other nodes in the document tree. Some methods are:

- getElementById()
- getElementsByTagName()
- createElement()
- createAttribute()
- createTextNode()

Note that unlike other nodes, there is only one document object in a page. All of the above methods (except getElementsByTagName()) can only be used against the document object, i.e., using the syntax document.methodName().These properties are intended to provide some backward compatibility so that pages designed for older browsers will still work properly with newer versions. They can still be used in scripts but they may not be supported in future versions.

Traversing the Document Tree

As mentioned, the document tree reflects the structure of a page's HTML code. Every tag or tag pair is represented by a element node with other nodes representing attributes or character data (i.e., text).

Technically speaking, the document object has only one child element, given by document.documentElement. For web pages, this represents the outer HTML tag and it acts as the root element of the document tree. It will have child elements for HEAD and BODY tags which in turn will have other child elements.

Using this, and the methods of the Node interface, you can traverse the document tree to access individual node within the tree. Consider the following:

```
<html>   <head> <title></title> </head>

<body><p>This is a sample paragraph.</p></body> </html>
```

and this code:

```
alert(document.documentElement.lastChild.firstChild.tagName);
```

which would display "P", the name of the tag represented by that node. The code breaks down as follows:

- document.documentElement - gives the page's HTML tag.
- .lastChild - gives the BODY tag.
- .firstChild - gives the first element in the BODY.
- .tagName - gives that element's tag name, "P" in this case.

There are some obvious problems with accessing nodes like this. For one, a simple change to the page source, like adding text or formatting elements or images, will change the tree structure. The path used before may no longer point to the intended node.

Less obvious are some browser compatibility issues. Note that in the sample HTML above, there is no spacing between the BODY tag and the P tag. If some simple line breaks are added,

```
<html>
<head>
<title></title>
</head>
<body>

<p>This is a sample paragraph.</p>

</body>
</html>
```

Netscape will add a node for this data, while IE will not. So in Netscape, the JavaScript code shown above would display "undefined" as it now points to the text node for this white space. Since it's not an element node, it has no tag name. IE, on the other hand, does not add nodes for white space like this, so it would still point to the P tag.

Accessing Elements Directly

This is where the document.getElementById() method comes in handy. By adding an ID attribute to the paragraph tag (or any tag for that matter), you can reference it directly.

```
<p id="myParagraph">This is a sample paragraph.</p>

...

alert(document.getElementById("myParagraph").tagName);
```

This way, you can avoid compatibility issues and update the page contents at will without worrying about where the node for the paragraph tag is in the document tree. Just remember that each ID needs to be unique to the page.

A less direct method to access element nodes is provided by document.getElementsByTagName(). This returns an array of nodes representing all of the elements on a page with the specified HTML tag. For example, you could change color of every

link on a page with the following.
var nodeList = document.getElementsByTagName("A");
for (var i = 0; i < nodeList.length; i++) nodeList[i].style.color = "#ff0000";
Which simply updates each link's inline style to set the color parameter to red. Give it a try.

Node Types

Before going further, it's probably a good time to explain node types in more detail. As mentioned, there are several types of nodes defined in the document object model, but the ones you'll mostly deal with for web pages are element, text and attribute.
Element nodes, as we've seen, correspond to individual tags or tag pairs in the HTML code. They can have child nodes, which may be other elements or text nodes.
Text nodes represent content, or character data. They will have a parent node and possibly sibling nodes, but they cannot have child nodes.
Attribute nodes are a special case. They are not considered a part of the document tree - they do not have a parent, children or siblings. Instead, they are used to allow access to an element node's attributes. That is, they represent the attributes defined in an element's HTML tag, such as the HREF attribute of the A tag or the SRC attribute on the IMG tag.

Note that attribute values are always text strings.

Attributes vs. Attribute Nodes
There are several ways to reference the attributes of an element. The reason for this is that the DOM2 standard was designed for many types of structured documents (i.e., XML documents), not just HTML. So it defines a formal node type for attributes.

But for all documents it also provides some more convenient methods for accessing, adding and modifying attributes, as described next.

