
UNIT1 WEB 2.0 AND XHTML/HTML5

Structure

- 1.0 Introduction
- 1.1 Objective
- 1.2 Web 2.0 : An Introduction
 - 1.2.1 Search
 - 1.2.2 Content Networks
 - 1.2.3 Blogging
 - 1.2.4 Social Networking
 - 1.2.5 Social Media
 - 1.2.6 Rich Internet Applications(RIA's)
 - 1.2.7 Web Services
 - 1.2.8 Mashups
 - 1.2.9 Widgets and Gadgets
 - 1.2.10 Podcasting, message board and Data Streaming
- 1.3 Introduction to XHTML/HTML5
- 1.4 Syntactic Differences between HTML and XHTML
- 1.5 Standard XHTML/HTML5 Document Structure
- 1.6 Example of HTML5
- 1.7 Summary
- 1.8 Solutions/Answers
- 1.9 Further Readings

1.0 Introduction

Since the advent of World Wide Web (WWW) in 1990, the Internet and its uses are growing rapidly. The Web Programming technologies have changed in the past substantially. You have already been introduced to the process of designing web pages in the course MCSL-016. The collection of web pages is website. The way of writing a web page is web programming. Web development refers to building, creating, and maintaining websites. It includes other aspects such as web design, web publishing, web programming, and database management. A web developer may involve in web designing using languages viz., HTML, PHP and maintaining a database used by dynamic websites. The initial web was focussed on information. There is no or little need to user interaction. With the advent of Web 2.0, there are more than two billion Internet users worldwide. Communication has changed these days, you can exchange your ideas by social networking sites and you can talk to or chat to your friends any time. Blogs, Wikis and Web services are components of Web 2.0. In this unit, you will learn more about web 2.0 concepts and their characteristics, how to create web page using HTML5.

1.1 Objectives

After going through this unit, you should be able to:

- state the concepts applicable to web 2.0

- define the technologies used in web 2.0
- explain about the XHTML/HTML5
- differentiate between XHTML and HTML
- use XHTML/HTML5 tags to create simple static web pages
- add <audio>, <video> tags without the need of flash in HTML5

1.2 Web 2.0: An Introduction

Web 2.0 is a second generation of the World Wide Web. Web 2.0 refers to the transition from static web pages to dynamic web. Web 2.0 is more organized and is based on creating web applications to the specific needs of the users. Web 2.0 has improved communication among the users, with an emphasis on Web-based communities of users and more open sharing of information. Web 2.0 concepts highlight services that allow user to find and manipulate contents.

How is Web 2.0 different?

In early days of Web (Web 1.0), there was no or little need for different web applications to communicate and share data with each other. It was concerned with only creating and viewing online contents. However, web 2.0 has changed this scenario. Web 2.0 offers services that allow user to find and manipulate contents, coupled with all types of media services. An example of web 2.0 is social networking sites, blogs, video sharing, wikis, web application and mashups. The basic idea of each of these websites is user interaction. On blogs, you can post comments, on social networking site, you can make a friend, on social news, you can vote for article and on wikis, you can create, edit and share information. The concept of web 2.0 is based on participation of user. This design encourages user interaction and community contribution.

In client side, web 2.0 provides a user with more interaction, software and storage facilities all through their browsers. The client-side technologies used in Web 2.0 development include Ajax (Asynchronous JavaScript and XML), JavaScript, XML (eXtensible Markup Language-basis for XHTML), SOAP(Simple Object Access Protocol), JSON(JavaScript Object Notation). Ajax is used to create web application using XHTML, CSS, the DOM (Document Object Model) and XMLHttpRequest. You can use Ajax to create more dynamic and interactive Web pages without needing to refresh or reload the page. SOAP is an XML based protocol to allow you to activate an application across the internet. During the user's communication to the web, data is fetched by an Ajax application which is formatted in XML or JSON format, the two widely used structured data formats, the JavaScript program uses the DOM to dynamically update the web page data. For example, Google Docs using this technology to create web based word processor.

In addition to Ajax and JavaScript, Adobe Flex is another technology often used in web 2.0 applications. It is used to populate large data grid, charts and other user interactions. It is used as a plugin component. Application build in flex is displayed as Flash in the browser. Now, HTML5 is capable for doing such things. HTML5 is used for gaming, mobile and business application.

On server side, you can use technologies like PHP, Ruby, ASP.Net and Java Server Pages for developing web applications. Over time Web 2.0 has been used more as a marketing term than a computer-science-based term.

One of the key points of web 2.0 development is the availability of open content access which means that you can modify the content with few or no restrictions. Some of the key web 2.0 websites are Google, YouTube, Twitter, Wikipedia etc. MySpace, Flickr, Wikipedia and YouTube are the websites that provide platforms for users to create and modify the content.

Figure 1 shows the characteristics and techniques of web 2.0. The following subsection explains some of the important web 2.0 technologies.

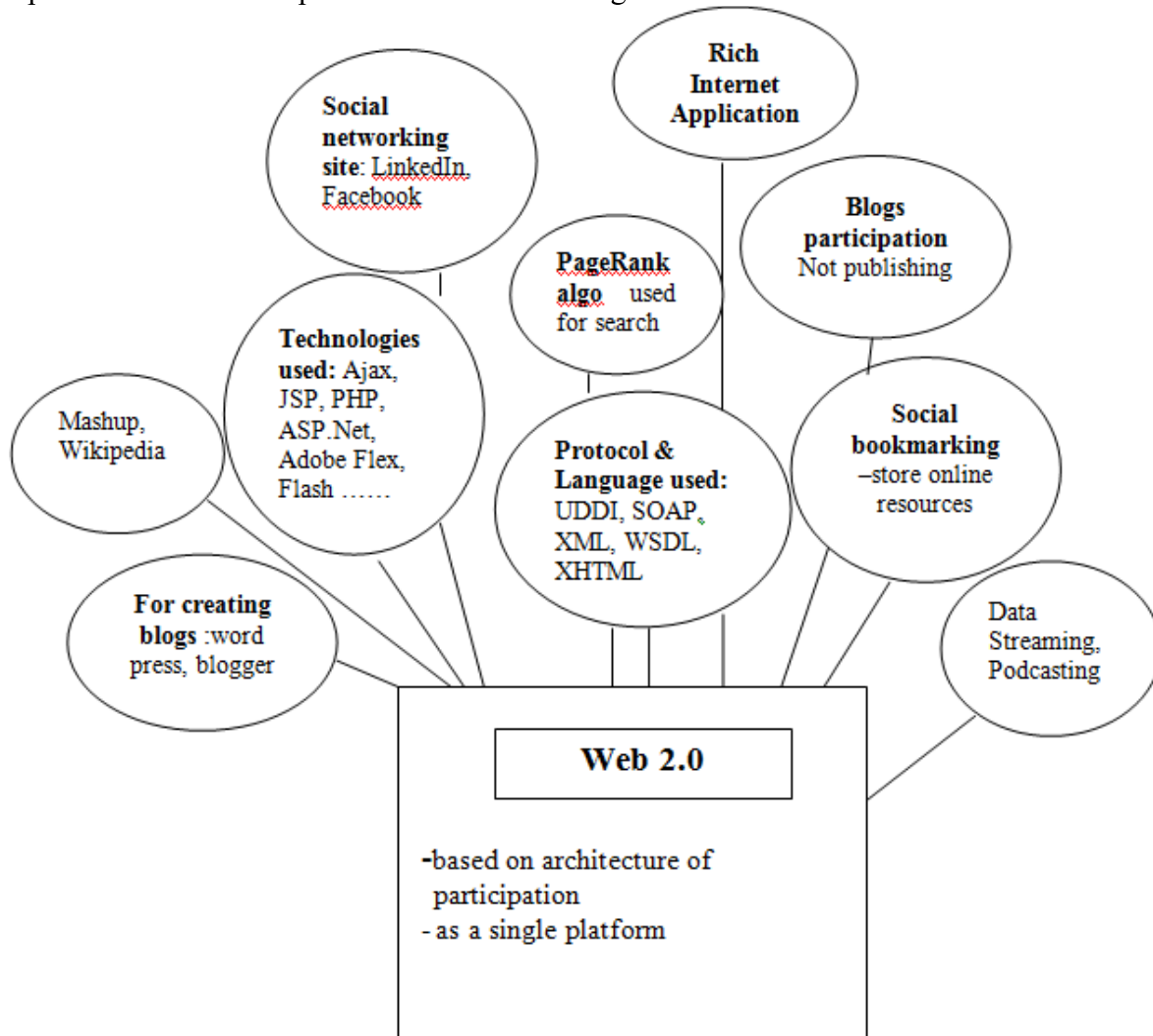


Fig.1: Web 2.0 Tree

1.2.1 Search

The term “Search” syntactically means *to look over carefully in order to find something*. In the context of Web, it defined as *to obtain information on the Internet using search engines*. Some of the popular search engines are Google, Yahoo, MSN, ASK, Bing, Alta Vista, Hot Bot etc. Search engines are the primary tools to find information on the web like text, images, news, videos, blogs and more. When you enter a keyword or phrase, the search engine finds the relevant web pages and shows in the order of ranking of pages. But, how does these search engines work?

In early days, search engines used text based ranking system to decide which pages are more relevant to given query. Modern search engines used different ranking methods to give best result. Google search is based on a priority rank called a PageRank. The PageRank algorithm

considers the web pages that match a given search string and display the sites with the highest PageRank at the top of the search results. Google's PageRank algorithm is based on the premise that a good page is the one that is pointed to by good pages. A detailed discussion on this algorithm is beyond the scope of this Unit.

1.2.2 Content Networks

A vast amount of information resides on the web. Searching a particular term is a complex problem because search engine are listed with number of links related to the search string. The solution of this problem is content networks. Content Networks are website or collection of websites that provide information in the form of articles, wikis, blogs, etc. These sites provide another way of filtering the vast amounts of information on the Internet by allowing users to go a trusted site that has already sorted through many sources to find the best content or has provided its own content. Some names of the content networks sites are eHow, About and HowStuffWorks.

1.2.3 Blogging

A blog or weblog is an informational website maintained by its blogger (or author). It contains discrete entries or post. It is displayed in reverse chronological order (the most recent post appears first). A typical blog contains text, images and links to other blogs or web pages. Most of the blogs are textual. Other blogs are on art (art blogs), music (MP3 blogs), videos (video blogs or 'vlog'), photographs (photoblogs) and audio (podcasts). In education, blogs can be used as instructional resources. These blogs are called as edublogs. WordPress and Blogger are the names of websites that enable anybody to start their own blog. WordPress is one such advanced blogging tool and it provides a rich set of features, through its administrative panels, you can set options for the behaviour and presentations of your blog and easily compose a blog post, push a button and be published on the internet instantly.

Microblogging is another type of blogging but it contains very short post. Microblogging is used in Twitter. The blogs are interactive as they allow readers to leave comments. Bloggers or blog author do not only produce content to post on their blog but also build social relations.

1.2.4 Social Networking

A Social Networking is an online platform that focuses on building the social network or social relation among the users who share their profiles, interests and activities. Friendster, Myspace, Xing, LinkedIn, Twitter, Orkut and Facebook are some popular social networking sites. Facebook is the largest social networking site. LinkedIn is a business oriented social networking site. It allows users to stay in touch with professional contacts. Social networks are also being used by teachers and students as communication tool. Some social networks have additional features such as to create groups or forums and upload photographs or videos. Using social networking site, you can interact by adding friends, commenting on profiles, joining groups and having discussions. Social networking sites allow two way interactions. You can make accounts, publish content and involve in two way communication.

1.25 Social Media

Social Media is a kind of interactive website that does not give any information but interact with you while giving you that information. In social media, user interaction is possible by commenting on photo, editing articles in wiki and voting on articles.

Here are some examples of social media websites:

- **Social Bookmarking:** allow internet users to organize and store to online resources. For example, Delicious, Blinklist. Some of the Bookmarking websites are free and some provide paid services. It allows users to save links to web pages with these bookmarking sites to remember and share.
- **Social News:** You may interact by voting for articles and commenting on them. For example, Digg, Propeller.
- **Social Photo and Video Sharing:** You may interact by sharing photos and videos. For example, YouTube, Flickr.
- **Wikis:** multi-lingual, web-based encyclopaedia Wikipedia. You may interact by adding and editing articles.

1.2.6 Rich Internet Applications (RIA's)

RIAs are a combination of user interface functionality of desktop applications, web applications and interactive multimedia communication techniques (Refer to figure2). Rich Internet Applications are fast, powerful and user friendly. A Rich Internet Applications have same features and functions as the traditional desktop applications. A RIA runs inside a web server and does not require software installation on the client side. RIA split the application processing on different stages as:

- operations related to data manipulation are operated on the server side
- user interaction activity is performed on the client side within a special area of the client desktop called sandbox.

The basic idea of this approach is to decrease the frequency of client – server traffic for handling local activity, calculations and so forth.

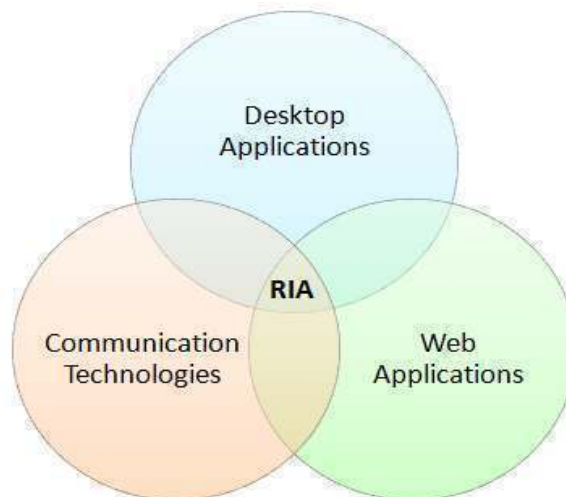


Fig.2: Rich Internet Application

For implementing the Rich Internet Applications, you can use different types of software such as Ajax, JavaScript, Adobe Flex, Flash, Ajax with PHP and many more. Using these technologies, Rich Internet Applications (RIA) solutions provide good interactivity to the users.

1.2.7 Web Services

Web services are a way to communicate between different computers over the World Wide Web. You already know that the Remote Procedure Call (RPC) allow distributed component to communicate. Two RPC technologies, Distributed Component Object Model (DCOM) and Common Object Request Broker Architecture (CORBA) are widely used. But, both the technologies are too complex to provide interoperability among the computers. DCOM is supported by Microsoft and uses the Object Remote Procedure Call (ORPC) to interface components whereas CORBA is designed to be cross-platform and it requires manual intervention. So, both the technologies are not fulfilling the goal of universal interoperability.

In Web 2.0, some of the websites are facing more user interaction such as YouTube and Facebook. Basically these websites are facilitates the user for free interpersonal content sharing, it means that it involves person-to-person interactions through these websites that enables the users for content creation, sharing and manipulation. This is the key feature of the web 2.0. When these interactions are between two or more people and other resources on the Web, then the role of web services are comes. The Web services provide the improvements in terms of interactions or interconnections that may exist between two or more different web resources and hence between those organizations that deliver them. For example, if any organisation wants to include a feature of credit card payments online, it can either set its own setup or can integrate the web service of a Payment Service Provider e.g. Paypal into the website. The user will make their purchase from the Company's website but for payment, the control is automatically transferred to the Service Provider's website. All of this will happen automatically between two organisations.

Web services combines' different protocol to allow all components to interoperate entirely under the control of computer, without human intervention. It means that when a system requires a service, computer can implicitly find that service on the web. For example: In client/server model, a client sends a request to web service (in the server) for weather information. The web service return forecast through service response to the client as shown in the following figure3.

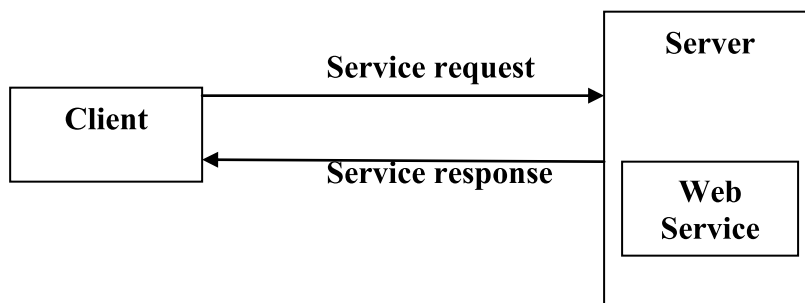


Fig.3: Client/Server Model for weather information

Web service has three roles for providing interaction between different components over web. These are:

- Service requestor,
- Service registry and
- Service provider.

These three roles are combined with find, publish and bind operation as seen in the following figure4. Different protocol and languages are used in web services such as XML, SOAP,

WSDL and UDDI open standards over an Internet protocol backbone. The service provider for providing services, write the services in WSDL (Web Services Definition Languages) and service registry uses UDDI (Universal Description Discovery and Integration Service) to listing what services are available. The service requestor for requesting service uses XML and SOAP standards over Internet Protocol backbone. XML is used to tag the data; SOAP is used to transfer the data. On the web, there are number of web services that use different standards.

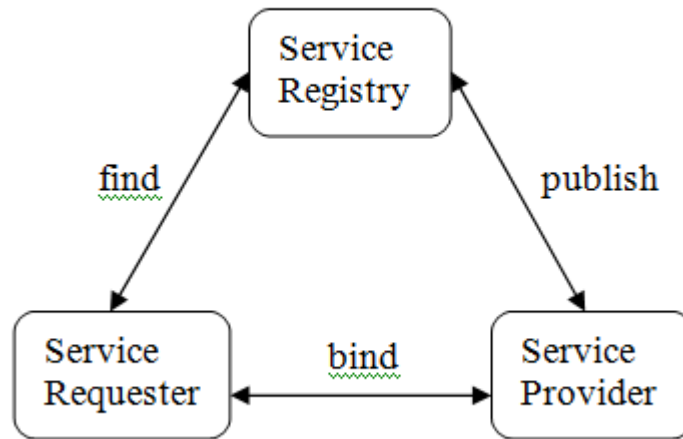


Fig.4: Web Services

1.28 Mashups

Mashups is a web application that retrieved data from two or more external sources to create entirely new and innovative services as shown in figure5. This is a new breed of web based data integration application. There are many types of mashups such as consumer mashups, data mashups and business mashups. Consumer mashups combine visual elements such as maps and data from multiple sources. For example, Wikipediavision is a site that combines Google Map and a Wikipedia API. Business mashups define applications that combine their own data and application with external web services to create single presentation. Data mashups combine data from multiple sources into single source on a similar type of media and information. Mash-ups are often created by using a development approach called Ajax.

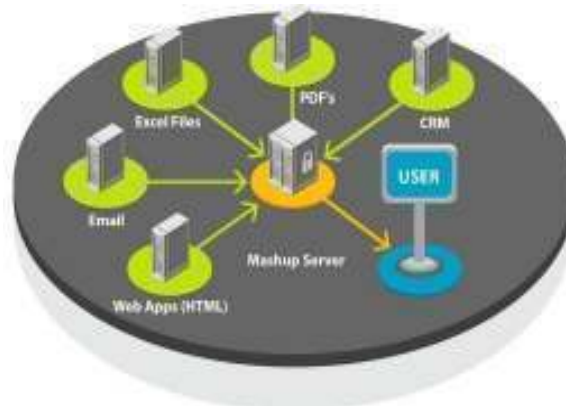


Fig.5: Mashups

1.29 Widgets and Gadgets

Widgets are also referred to as Gadgets. A web widget is a small piece of code that can be placed on a webpage, social media or blog. Widgets are useful application in the form of on-screen device viz., clocks, auction-trickers, event countdowns, flight arrival information and daily weather. A common example of a widget on a website is ad blocks such as Google Ads.

YouTube also provides a widget, allowing you to make play list of your favourite videos. These dynamic widgets can be used in your website to enhance its appeal to the users. Another kind of widget is desktop widget. A **desktop widget** is a small application that resides on your computer's desktop and does not require a web browser to be opened while a web widget is a component of web page, so it does require a web browser. Yahoo widgets are popular source of desktop widgets. Yahoo and Microsoft Vista also provide a widget toolbox to manage desktop widgets.

Windows 8 operating system has number of desktop Gadgets as seen in figure 6.



Figure 6: Windows 8 Desktop Gadgets

1.2.10 Podcasting, Message Board and Data Streaming

Podcasting is a way to allowing users to share their work over the web. A podcast is an audio file that is recorded and delivered over the web. User can listen to it any time. There are thousands of podcasts available on the web ranging from general topics to those which focus on specific topics such as computers, education and music. Teachers and library media experts are using podcasts to deliver their content to students.

A **message board or forum** is a Web 2.0 application that puts students in a web-based place where they can post a comment about any subject, and other students on the same message board or form can respond to the comments or post comments of their own. The message board is a good place for teachers and students for discussing a particular topic

Data Streaming is a technique for transferring data at high speed rate so that it can be processed as a steady and continuous stream. The client side computers do not have fast access to download heavy media files quickly. With data streaming, the client browser can start displaying the data before the entire file has been transmitted.

Check Your Progress 1

1. Explain the term Web 2.0.

2. Define the Search Engine.

3. Explain Social media with examples.

4. What is a Mashup?

5. Differentiate between web widget and desktop widget.

1.3 Introduction to XHTML/HTML5

As discussed earlier, a web page is a document that contains information that is displayed on a browser. These web pages are created using the markup languages XHTML/HTML5. This section describes some of the basic entities used in these markup languages in more details.

XHTML (Extensible Hypertext Markup Language) is a markup language. A markup language is a way of describing the meaning of and relationship between different parts of document. XHTML and its predecessor HTML (Hypertext Markup Language) are the most widely used languages on the web. XHTML defines a set of rules for encoding documents in a format that is both human-readable and machine-readable that reproduces all the features of HTML. XHTML documents can be processed by both Extensible Markup Language (XML) and HTML software. XHTML markup separates presentation details from the information. For this purpose they use style sheets called CSS, which you will learn in Unit 2.