The document.createAttribute() allows you to create a new attribute node, which you can then set a value for and assign to an element node.

var attr = document.createAttribute("myAttribute"); attr.value = "myValue";

var el = document.getElementById("myParagraph"); el.setAttributeNode(attr);

However, it's usually easier to access an element's attributes directly using the element getAttribute() and setAttribute() methods instead.

var el = document.getElementById("myParagraph");
el.setAttribute("myAttribute", "myValue");

An element's attributes are also represented as properties of the element node. In other words, you can simply use
var el = document.getElementById("myParagraph"); el.myAttribute = "myValue";

It's also interesting to note that you can define your own attributes in the HTML tag itself. For example,

<p id="myParagraph" myAttribute="myValue">This is a sample paragraph.</p>

we dont take any liability for the notes correctness. http://www.rgpvonline.com

...
```
alert(document.getElementById("myParagraph").getAttribute("myAttribute"));
```

will display "myAttribute." But note that you should use element.getAttribute(attributeName) instead of element.attributeName to get the value as some browsers may not register user-defined attributes as a properties of the element.

Attributes can also be removed from an element node, using either the removeAttribute() or removeAttributeNode() methods or setting by setting element.attributeName to a null string ("").

Altering attributes is one way to create dynamic effects. Below is a sample paragraph. Use the links to alter it's ALIGN attribute.

Text in a paragraph element.

Align Left | Align Right

The code is fairly simple:

```
<p id="sample1" align="left">Text in a paragraph element.</p>
```

... code for the links ...

```
document.getElementById('sample1').setAttribute('align','left');
document.getElementById('sample1').setAttribute('align', 'right');
```

Style Attributes

Most attributes for HTML tags are fairly simple, they define a single value for a property specific to that tag. Styles are a little more involved. As you know, CSS can be used to apply style parameters to an individual tag, all tags of a given type or assigned using
classes. Likewise, styles for a particular element can be inherited from any of these sources.

You can also alter these styles at various levels. For example, you can change the STYLE attribute of an HTML tag, or it's CLASS attribute. But these methods will alter all of the element's style parameters. Often, you may want to change just a single style parameter, or a select few, while retaining the others.

Fortunately, the style attribute of and element node is defined as an object with properties for every possible style parameter. You can access and update these individual parameters as you wish. Here's an example similar to the previous one.

Text in a paragraph element.

Align Left | Align Right

But in this case, the text alignment is defined and altered using a style parameter instead of the ALIGN attribute. Here's the code behind it:

```
<p id="sample2" style="text-align:left;">Text in a paragraph element.</p>

... code for the links ...

document.getElementById('sample2').style.textAlign='left';
document.getElementById('sample2').style.textAlign = 'right';
```

DOM event handling: [RGPV/Dec 2010(10)]

One of the keys to creating dynamic web pages is the use of event handlers. These allow you to execute specific script code in response to user or system initiated actions.Most events relate to the browser GUI, such as mouse movements, button or key clicks and updates to form inputs. These are usually tied to a specific page element. Others relate to browser actions such as when a document or image completes loading.Some objects have default actions defined for certain events, such as clicking on a hypertext link. The browser's normal action in that event is to load the URL associated with the link.

In any case, all events follow the same model. The DOM provides methods for capturing events so you can perform your own actions in response to them. It also provides an Event object which contains information specific to a given event that can be used by your event processing code.

Assigning Event Handlers

There are several ways to set up the capture of an event on an object using either HTML or scripting. In each case, you assign a function to handle the specific event when it occurs.

HTML Tag Attributes

Many HTML tags have intrinsic events associated with them. You can define script code to be executed when one of these events occurs using the event name as an attribute. For example,

```
<span style="background-color:yellow;"
onmouseover="this.style.backgroundColor='black';this.style.color='white'"
onmouseout="this.style.backgroundColor='yellow';this.style.color=''"> Sample element with
mouse event handlers.
</span>
```

Here, two different events are captured, mouseover and mouseout. the value assigned to the corresponding attributes is a string containing JavaScript code. Note the use of single quotes (') for string constants within the double quotes (") used to delimit the attribute value.The script code for both events simply changes the text and background colors on the element's style. The keyword this keyword refers to the object that fired the event. Here it refers to the SPAN element.It was stated before that when you set up capturing for an event, you assign a function to be called to handle that event. This seems to contradict the example above where the event code is simply a couple of JavaScript statements. However, the browser actually creates an anonymous function for the handler with those statements in it.