Several features and structural rules of all XHTML document are similar to HTML document (pl. refer to MCSL-016 Block1) and for more details you may refer to the sitew3schools.com.

HTML5 is the new standard for HTML. HTML5 is markup language for structuring and presenting content for web. HTML5 is a platform for mobile, gaming and enterprise applications. Major web browsers such as Internet Explorer, Mozilla Firefox, Apple Safari, Google Chrome, Opera support many of the new HTML5 elements to correctly display web pages. You must use the most recent version of browser to use HTML5.

HTML5 supports most interesting new features like <canvas> element for 2D drawing, <video> and <audio> elements for media playback, scalable vector graphics (SVG) content (that replaces the use of <object> tags) and MathML for mathematical formulas. In addition to this, some new content-specific elements like <header>, <footer>, <section> and <nav> are used to enrich the content of documents. Other new form controls such as calendar, date, time, email and search are to be added to the functionality of HTML5. Before starting the new features of HTML5, let us define some basic tags.

<!DOCTYPE html> Tag: In HTML5, there is only one <!doctype> declaration. This declaration must be first statement in your HTML document that is even before <html> tag. It is not a HTML tag rather it is an instruction to the browser about the version of HTML in which the page is written.

<HTML> Tag: This is root element. It defines the content of web page. The opening HTML tag that is <html> immediately follows the DOCTYPE declaration and closing tag

that is `</html>` is placed at the end of the document. The html element must contain head and body element.

<HEAD> Tag: In head portion of the document, you can write information about the document. You can define title, meta, style, script and link element. You will learn about the style, script and link element in the next units of this block. The head element is written immediately after the opening html tag and end with closing tag appears before the body tag.

<TITLE> Tag: It displays text at the top of browser window.

<META> Tag: The `<meta>` tag is used for declaring metadata (i.e. data about data) for html document. This element is placed between the head element and the basic use of meta element is to provide information about the page to search engine (i.e. Goggle, Yahoo), browsers and other web services. If you want top search engine rankings, just add meta tag in your web pages. You can use multiple meta element with different attributes on same page.

Some examples of usage of meta tags are given in the following three examples:

Example 1: Define keywords for Search engine

```
<meta name="keywords" content="HTML, CSS, XML, JavaScript" />
```

Example 2: Define description of your page

```
<meta name="description" content="Web Page Designing Issues" />
```

Example 3 - Refresh your document every 60 seconds:

```
<meta http-equiv="refresh" content="60">
```

<BODY> Tag: This tag contains the body of the web page that is displayed in the browser display area. In body tag, you can write those tags which are specific to the contents to be displayed in browser.

You can now create a simple web page with minimum but required tags. You can simply use a Notepad as a text editor to create a file of a webpage and save file this with an .html file extension. To see the result of your work in one of the two ways:

- 1) Find the respective location of html file (prg1.html) and double click on it.
- 2) In your browser, click on File/Open File and browse to the name of file.

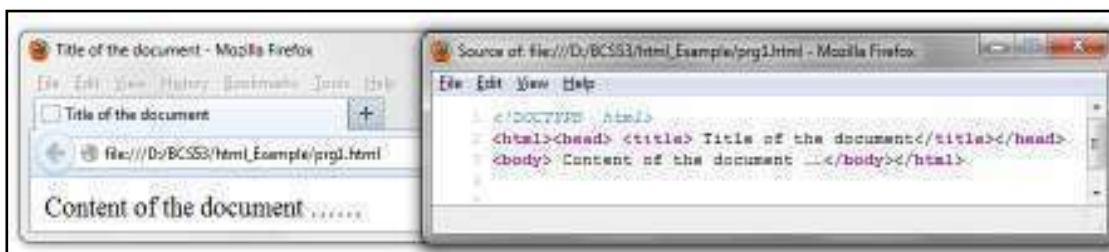


Figure 7: Display of a simple web page and the source code of this web page prg1.html

HEADING Elements: `<h1>`, `<h2>`, `<h3>`, `<h4>` and `<h5>` heading elements are used for different level of heading. `<h1>` is the highest level of heading and `<h6>` is the lowest level of heading. `<h1>` is used once in web page and headings should be used in order. The output and source code of the program are as follows:

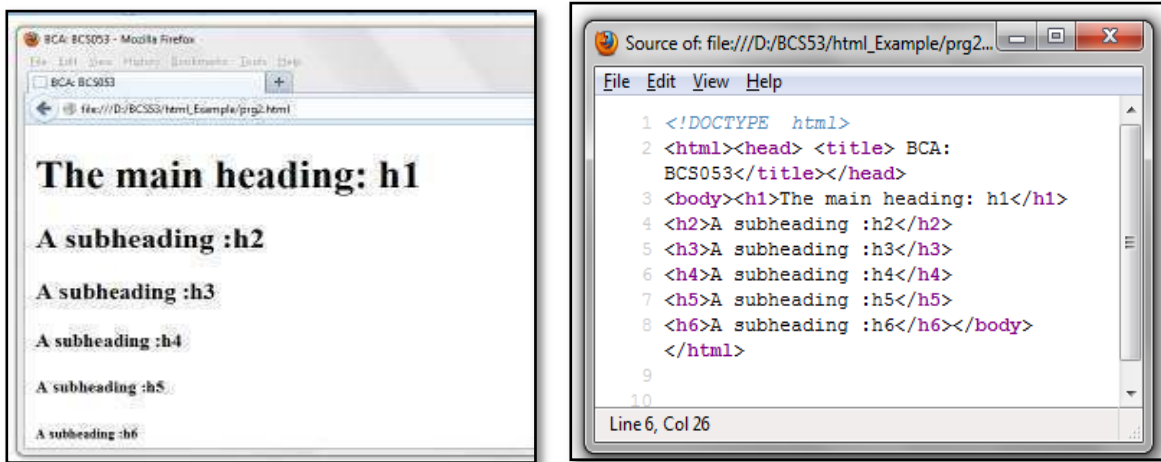


Figure 8: A page with headings and related source code.

<P> Tag: This tag is used for writing paragraph text.

<TABLE> Tag: As you know, table contains row and column. This tag is used to create a table within a web page. In conjunction with table tag, you can use <tr>(table row), <th> (table head cell), <td> (table data cell), <thead>(table header), <tfoot>(table footer) and <tbody>(table body) tags.

<TR> Tag: This tag is used for creating a row. It must appear inside the table element. Two types of cell used in table row. The first cell is <th> and other is <td>.

<Th> Tag: It defines a row or column header. It acts as a table header or data. <td> should be used instead <th>.

<TD> Tag: It creates a column that holds data of table such as text, images, links, lists and other table also. It appears within <tr> element. Table data cell has attributes like rowspan and colspan. The rowspan attribute defines the number of row a cell should span and colspan attributes specifies number of column should span. The syntax for colspan and rowspan are as follows:

<td colspan="number"> and <td rowspan="number">

Attributes appear inside the opening tag and their values sit inside quotation marks.

<THEAD> Tag: This tag is used for table header along with <tfoot> and <tbody> tags.

<TFOOT> Tag: It can be used once within table and must appear before the <tbody> tag.

Consider the following example for table element:

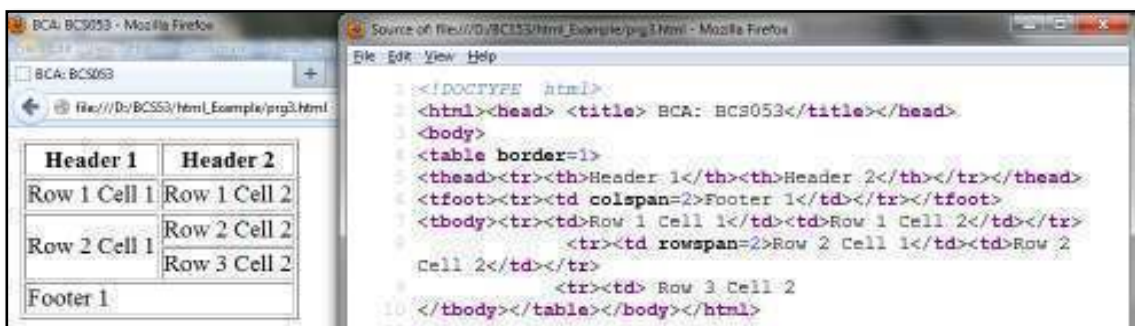


Figure 9: Web page and source code for a table

ANCHOR Elements: If your website has several pages, you need to provide links between HTML pages, the tag is used to create link is called <a> which stands for anchor. The anchor <a> element is used to create a hyperlink which provides a link from one page to another or within the same page. The syntax of this tag is as follows:

```
<a href="url of page" target="_top|_blank|_parent|_self" name="defines an anchor" >
```

In the above syntax, href attribute defines the url of linked resources or web page and target specifies where to open a linked file and name specify the name of intra link which is used within same page.

Some examples of usage of <a> element are given in the following two examples:

Example 1:

```
< a href="head.html"> Header page</a> <!--linking to other pages-->
```

Example 2:

<p> The above example is used to provide links from one page to another page. Now, this example shows you how to use the same element to links within a page. This is used at top the page and at end of the page, you can use following statement which transfer control back to top.</p>

```
<a href="#top">Back</a> <!--links within a page-->
```

Comments:

You can use the following two tags that enable you to add comments within the HTML code. These comments are not displayed by the browser.

Example:

```
<!--This is a HTML comment -->
```

```
<comment> This is also a comment </comment>
```

FORM Elements:

Form elements, as the name suggests, contain predefined field names with blank or default values which you can fill and submit to their respective website. You can create a form for your website using form elements such as input, password, text area, radio button, submit button, select with option, label, fieldset and legend. Forms can be used for filling information online. Some examples of forms include online examination form, feedback forms and so on. Forms are useful tool for client interaction on a website.

<INPUT> Element: The INPUT element is most important element. You can build an entire form using no other element due to its type attributes which have wide variety of values. The syntax is as follows:

```
< INPUT [type=text| password| checkbox| radio| submit| reset| image| button| hidden]
[name=controlName] [value=controlValue] [checked] [size=controlWidth]
[maxlength=word Length] >
```

In the above syntax, name attribute is used to define a name of control element and value attribute specifies value of input control element. The size attribute is used to define width of control element and maxlength attribute indicates a length of input element in characters. The type attribute is defined in the following sections:

The text value: The text value of type attribute enables you to input single line text in a text box. For example, define text box using input element

```
<input type="text" name="username" size=30 >
```

The password value: It is similar to text attribute except that the visible portion of text on screen is masked with other character so that no one can read the password. For example,

```
<input type="password" name="pwdname" size=10 maxlength="8">
```

In the above example, name attribute specify the name of input element and maxlength attribute specify the maximum numbers of characters allowed in input password control. The size attribute indicates width of input element.

In similar way, you can change only value of attribute type of input element for creation of other elements.

<TEXT AREA> Element: This element is similar to INPUT elements text type. Using this tag, user can type a larger section of text in multiple lines rather than a single line text in text boxes. The rows attribute of textarea is used to specify the viewable row and cols attribute is used to specify the viewable columns of the text. For example:

```
<textarea name="username" rows=5 cols=20> </textarea>
```

<LABEL> Element: It is used to add label to form control. It is used with radio button or checkbox option element.

<SELECT> Element: The select form control element is used in conjunction with optgroup and option element to create a list of choices that the user can choose from. It may be drop down menu or list box. The optgroup (option group) is used to define group of elements in a <select> form element.

For this, you can see the following program. Three <select> elements are used in this example. The first <select> statement is used in conjunction with <optgroup> to define group of elements. The label attribute of <optgroup> element specifies the two names such as Maruti Cars and Germans Cars as group header. Each option group element have <option> element to describe the further options in the list such as Wagon R and Swift Dzire in Maruti Cars header. The second <select> element is displayed a simple drop down menu with two option i.e. male and female. The third <select> element is used as a list box.

```

<!DOCTYPE html><html><head>
<title>BCA:BCS053</title>
<h2>Example for select element</h2>
<select> <optgroup label="Maruti Cars">
  <option value="wagon R">Wagon R</option>
  <option value="Swift Dzire">Swift Dzire</option>
</optgroup> <optgroup label="German Cars">
  <option value="mercedes">Mercedes</option>
  <option value="audi">Audi</option>
</optgroup></select><select name="Gender"><option
value="">Male</option><option value="">Female
</option></select><select name="country" size="2">
<option value="">India</option><option
value="">Nepal</option><option
value="">Canada</option></select></body></html>

```



Figure 10: Source code and output screen for select element

<FIELDSET> Element: This element is used for grouping related form elements by drawing a box around the related elements.

<LEGEND> Element: It is used to display a title or caption for group of elements which is grouped by <fieldset> element. It is also used in conjunction with <figure> and <details> element. By using the both elements <fieldset> and <legend>, you can make your web page easier to understand for your user.

```

<!DOCTYPE html><html><head></head><body>
<fieldset>
<legend><b>Personal Information</b></legend>
Name: <input type="text" size="20">
Address<input type="text" size="20">
</fieldset>
</body></html>

```

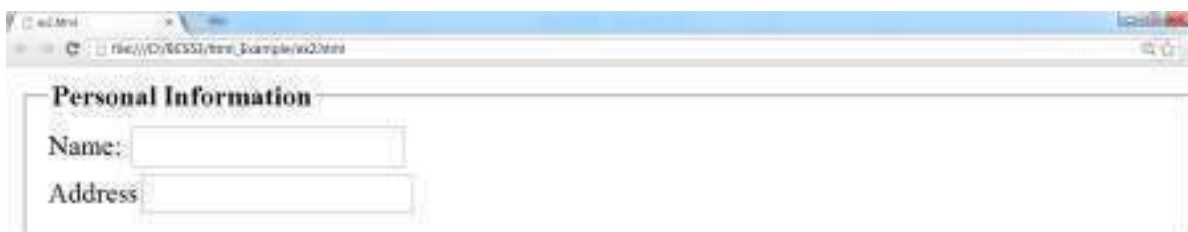


Figure 11: A web page showing Personal information and source code

The above example shows that the legend element creates a caption as 'Personal Information' for fieldset element. The fieldset element is used as grouping the form elements i.e. name and address.

A detailed example of these elements is given in section 1.6. In the remaining section, you can learn about the new features of the HTML5 markup language:

<ARTICLE> Tag: This tag is used to represent an article. It is used in blog for post entry, a newspaper article. The following example shows the result for anchor, article and paragraph elements.

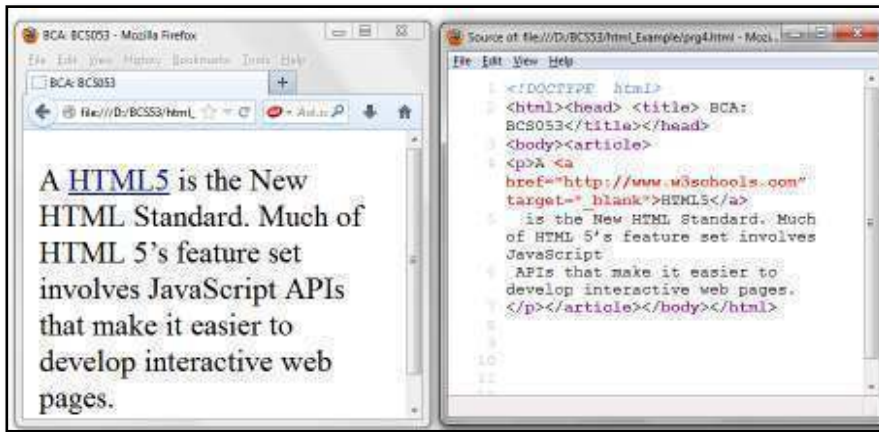


Figure 12: A web page and source code for article and anchor element

<HEADER> Tag: The <header> element is used to define section's heading. The <header> tag cannot be placed within a <footer>, <address> or another <header> element. It is used in a search form or any relevant logos.

<FOOTER> Tag: The HTML <footer> tag is used for defining the footer of an HTML document or section.

<HGROUP> Tag: It is used to group a set of <h1> to <h6> heading elements when a heading has multiple levels.

Example of <article>, <header>, <footer> and <hgroup> tags:

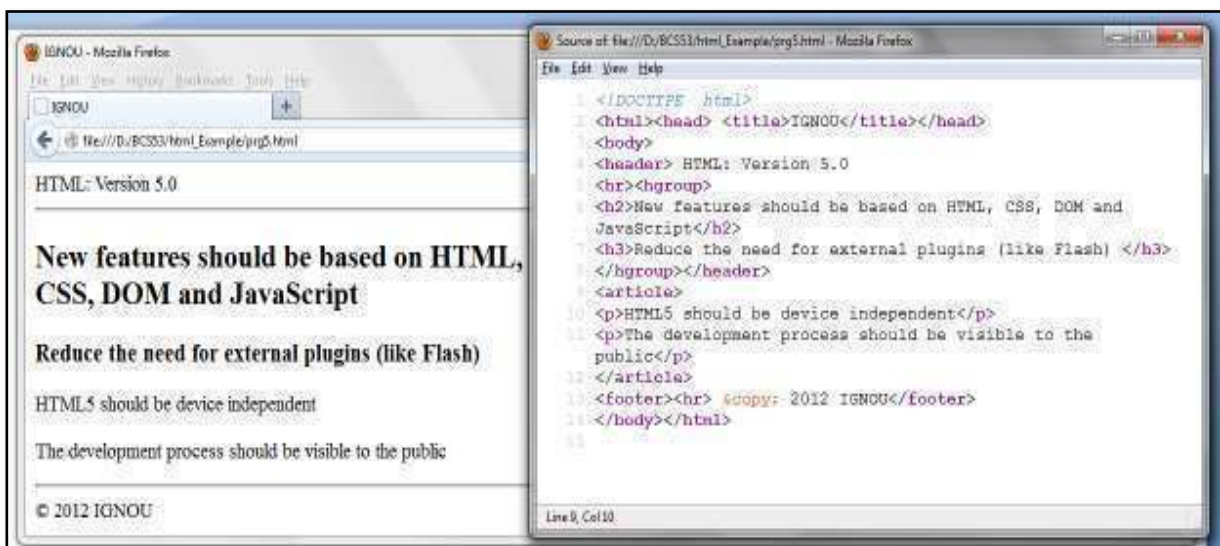


Figure 13: A web page and source code for different HTML5 tags

<AUDIO> Tag: The <audio> tag is used to specify audio on an HTML document.

<VIDEO> Tag: The HTML5 <video> tag is used to specify video on an HTML document.

Both the tag audio and video accepts attributes i.e. control, preload, autoplay, loop, src that specify how the video/audio should be played. The description is as follows:

- controls – it display the respective control on web page
- autoplay- makes the audio/video play automatically.
- loop – play again once it has finished.
- src – name of audio/video source file

This is an example of how you can make use of the HTML5 <video> and <audio> tag in a web page to include a video clip without the need for Flash or Silverlight to be installed in the browser.

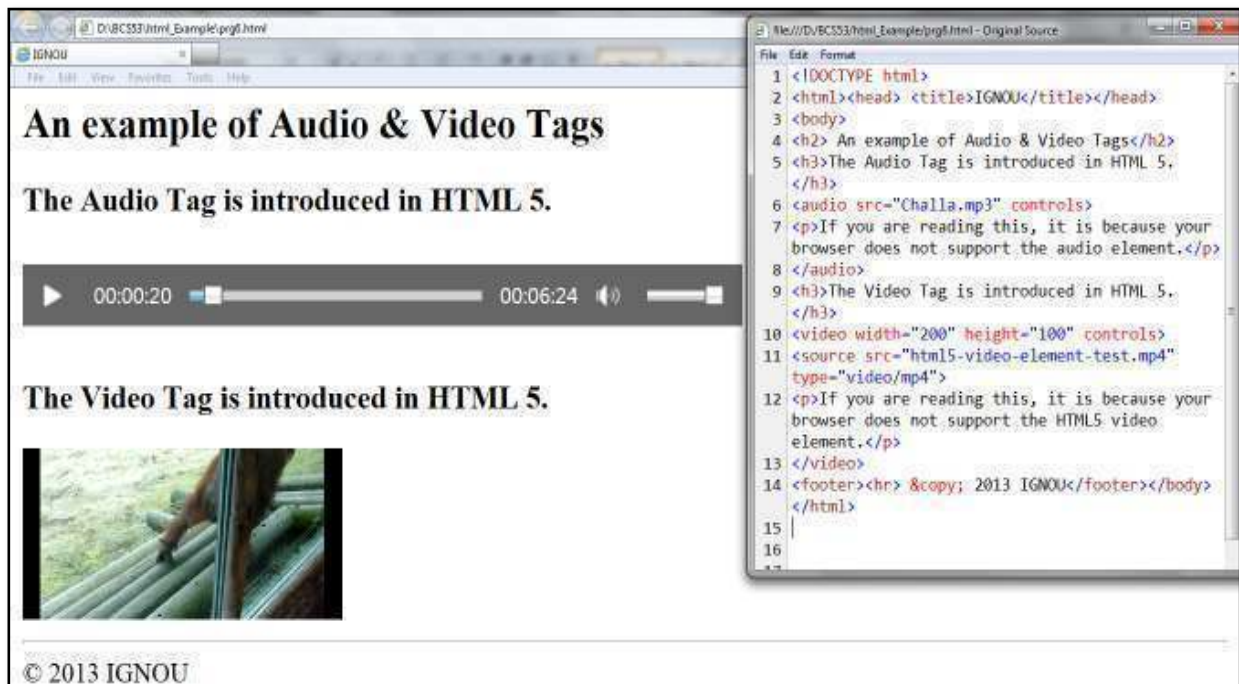


Figure 14: A web page for Audio and video tag

<SVG> Tag: Scalable Vector Graphics is used for describing 2D-graphics application. It is useful for vector type diagrams like Pie charts; Two-dimensional graphs in an X, Y coordinate system etc. For example

```
<!DOCTYPE html>
<head><title>BCA:BCS053-
SVG</title></head>
<body><h3>HTML5 SVG Circle &
Rectangle</h3>
<div><svg >
<circle cx="100" cy="50" r="40"
stroke="green" stroke-width="2"
fill="yellow" /></svg>
</div><div>
<svg >
<rect width="200" height="50" fill="cyan"
stroke="black" stroke-width="3" />
</svg></div></body></html>
```