You should see an alert box with a function definition. The actual definition will vary according to browser but it will contain a single statement, the one defined in the HTML:

```
function name(argument_list) { alert(this.onclick);
}
```

This corresponds to the code set on the ONCLICK attribute:

```
<span style="background-color:yellow;" onclick="alert(this.onclick)">Sample element with an onclick event handler.</span>
```

Scripting

You can also assign event handlers to elements in the DOM using client-side scripting. Like other element attributes, events are represented as properties of the element object

so you can set their value just like any other attribute.The main difference is that unlike most attributes, which take a string value, event handlers must be set using a function reference. Here's an example,

```
<span id="sample1" style="background-color:yellow;">Sample element with mouse event handlers.</span>
... JavaScript code ...
function highlight(event) {
this.style.backgroundColor='black';
this.style.color='white';
}
```

```
function normal(event) {
this.style.backgroundColor='yellow';
this.style.color='';
}
document.getElementById('sample1').onmouseover=highlight;
document.getElementById('sample1').onmouseout = normal;
```

Note that the onmouseover property is set to highlight, i.e., the name of the function without parenthesis. This represents the Function object for highlight which is what the event property expects, a reference to the function to call when the event is fired.If highlight() had been used instead, the browser would call the function and assign whatever value it returned as the event handler. In this particular case, an error would result in the function execution because this has no meaning in the given context.

Event Listeners

DOM objects can also be registered as event listeners. This feature can be used to assign multiple handlers for a given event. It also allows you to capture events during either one of two phases in the event flow, capture or bubble.

The difference between these two phases will be covered later on in the discussion on event flow. But for now we'll just look at how to register an event listener.

The basic methods and syntax are

```
node.addEventListener(eventType,function,useCapture);
node.removeEventListener(eventType, function);
```

where eventType is a predefined event name such as "mouseover" or "click" (the same as the corresponding event attribute but without the preceding "on"), function is the name of the handler function and useCapture is a boolean flag which specifies which phase of the event flow the handler will be called on.

To demonstrate, the previous example could have assigned event handlers for the sample paragraph element with the following:

```
var el = document.getElementById('sample1'); el.addEventListener("mouseover", highlight, false); el.addEventListener("mouseout", normal, false);
```

Additional handlers could also be assigned for a given event:

```
el.addEventListener("mouseover",highlight2,true);
el.addEventListener("mouseover", highlight3, false);
```

Individual event listeners can be subsequently removed by specifying the same arguments:

el.removeEventListener("mouseover", highlight2, true); el.removeEventListener("mouseover", highlight3, false);

Note that you must specify the same useCapture value when removing a listener as when it was added. This is because,

el.addEventListener("mouseover", highlight, true); el.addEventListener("mouseover", highlight, false);

defines two unique event listeners, calling the same function for the same event on the same element but which activate during different phases.

One advantage of this method is that event listeners can be assigned to any node, even text nodes which you could not normally assign event handlers to as they have no attributes.

Event Flow

Before going into event processing, it's helpful to understand event flow in the DOM. HTML documents (and XML or XHTML documents) are hierarchical in nature. lements and text are nested within other elements. Because of this, when an event occurs on a particular object, it effectively occurs on any containing objects as well.

To illustrate, consider the following HTML:

<div>

<p>Some text in a paragraph element.</p> <p>Another paragraph element with text.</p> <p>Yet another paragraph with text and also a

<a href="blank.html">link</a>.</p> </div>

If you click on the link defined in this code sample, it will trigger a onclick event on the A tag. But you're also clicking on the paragraph that contains that link, and the DIV that contains that P element and so on up to the document object itself.Any of the elements in this chain can have an event handler assigned to it to capture the onclick event, regardless of which element it originated at.

Event Bubbling

The DOM event model provides for this using a concept called event bubbling. For example, suppose an onclick event handler were assigned to the DIV tag above. Clicking on the link would fire the event first on the A element, it would then "bubble" up to the containing P element and then up to the DIV where the handler function would be called.