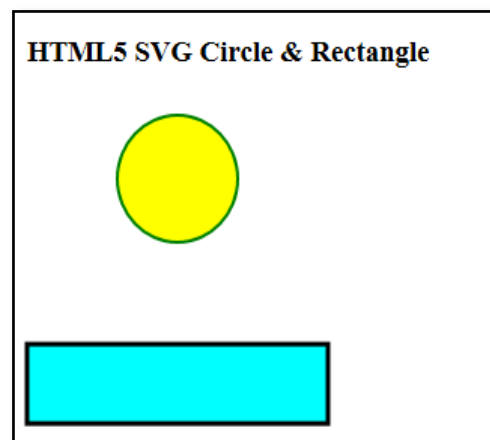


Figure 15: HTML5 program and output screen for Circle & Rectangle

The above program shows circle and rectangle which has drawn by <svg> element. For the creation of circle, a keyword 'circle' and x, y coordinate is defined. The <circle> element is written inside the <svg> element. In this example, cx attribute of circle specifies x coordinate, cy is y coordinate of the circle and r represents radius of the circle. The attribute stroke is used for fill the color and attribute stroke-width is used for defining the width size of outermost area of circle. The color of inner circle is filling using the fill attribute. In the same way for creation of rectangle, a keyword 'rect' and width, height is defined. The <rect> element is written inside the <svg> element.

<CANVAS> Tag: This element is used to for creating graphics on the fly via scripting (usually JavaScript). Canvas has several methods for drawing boxes, characters, path and adding images. Canvas is a rectangular area on which you can draw object using javaScript.

<TIME> Tag: This tag is used for declaring the date/time within an HTML document.

<MENU> Tag: The HTML menu tag is used for specifying a list of commands.

<MARK> Tag: This tag is used for indicating text as marked or highlighted for further reference.

Consider the following code for <mark>, <time> and <menu> Tag.

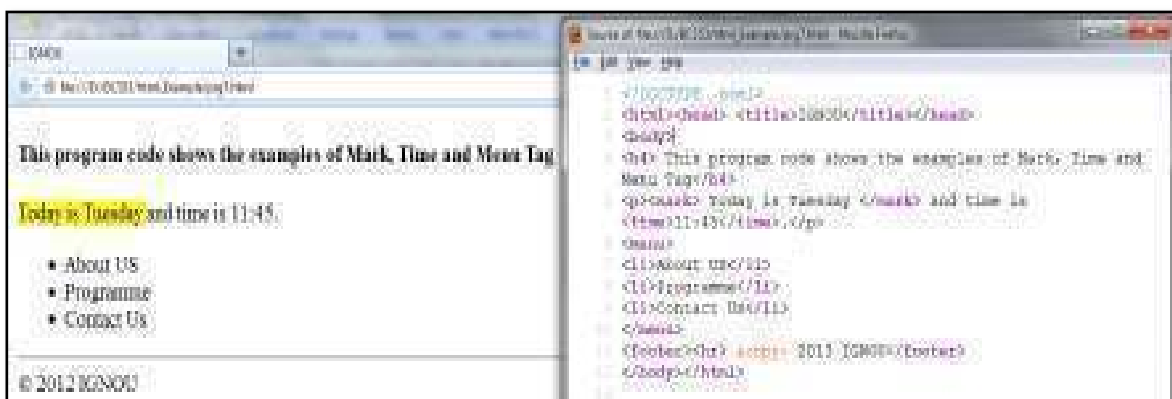


Figure 16: A HTML page with source code

<EMBED> Tag: If you want to play a video in a web page, you can upload the video to YouTube in proper HTML code to display the video.

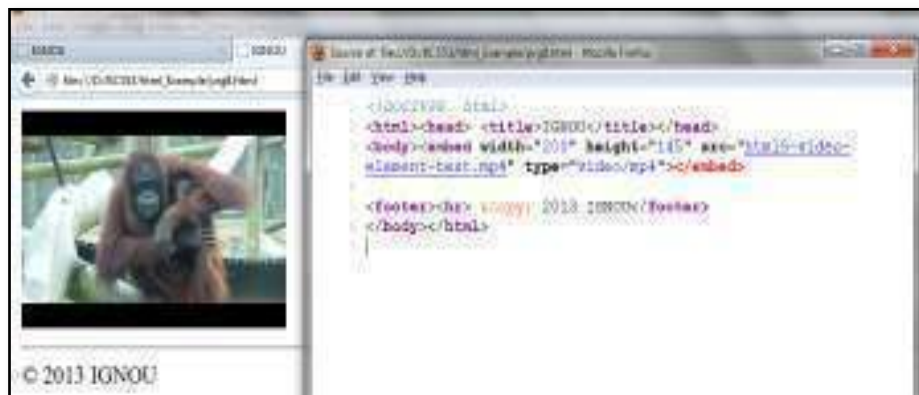


Figure 17: A web page source code and screen for embed tag

You can embed a video file in the following format:

```
<embed width="300" height="150" src="videofile.mp4" type="video/mp4"></embed>
```

In the above code, src attribute define the url of the video file and using width, height attribute, you can defined the width and height of the video clip

<DETAILS> Tag: This tag specifies additional details that the user can view or hide on demand. It can be used in conjunction with **<SUMMARY>** tag to provide a heading that can be clicked on to collapse/expand the details as required. You could use `<details>` to toggle the comments section of a blog, member profiles, details of a download, complex forms, or in web applications. **Only browser Chrome supports the `<details>` and `<summary>` element.**

For Example:

```
<!DOCTYPE html><html>
<head><title>BCA:BCS053</title></head><body>
<details> <summary><label for="name">Name:</label></summary>
  <input type="text" id="name" name="name" />
</details> </body></html>
```

In this program, detail element create a collapse/expand toggle mark on the page, whenever you clicked on this mark, it show/hide the text.

<ASIDE> Tag : This tag is used for defining the content aside from the page content. Basically this tag is used to represent quotation text that is related to page.

<MATHML> Tag: The MathML standard is used for mathematical notations on web page.

<BDI> Tag: This tag can be useful when the text direction is unknown such as in the case with user generated content.

<NAV> Tag : The `<nav>` tag defines a section of navigation links.

<FIGURE> Tag: This tag is used for annotating illustrations, photos and diagrams. The `<figure>` tag is used in conjunction with **<FIGCAPTION>** tag to provide a caption to the figure.

Consider the following code for `<aside>`, `<bdi>`, `<figure>`, `<figcaption>` and `<nav>` tags.

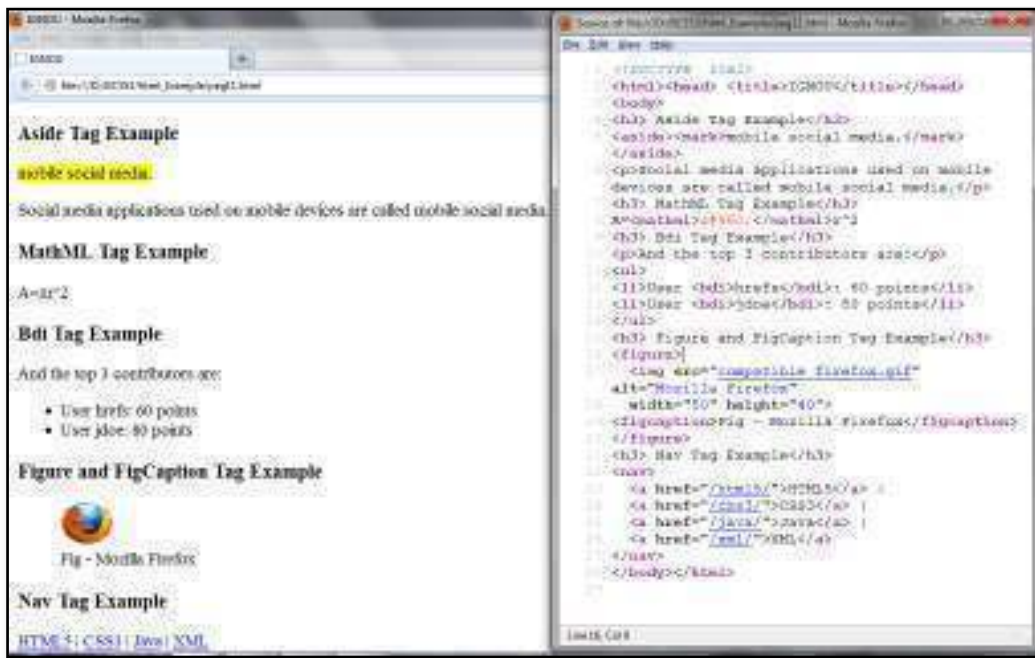


Figure 18: A web page display different tags of HTML 5

Check Your Progress 2

1. What is the use of `<!DOCTYPE html>` tag in HTML5?

2. Explain the SVG element in HTML5 with Example?

3. Write a HTML5 program using `<details>` and `<summary>` tag.

4. Differentiate between SVG and Canvas tag?

5. What is the purpose of aside tag in html5?

1.4 Syntactic Differences between HTML and XHTML

There are some significant differences between the syntactic rules of HTML and XHTML. These are:

1. Case sensitivity: In HTML, it does not matter whether tags are lowercase, uppercase or a mixture of both. In XHTML, all tags must be lowercase. For example, use `<head>` instead of `<HEAD>`.
2. Closing tags: Closing tags are often optional in HTML but are always required in XHTML. For example, every paragraph must begin with `<p>` tag and end with a `</p>` tag. Likewise, every `<tr>` table row must have a closing `</tr>` and so forth.
3. HTML tags that do not enclose any content such as `<hr>`, `
` and `` must contain a slash. For example, `<hr />`, `
`, ``. These statements are treated by XHTML interpreter as a closing tag.
4. Quoted attribute values: In XHTML, all attribute values must be enclosed in quotation marks regardless of what characters are included in the value. For example

`` is valid HTML but it must be written as `` to be valid in XHTML.

5. Explicit attribute value: All attributes must have values in XHTML. For example,
`<input type = "checkbox" checked>` should be written as `<input type="checkbox" checked="checked" />`
6. id attribute : The id attribute is used to identify objects. For example,

``

In XHTML, the use of id is encouraged and the use of name is deprecated.

7. Element nesting: Tags must be properly nested in XHTML. For example,

` _{This is bolded subscript }` should be written as ` _{This is bolded subscript}`.

8. Extra whitespace that appears in attribute values is stripped out. For example, If the following code is encountered

``

The XHTML processor will interpret it as ``.

1.5 Standard XHTML /HTML5 Document Structure

The previous section discussed rules for writing XHTML. Documents that satisfy these simple rules are said to be well-formed XML but they do not reference a DTD that provides specific rules for allowed names and usage of elements and attributes.

Three features in figure 19.0 are characteristic of XHTML documents. These are:

1. Every XHTML document must begin with an XML declaration. All XML documents should begin with the following line: `<?xml version ="1.0" encoding = "utf-8" ?>`

This declaration states that the rest of the document is XML. The version attribute specify the version number, which is still 1.0. The encoding property defines the encoding in which the document is stored on a computer.

In HTML5, every document must begin with `<!DOCTYPE HTML>`

2. XHTML documents require a DOCTYPE declaration just after the XML declaration. A DOCTYPE declaration is the XML mechanism for specifying the document type declaration (DTD) relevant to the document.

```
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
```

<http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>> this declaration defines the location of the DTD specified via a URL.

<pre><?xml version="1.0" encoding="utf-8" ?> <!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN" http://www.w3.org/TR/xhtml11/DTD/ xhtml11.dtd> <html xmlns=http://www.w3.org/1999/xhtml> <head> </head> <body> </body> </html></pre>	<pre><!DOCTYPE HTML> <html> <head> <title>Document Name</title> <link href="name_of_cssFile " rel="stylesheet" /> <script src="name of js file"></script> </head> <body> <p>Your content</p> </body> </html></pre>
---	---

Figure 19.0: Standard XHTML and HTML5 Document Structure

3. XHTML document must include the four main tags <html>, <head>, <title> and <body>. The <html> tag identifies the root element of the document and this element must use the xmlns attribute to declare the location of the namespace for XHTML syntax. The only allowed form is :<html xmlns=http://www.w3.org/1999/xhtml>

In the same way, HTML5 document must contain < html>, <head>, <title> and <body>. Its also include <script> and <link> tag. The link tag is used to include cascading style sheet and script tag is used for java script file.

1.6 Example of HTML5 Form

The following program creates an Application Form which includes fields such as name, gender, address, qualification and some other information.

In the example, no. of elements are used such as title, meta, table, select, fieldset, legend, optgroup, option, input, textarea and anchor. The title and meta elements are defined in the head portion of the HTML page and rest of the tags are defined in the body tag. The title tag tells to users and search engine what the topic of a particular page. The meta tag is used to provide information about the page to search engine. The method, name and action attributes are defined within the form element. The name attribute is used to define a name of the form. It is used to reference form data after the form is submitted.

```

<!DOCTYPE html>
<html><head> <title>IGNOU</title>
<meta name="keywords" content="HTML5, IGNOU, application">
</head><body ><h3><a name="top"> Application Form</a></h3>
<form name="HTMLForm" method="get" action="">
<table border="0">
<tr><td>Name of Applicant:<input type="text" name="FName" size="20"
maxlength="50"> </td>
<td>Gender:
<select name="Gender">
<option value="">Male</option><option value="">Female</option>
</select>
</td></tr><tr><td >
<fieldset ><legend>Address Details</legend>
Current Address: <input type="text" name="caddress" size="40" maxlength="50" >
<br />
Parmanent Address :<input type="text" name="Paddress" size="40" maxlength="50">
</fieldset>
</td><td></td></tr><tr><td>City :

<select>
<optgroup label="Assam">
<option value="">Dibrugarh</option>
<option value="">Jorhat</option></optgroup>
<optgroup label="Punjab"><option value="">Amritsar</option>
<option value="">Jalandhar</option></optgroup>
</select>

</td><td>Country:
<select name="country" size="2">
<option value="">India</option><option value="">Nepal</option>
<option value="">Canada</option></select>

</td></tr>
<tr><td colspan="2"><table border="1" cellpadding="0" cellspacing="0"
width="40%"><tr><td >Examination</td><td >University/Board</td>
<td >Year of Passing</td> <td >Division/Marks</td></tr>
<tr><td >10th</td><td ><input type="text" name="10_univ" size="20"
maxlength="50"></td><td ><input type="text" name="10_year" size="20"
maxlength="50"></td><td ><input type="text" name="10_div" size="20"
maxlength="50"></td></tr><tr><td>12th</td><td ><input type="text"
name="12_univ" size="20" maxlength="50"></td><td ><input type="text"
name="12_year" size="20" maxlength="50"></td><td ><input type="text"
name="12_div" size="20" maxlength="50"></td></tr></table></td></tr><tr>
<td>Other Information<textarea rows="3" cols="40">Enter text
here.....</textarea></td></tr><tr><td colspan="2"><input type="submit"
value="Submit" ></td></tr></table></form>
<footer><hr>&copy; 2013 IGNOU </footer>
<p align="right"><a href="#top" >Return to Top</a></p></body></html>

```


The action attribute is used to send form-data after the form submission. The get method of form element is used to get requests data from the specified resource which is defined in action attribute. The table tag is used in conjunction with tr, td tag for creating row and column. The border attribute is used to specify border of the table. The value “0” indicates the table is displayed without border. The td element is used to create a table cell. The table cell can span across more than one column or row by using the rowspan and colspan attributes. The select element is used with option group/option element to create drop down or list box. When the value of attribute type of input element is text, it create a text box for users input and when the value is submit, it creates a submit button for submission of page. The textarea element is used for creation of multiple lines of text box for entering some other information which are not included in the form. The rows and cols attributes of textarea is used to create a larger section for input data.

Output of the above program is as follows:

The screenshot shows a web browser window titled 'IGNOU - Mozilla Firefox'. The address bar shows the file path 'file:///D:/BCS53/html_Example/complete.html#top'. The main content area displays an 'Application Form' with the following elements:

- Name of Applicant:** A text input field.
- Gender:** A dropdown menu with 'Male' selected.
- Address Details:** A container for 'Current Address' and 'Parmanent Address' (sic) text input fields.
- City:** A dropdown menu with 'Dibrugarh' selected.
- Country:** A dropdown menu with 'Nepal' selected.
- Exam Table:** A table with 4 columns: Exam, City/Board, Year of Passing, and Division/Marks. It contains rows for 10th and 12th exams under Assam and Punjab boards.
- Other Information:** A large text area for additional details.
- Submit:** A button to submit the form.
- Footer:** Copyright notice '© 2013 IGNOU' and a 'Return to Top' link.

Exam	City/Board	Year of Passing	Division/Marks
10th	Jorhat		
12th	Amritsar		
	Jalandhar		

Check Your Progress 3

1. What do you mean by HTML? Differentiate between HTML and XHTML.

2. Write program for header section of the page which include name of IGNOU as heading and menu bar as Unit1, Unit2, Unit3.

3. Write an html5 program for highlighted text “I am student of IGNOU”.

4. Create footer for web page with copyright symbol.

5. Write a complete web page including header and footer.

1.7 Summary

Web 2.0 is the popular term for advanced internet technology and applications. It allow to users to actually interact with each other in the form of blogs, social networking site, social media, wikis and many more. An important part of web 2.0 is the social media, which is the way to enable people to create social networks online. This network provides online tools for sending individual messages, photo sharing and online chat. The most famous social networking site is Facebook. Related to social media is social bookmarking site. Wikis are website that allows users to add and modify content. Wiki is the multi-lingual, web-based encyclopaedia Wikipedia.

You have learned more elements and attributes about HTML5. Now, you can able to create a new webpage using this technology.

1.8 Solutions/Answers

Check Your Progress 1

Ans1: Web 2.0 is online communication platforms that facilitate online sharing and interaction. Tools like blogs, podcasts and message boards are common Web 2.0 platforms. For more detail, pl. refers to section 1.2.

Ans2: Search engines are programs that search web page for given keywords and returns a list of the web pages where the keywords were found. It enables users to search for

web pages on the Internet. The search engines are Google, Yahoo, Bing and many more.

Ans3: Social media is type of online media where it provides online conversations between users. It also delivers content but does not allow users to participate in creation or development of the content. It provides a platform where users are interact by sharing photos and videos and commenting on them. For example, YouTube, Flickr

Ans4: Mashups is a web application that retrieved data from two or more external sources to create entirely new and innovative services. This is a new breed of web based data integration application

Ans5: Web widget is a component of web and it require web browser. On the other side, desktop widget is a small application that resides on computer desktop and does not require web browser.

Check Your Progress 2

Ans1: Every HTML5 program must begins with `<!doctype>` declaration. It is an instruction to browser about what version of HTML page is written in.

Ans2: SVG is used to create scalable vector graphics. SVG is container element that can have other element to describe the shape within itself. for example, you can see respective section.

Ans3: `<!DOCTYPE html><html><head><title>BCA:BCS053</title></head><body><details open="open"><summary>Name</summary><p>Dr. A.P.J. Kalam</p><p>If your browser supports this element, it should allow you to expand and collapse these details.</p></details>`

Ans4: Canvas is used to create a raster graphics whereas SVG is used to create vector graphics. As you know that the raster graphics is loses quality image when enlarged and vector graphics retains it. canvas is rectangular area on which you can create something using JavaScript, on the other hand, SVG is container element that can have other element to describe the shape within itself. SVG can have css control over it.

Ans5: The aside tag is used to represent text aside from main article. It is appears in bold typeface in box or as a quotation text on the page.

Check Your Progress 3

Ans1: HyperText Markup Language (HTML) is a language. It consists of a variety of elements called tags, which is used for everything from defining a title to including audio/video clip, from creating a heading to inserting a background image in a web page. For other part of answer, pl. refer to section 1.4

Ans2: `<!DOCTYPE html><html><head><title>BCA:BCS053</title></head><header><hgroup><h1>Indira Gandhi National Open University</h2><h2>SOCIS </h2><nav>Unit 1| Unit 2| Unit 3</nav>`

```
</hgroup></header></body></html>
```

Ans3:

```
<!DOCTYPE html><html><head><title>BCA:BCS053</title></head>
<body><p><mark>I am student of IGNOU</mark></p></body></html>
```

Ans4:

```
<!DOCTYPE html><html><head><title>BCA:BCS053</title></head>
<body><footer><hr> &copy; 2013 IGNOU</footer></body></html>
```

Ans5: pl. refers to above program answer 2, 3 and 4.