It's possible for the handler to cancel the event, but assuming it doesn't the event would continue bubbling on up to the document root and finally, the browser would follow the default action of loading the link's URL in the window.Note that the P element could also have had an onclick event handler set, as could any elements above the DIV in the document tree. All of these handlers would be called in turn as the event bubbles up to the document root.This is known as the bubble phase in the DOM event model. Not all events bubble, for example onfocus and onblur do not. Likewise, not all bubbling events can be canceled, stopping the propagation. You can determine which events bubble and can be canceled either by looking up the documentation for the event or, as we'll see, using the Event object.

Event Capture

You can also catch events during the capture phase using the event listener detailed previously. The capture phase compliments the bubble phase. The capture phase runs first. The event flows down from the root of the document tree to the target element, then it bubbles back up.

In this phase, outer elements will receive the event before it reaches its intended target. This can be useful if you want to intercept an event for some element even if it was initially targeted at one of its children or other descendants.

It should be noted that the term "event capture" is often used loosely to describe the act of setting an event handler or listener, during either phase. Here it specifically means intercepting an event during this downward phase of the event flow, before it reaches its intended target.

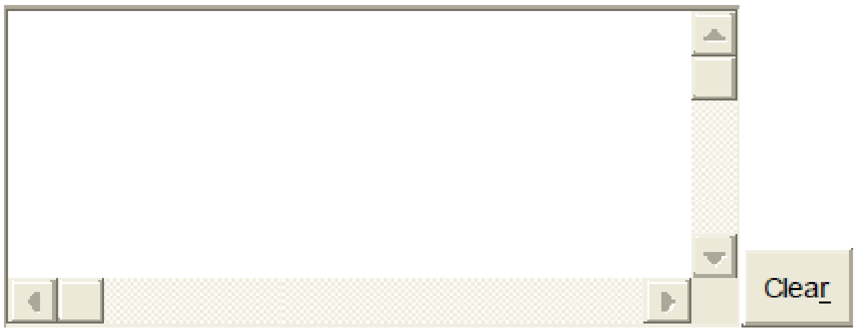
Event Flow Example

The full process can be illustrated with a demo. Below is a set of nested DIV tags The onclick event is caught for each element both on the capture phase (if supported by the browser) and the bubble phase. Click anywhere on the boxes and the path of the event will be traced in the text box below it.

DIV A

DIV B

DIV C



Clicking on the innermost DIV fires six event handlers, three going down the document tree and three as it bubble back up. In this example, the same function happens to be used for all six. but each could be assigned a unique handler function.

In the trace, the "Target" element is the one that the event initiates from while the "Current" element is the one that has the event listener attached. Both of these values are derived using the Event object passed to the handler.

The Event Object

Within an event handler, you can do pretty much anything you want with your script code. Chances are, you'll want to perform some action related to that event depending on one or more factors.Recall that event handlers are passed one argument, an Event object. It provides several properties describing the event and its current state. You can use these to determine where an event originated from and where it currently is in the event flow. Or use the methods it provides to stop the event from flowing on and/or cancel the event.

Mouse Events

Mouse-related events include:

- click
- mousedown
- mouseup
- mouseover

- mouseout
- mousemove

For these events, additional information is provided in the Event object.

Mouse Event Properties

Property Name	Description
altKey, ctrlKey, metaKey, shiftKey	Boolean values. If true, the associated key was depressed when the event fired.
Button	An integer indicating which mouse button was pressed or released, 1 = left, 2 = middle, 3 = right. A one-button mouse will return only 1, a two-button mouse could return either 1 or 3.
clientX, clientY	Give the pixel coordinates of the mouse relative to the client area of the browser, i.e., relative to the viewport, when the event fired.
relatedTarget	On a mouseover this indicates the node that the mouse has left. On a mouseout it indicates the node the mouse has moved onto.
screenX, screenY	Give the pixel coordinates of the mouse relative to the screen when the event fired.

As you can see, these properties can be used to determine the position of the mouse, what buttons are depressed and the elements it is moving over or leaving.

Notes on Mouse Events

Mouse events are always taken to occur on the lowest element in the document tree. That is, the target of the event will be the most deeply nested element under the mouse when the event occurs.

Keyboard Events

The DOM2 Event Model does not include specifications for key events. However, the HTML 4 standard does permit the keyup, keydown and keypress events for many elements. Both

Netscape 6 and IE 5.5 support these and include properties in the Event object to reflect information on what key or keys were pressed.In Netscape, the ASCII value of the key is given by the charCode property on keypress events and in the keyCode property for keydown and keyup events.Internet Explorer stores the Unicode value of the key in the event keyCode property for all three key events.