1.9 Further Readings

- 1) Beginning Web Programming with HTML, XHTML and CSS by Jon Duckett. Publisher: Wrox.
- 2) Internet and the World Wide Web by Harvey M. Deitel, Paul J. Deitel
- 3) Programming the World Wide Web by Robert W. Sebesta. Publisher: Pearson
- 4) HTML 4 Unleashed by Rick Darnell. Publisher: Techmedia
- 5) XHTML 1.0 by S. Graham. Publisher: Wiley
- 6) <http://www.w3schools.com>

UNIT 2 STYLE SHEETS

Structure

- 2.0 Introduction
- 2.1 Objective
- 2.2 Cascading Style Sheet
- 2.3 Style Sheet Types
 - 2.3.1 Inline Styles
 - 2.3.2 Embedded Style Sheets
 - 2.3.3 Linking External Style Sheets
- 2.4 Some Basic Properties in CSS
 - 2.4.1 Font Properties
 - 2.4.2 List Properties
 - 2.4.3 Color Property Value Forms
 - 2.4.4 Alignment of Text
- 2.5 Selector Forms
 - 2.5.1 Simple Selector
 - 2.5.2 Class Selector
 - 2.5.3 id Selector
- 2.6 The Box Model
- 2.7 Background Image
- 2.8 The <div> and Tags
- 2.9 CSS Complete Example
- 2.10 Summary
- 2.11 Solutions/Answers
- 2.12 Further Readings

2.0 Introduction

In the previous Unit, you were introduced to some of the tags of HTML5. In each tag of HTML, you can define a number of attributes that may define the display characteristics, such as font, colour, etc. of the content in the tag. However, this model of putting the display characteristics inside the tags makes a web site inflexible, inconsistent and difficult to change. There was a need of separating content of a web page from its presentation that is display in browser. This was made possible by Cascading Style Sheets (CSS). CSS allows separation of content of web pages from their presentation. The content can be part of an HTML page, whereas the presentation can be moved to CSS. In fact, a CSS can be used to control the style and layout of multiple html pages all at once. Cascading Style Sheet is a mechanism for adding style such as background color, font color to web pages or documents. The W3C, an organization that oversees the development of web technologies, has released three versions of CSS, namely CSS1 in 1996, CSS2 in 1998 and the latest version CSS3.

This unit introduces you to CSS and their types. It also explains how style sheet can be created and used to control the display of web pages on a browser. An external style sheet is a file that contains only CSS code and it will allow you to change the formatting style in multiple pages at once. This unit provides a complete example to demonstrate this concept. You must use CSS while designing web pages as they allow the web pages - to be portable across browsers, - to have a uniform look and feel, to be have simple tableless design and to

easier to maintain. Please note that a complete discussion on CSS is beyond the scope of this Unit, you may refer to the site <http://www.w3schools.com> for more details on CSS.

2.1 Objectives

At the end of this Unit, you will be able to:

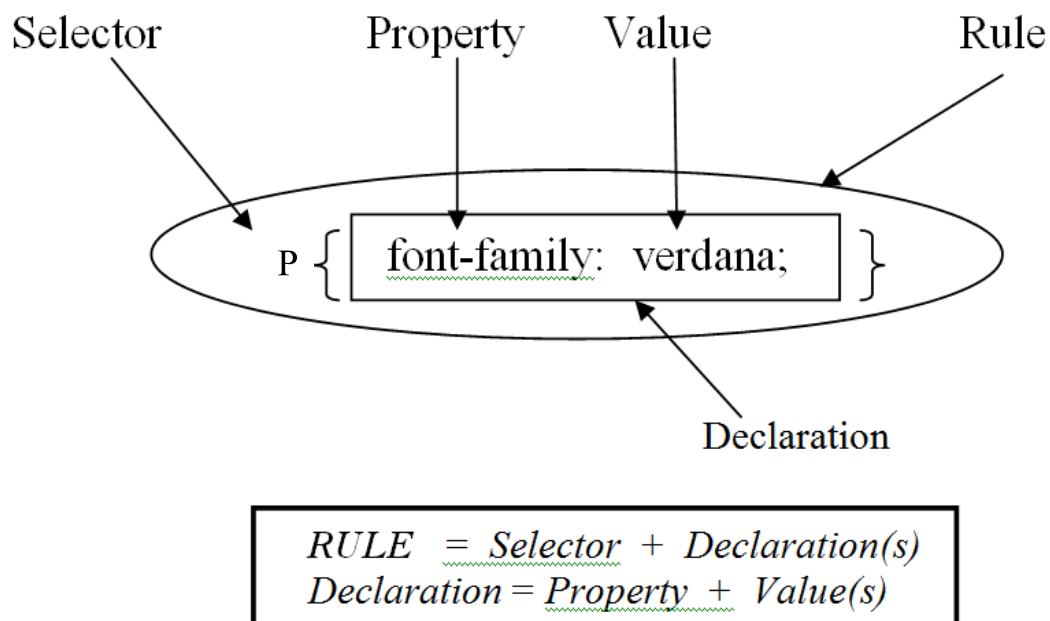
- write CSS rule for different elements of a web page
- link an external CSS to a web page
- control the presentation of text using CSS
- define a box model and how you set different properties for this model
- show how simply changing the CSS can alter the display of a web page

2.2 Cascading Style Sheet

Cascading Style Sheets (CSS) are used to control the presentation of a single or many web pages. In the earlier web pages, the display attributes were put in the particular tags. One of the major problems of such a situation was that if you have to insert same style to number of occurrences of that tag in same or different pages, you have to enter that style in every tag. Not only this was tedious and time consuming but also are difficult to change. But, if you use Cascading style sheet, you can set same formatting style to all such tags or elements using a single rule. This rule can then be applied to all the occurrences of that element.

A style sheet consists of *rules* that specify how formatting should be applied to particular elements in a document. In general, a style sheet contains many such rules. Each CSS rule contains two parts. The first part is *selector* and the other part is *declaration*. The selector indicates which element or elements, the declaration applies. In the selector part the elements can be separated by comas. The declaration specifies formatting properties for the element.

The following example shows you that how to use CSS rule within your document:



In above example, the rule says that all the paragraphs, that is, <p> elements should appear in Verdana font. Notice that the declaration part is split into two parts namely property and value separated by colon. The declaration part is always written inside the curly brackets ({ }). Semicolon marks the end of a declaration.

A property defines the property of the selected element(s), in this example the font-family property, that is, different display fonts. A value which specifies the value of this property, in this example, it is Verdana fontface.

The example of CSS rule given below applies to several elements such as <h1>, <h2> and <h3> elements. Note that in the example, selector is h1, h2, h3 (separated by comma). The declaration of the rule specifies several properties for these elements with each property-value pair separated by a semicolon and all properties are kept inside the curly brackets.

```
h1, h2, h3 { font-family: verdana, tahoma, arial; font-weight: bold; font-style: italic; }
```

In the above example, the font-family property specifies the font for a heading element. The font-family property can hold several fonts. This rule specifies that the content of each heading element <h1>, <h2> and <h3> of the related web page will be displayed in bold, italic and Verdana font. If the Verdana font is not installed on the computer then it will look for Tahoma and even if Tahoma is not installed then Arial font will be used.

CSS can be attached to a web page in three different ways. The next section is devoted on this issue. More details on CSS properties will be provided in the latter sections.

2.3 Style Sheet Types

In previous section, we have discussed about the CSS rule. These rules are differently written in different style sheets. In this section, you will learn that how to place these CSS rules in your documents.

There are three different types of style sheets namely Inline Style, Embedded Style and Linked External Style Sheets. The inline style sheet is used to control a single element; embedded style sheet is used to control a single page, while the linked external style sheet is used to control the entire site. This is discussed in more details in the following sub-sections.

2.3.1 Inline Style

Inline style sheet provides the finest level of control. This style sheet is written only for the single element. These style specifications appear within the opening tag of the element.

The style specification format for Inline style sheet are as follows:

Style = “property_1: value_1; property_2: value_2;;property_n: value_n;”

The above syntax appears as a value of the style attribute of any opening tag. **For example:**


```

<!DOCTYPE html>
<html><head></head><body>
<table border="2"><tr><td>Normal TD Cell</td>
<td style="font-family: arial; border-style: solid; background-color: red">
IGNOU:BCS053</td>
</tr></body></html>

```

In the above code, a table data cell will display data in Arial font with background color red and solid border. The output is as follows:



The following Inline Style Sheet sets the background colour of a paragraph. Please note that text colour and paragraph elements background colour have been specified using hexadecimal numbers.

```

<!DOCTYPE html>
<html>
<head></head>
<body>
<p style="text-transform: capitalize; background-color:#00FFFF; color:#3300CC;">
css examples for inline style sheet</p>
</body></html>

```

Output:

Css Examples For Inline Style Sheet

Using Inline definition, paragraph's data is converted from lowercase to sentence case form style with cyan background and blue text colour.

2.3.2 Embedded Style Sheets

This style is placed within the document. For this style sheet, CSS rule can appear inside the <head> element or between the <body> tag contained with a <style> element. The format for this embedded style is as follows:

```

<style type="text/css"> ..... rule list ..... </style>

```

The *type* attribute of the <style> element tell the browser the type of style specification which is always *text/css* and rule is written in the following form:

```

Selector {property_1:value_1; property_2:value_2;.....; property_n:value_n;}

```

For example:

```

<html><head>
<style type="text/css">
body { background-color: #FF0000; color: #000000;}
h2, h3 {font-family: arial, sans-serif; color: #ffffff;}
p { color: #3300FF; font-size:20px;}
</style>
</head><body>
<header> HTML: Version 5.0
<hr><hgroup>
<h2>New features should be based on HTML, CSS and JavaScript</h2>
<h3>Reduce the need for external plugins (like Flash) </h3>
</hgroup></header>
<article>
<p>HTML5 should be device independent</p>
<p>The development process should be visible to the public</p>
</article>
<footer><hr> &copy; 2013 IGNOU</footer>
</body></html>

```

Output of the above program:



The above program shows that the how to override the color of one element to other elements. The color property is defined for body, heading and paragraph elements. The heading is defined within the section heading element named <header> in white color, content of paragraph element is displayed in blue color and footer is in black color. The color of heading and paragraph element overrides the body color. If it is not defined than whole body text is displayed in black color. The <h2> and <h3> are defined within the <hgroup> tag. The <hgroup> tag is used to grouping heading elements when the heading has multiple levels. The paragraph element is defined within <article> tag. The article tag is used to write text in article form like newspaper and blog. The <hr> element is used to write a horizontal row or line as a separator. The copyright symbol is defined using ampersand sign ('&') and copy.

2.3.3 Linking External Style Sheets

As discussed earlier the Inline style apply within the tag and embedded style apply at document level. In those cases, when style specifications is required in more than one document. For this purpose, Linked External Style Sheet is used. Linked External style exists as a separate file that is linked to a page with the <link> element inside the <head> tag. This

file has .css extension and is referenced with URL. Inside the .css file, all style specifications are written which applicable for the document(s). For linking an external style sheet in a web page, the following command is included:

```
<link rel="stylesheet" href="name_of_file.css" type="text/css" >
```

The *<link>* element is used to create a link to css style sheets. The *rel* attribute specifies the relationship between the document containing link and the document being linked to. The *href* attribute specify the URL of the css file.

How to Create an External Style Sheet

External style sheets are created with a similar syntax as document level style sheet.

```
Selector {property_1:value_1; property_2:value_2;.....; property_n:value_n;}
```

For example:

```
body { color: #333333;
      background-color: #FFFFFF; // it indicate white background
      background-image: url(innovation_files/innovation_bg.gif);
    }
h1 { font-size: 17px;
     color: #ff0000; // red color
     font-family: "Times New Roman", Times, serif;
     font-weight: bold; }
```

Save these rules into a text file with the extension .css. In this way, you can write many more elements in .css file. Now, you need to link it to your Web pages.