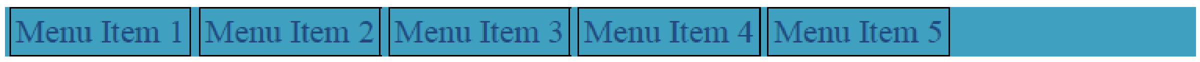
In both, the keydown and keyup events will fire when any modifier key is pressed, such as ALT, CTRL and SHIFT. The keypress event can be used instead capture combinations such as SHIFT-A.Some example key combinations with the relevant property values for each key event type are shown below, arranged by browser.

Handling Events

With dozens of different events constantly firing and moving up and down the document tree, the real trick to using event handlers is in figuring out what event to catch, where to attach the handler and how to process it.

Sometimes the answers are obvious and sometimes they aren't. To demonstrate, let's look at a particular effect and see how to implement it using event handlers.

Example Using Mouse Events



Suppose you wanted to create a pop up menu like the one at right. It consists of a DIV tag with several A tags for the individual items. CSS styles are used to achieve the look including a hover effect on the item links.

The hover effect could have been done using events but why complicate matters when the CSS :hover pseudo-class does the job? Besides, we'll be using events to handle the hiding and showing of the menu.

Here's what the CSS and HTML code look like:

```
.menu { background-color: #40a0c0;
border-color: #80e0ff #006080 #006080 #80e0ff; border-style: solid;
border-width: 2px; position: absolute; left: 0px;
top: 0px; visibility: hidden;
}
a.menuItem { color: #ffffff; cursor: default; display: block;
font-family: MS Sans Serif, Arial, Tahoma,sans-serif; font-size: 8pt;
font-weight: bold;
padding: 2px 12px 2px 8px; text-decoration: none;
}
a.menuItem:hover { background-color: #004060;
color: #ffff00;
}
...
<div id="menuId" class="menu">
<a class="menuItem" href="blank.html">Menu Item 1</a> <a class="menuItem"
href="blank.html">Menu Item 2</a> <a class="menuItem" href="blank.html">Menu Item 3</a>
<a class="menuItem" href="blank.html">Menu Item 4</a> <a class="menuItem"
href="blank.html">Menu Item 5</a> </div>
```

Say we want to make the menu appear at the cursor when the user clicks on a particular link. We set up a function called openMenu() and add an onclick handler on the A tag like this:

```
<a href=""
onclick="openMenu(event, 'menuId'");return false;">Open Menu</a>
```

Notice that openMenu() expects two arguments, an Event object and the ID of the menu DIV to display. Remember that the string value of the ONCLICK attribute is actually used to create an anonymous function which, in this case, would look something like this,

```
function anonymous(event) { openMenu(event, 'menuId'); return false;
}
```

with the menu DIV id hard-coded. So the handler really has only one argument, event, as all event handlers do. Returning false will prevent the browser default action of following the link.

Here's the openMenu() function itself:

```
function openMenu(event, id) {
var el, x, y;
el = document.getElementById(id); if (window.event) {
x = window.event.clientX + document.documentElement.scrollLeft + document.body.scrollLeft;
y = window.event.clientY + document.documentElement.scrollTop + document.body.scrollTop;
}
else {
x = event.clientX + window.scrollX; y = event.clientY + window.scrollY;
}
x -= 2; y -= 2; el.style.left = x + "px"; el.style.top = y + "px";
el.style.visibility = "visible";
}
```

It basically just positions the menu DIV under the cursor and makes it visible. The positioning is complicated a bit by the fact that different methods are needed to determine the cursor position based on the browser.

S.NO	RGPV QUESTION	YEAR	MARKS
Q.1	Explain document object model(DOM) event handling	Dec.2010	10
Q.2	What is document object model? Explain	June.2011	10

REFERENCE

BOOK	**AUTHOR**	**PRIORITY**
Web Technologies- TCP/IP Architecture, and Java Programming	Achyut S. Godbole and Atul Kahate	1
Web Technologies- A computer science perspective	Jeffrey C. Jackson	2

we dont take any liability for the notes correctness. http://www.rgpvonline.com