In example given above, there are defined style sheet rule for only two elements i.e. body and h1. You can define in .css file as many rule as per your requirement. When you are included this external sheet in your web page, the heading text is displayed in red bold color and times new roman font with size in 17 pixels. The background-color property is used to display the body color of the web page, in this example, it is white color. The background-image property is used to display an image in background of the body of the page. For this, you can define a path with name of image as a URL of the background-image property.

Precedence of styles

There are several rules that apply to determine the precedence of style sheets. The level of priority depends on the style definition which is closer to the tag. For this order, linked external style sheets have lower priority than embedded style sheets, which are of lower priority than inline style sheets. If the browser did not find any style specification then it uses default property values of its own.

For example:

CSS code:ignou.css

```
p{ color:blue;}
```

Source code:

```
<!DOCTYPE html>
<html><head>
<link rel="stylesheet" href="ignou.css" type="text/css" >
<style>p{ color:green;}</style>
</head><body>
<p style="color:red;">BCA:BCS053</p>
</body></html>
```



In the above code, style definition is given at inline, embedded and external level for <p> element. The paragraph data 'BCA:BCS053' is displayed in red colour. Inline styles override the embedded and external style.

In the above code, if you delete the style attribute from paragraph <p> element, then paragraph data will be display in green colour. If you will delete both the styles from inline and embedded level then color of paragraph will be blue.

Style Sheet for more specific tags has priority over general tags. For example, if a page has <body> tag with a certain style definition and an <h1> tag with the same properties and different value then the <h1> tag have the priority, even though it is also part of the body.

Consider the following example for the above definition. In this case, content of heading will be display in red with light grey background.

```
body{color: #000000; background-color: #FFFFFF; }
h1 { color: #FF0000; background-color: #D3D3D3;}
```

For more rules you may please refer to the website <http://www.w3schools.com>

Check Your Progress 1

1. Explain the term CSS.

2. Define the term 'Rule' in respect of CSS.

3. What are different ways to apply styles to a Web page?

4. What is embedded style sheet? Explain with example.

5. What is external Style Sheet? How to link?

2.4 Some Basic Properties in CSS

Previous sections, you have gone through the basics of style sheet and how to use them along with the web document. This section and next section provides details on some of the properties relating to font, lists, color and alignment of text. A detailed discussion on all the properties is beyond the scope of this section. However, you may refer to <http://www.w3schools.com> for more details on various properties.

2.4.1 Font Properties

The font properties are most commonly used style sheet property. It allows you to control the appearance of text in a web document.

***font-family* Property:**

The font-family property allows you to specify typefaces that should be used for the text. The typeface is the name of font.

For example:

`font-family: arial, verdana, sans-serif`

In this case, the browser will use arial if it support that font. If not, it will use verdana if it supports it. If the browser supports neither arial nor verdana, it will use sans-serif. If the browser does not support any of the specified fonts then it will use a default font of its own. If font name has more than one word like Times New Roman, the whole name should be delimited by single quotes as ‘Times New Roman’.

***font-size* Property:**

The font-size property is used to specify a size the font. You can specify a size value for this property in several ways:

- Absolute Size: xx-small, x-small, small, medium, large, x-large, xx-large
- Length: px, pt, em, cm, mm
- Relative Size: smaller, larger
- Percentage: 2%, 10%, 25%, 50%

You can use any of the above value for this property. The most common unit is px for pixels. The pt (points) may also be used.

The following example shows font-size property using different units for heading <h1> to <h4>.

```
h1 { font-size: medium; }
```

```
h2 { font-size: 18px; }
```

```
h3 { font-size: 14pt; }
```

```
h4 { font-size: 10%; }
```

***font-weight* Property:**

This property is used to define a degree of boldness of text. Some value for this property is normal, bold, bolder, lighter, 100 and 200.

For example: The below example shows font-weight property using normal and bold as a value for heading element <h4> and <h5>.

```
h4 {font-weight: normal ;}
```

```
h5 {font-weight: bold ;}
```

***font-style* Property:**

The font-style property is used to specify a normal and italic font for the text.

In following example, normal font style is defined for heading 5 and italic font style for heading 6.

```
h5 { font-style: normal; }
```

```
h6 { font-style: italic; }
```

***font* Property:**

This font property is used as shorthand to declare all of the font properties at once. Consider the following specifications:

```
<p style="font: bold italic 12pt verdana">BCS053</p>
```

This specifies that the font weight should be bold, font style should be italic, font size should be 12 points in verdana typeface.

2.4.2 List Properties

As you know, there are two types of lists namely ordered lists and unordered lists. In ordered lists, the list items are marked with bullets and in unordered lists, list items are marked with numbers or letters.

Using CSS rules, the lists can be more styled and images can be used in place of bullets and numbers as the list item marker. The list properties allow you to control the presentation of list item markers. The different list properties are:

***list-style-type* Property:**

The list-style-type property is used for specifying the ‘list style type’. Using this property, you can set different list item marker for ordered and unordered lists.

For example:

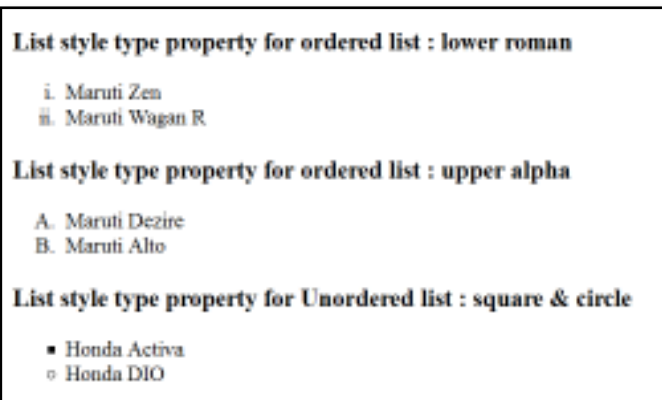
In the below example, there are two classes defined i.e. ‘a’ and ‘b’ which is used with ordered list and another two classes ‘c’ and ‘d’ is used with element of unordered list. When you are used class ‘a’, the list item of ordered list is displayed in lower-roman form. Using class ‘b’, list item are marked as upper-alpha. For unordered list, using class c and d, list markers are displayed in square and circle form.

```

<!DOCTYPE html><html><head><style type="text/css">
ol.a {list-style-type: lower-roman ;}
ol.b {list-style-type: upper-alpha ;}
li.c {list-style-type: square ;}
li.d {list-style-type: circle ;}
</style></head>
<body><h3>List style type property for ordered list : lower roman</h3>
<ol class="a"><li>Maruti Zen</li>
<li>Maruti Wagan R</li></ol>
<h3>List style type property for ordered list : upper alpha</h3>
<ol class="b"><li>Maruti Dezire</li>
<li>Maruti Alto</li></ol>
<h3>List style type property for Unordered list : square & circle</h3>
<ul><li class="c">Honda Activa</li>
<li class="d">Honda DIO</li></ul></body></html>

```

Output of the above program is as follows:



***list-style-position* Property:**

The list-style-position property indicates position of the list item marker. This property contains two values: inside and outside. The inside value places the marker inside the display box and outside places them outside the box. By default, the value of list-style-position property is outside.

The following example shows that the list style positions for ordered and unordered list. For unordered list, list style position of item marker are placed inside the box and for ordered list, position of item marker are outside the box.

```

ul {list-style-position: inside ;}
ol {list-style-position: outside ;}

```

***list-style-image* Property:**

Using this property, you can specify an image as list item marker for unordered list. This property takes precedence over the type of marker specified by the *list-style-type* property.

For example: When you are applied the following rule in your page, list item marker are displayed as an image for unordered list.

```
ul { list-style-image: url("Img.gif"); }
```

***list-style* Property:**

This property is used to set all the properties of list in one declaration.

For example:

```
ul { list-style: circle inside url("Img.gif"); }  
ol { list-style: upper-roman outside ; }
```

2.4.3 Color Property Value Forms

The style definition includes different property values in different categories such as colors, fonts, list, alignments of text and many more. The complete property values for all the categories are beyond the scope of this Unit. You can refer to <http://www.w3schools.com> for complete details. In this section, you will be introduced to one of the most used property value – color value. You can specify a color property as color names such as Red, Yellow etc or a six-digit hexadecimal number with 2 hex digits each for Red Green Blue (RGB) or using RGB form.

Besides the standard color names, there are a wide variety of names used for colors. For example, Blue color has many shades with different names i.e. light blue, dark blue. Some web browsers may not recognize these names. The most reliable way to specify a color in style sheets is hexadecimal code. This is because any Web browser, even an old one, recognises the hexadecimal notation as a color code. The hexadecimal notation is preceded by hash character '#'. As the base of hexadecimal is 16, you could use any one of $16^6 = 16,777,216$ values for a color. In RGB form, you can write *rgb* followed by decimal number between 0 to 255 or as percentages within parenthesis. Here are the depictions of different form of color notation:

```
<!DOCTYPE html>  
<html><head><title>BCA:BCS-053</title>  
</head><body>  
<h3> Example of Color property</h3>  
<ul><li style="color:red"> as name</li>  
    <li style="color:#6600CC"> as hexadecimal number</li>  
    <li style="color:rgb(40%,70%,20%)"> as RGB Form in percentage</li>  
    <li style="color:rgb(176,48,96)"> as RGB Form in decimal number</li>  
</ul></body></html>
```

Output:

Following output of the above program shows the color property using inline style sheet for unordered list. The color code is defined in different forms such as simple color name, hexadecimal number and RGB form.

Example of Color property

- as name
- as hexadecimal number
- as RGB Form in percentage
- as RGB Form in decimal number

2.4.4 Alignment of Text

The ***text-align*** property specifies the horizontal alignment of text in an element. By default, it is left. You can define the alignment of the text by the following manner:

For example: The following program contains the different form of text alignment.

```
<!DOCTYPE html>
<html><head><title>BCA:BCS-053</title><style>
h2 {text-align:center}
h3 {text-align:right}
p {text-align:justify}
</style></head><body>
<h1>CSS text-align Example</h1>
<h2> Alignment of text is centre</h2>
<h3>Alignment of text is right</h3>
<b> By default, it is left</b>
<p>The Alignment of text is <span style="font-weight:bold;"> justified</span>.
stretches the lines so that each line has equal width. </p>
</body></html>
```

Output of the above program:

CSS text-align Example

Alignment of text is centre

Alignment of text is right

By default, it is left

The Alignment of text is **justified**. Stretches the lines so that each line has equal width.

The above CSS text-align property example shows the all forms of text alignment such as left, right, centre and justify.

2.5 Selector Forms

As the names indicate that the selectors are used to ‘select’ element or group of elements on an HTML page. In other words, you can say that selector is a pattern which can apply to different element(s). By using the selector, you can select elements in several ways. The general form of selector is as follows:

Selector{property: value;}

Here are depictions of the different types of selectors:

2.5.1 Simple Selector

The simple selector form is used for a single element, such as <p> element. It is a normal form of writing CSS rule. In this case, rule is applied to all the occurrences of named element. For example,

p {font-weight: bold; color:#336633}

In the above case, rule is written for all the <p> elements in a page. The text is displayed in a bold and green colour. The following rule is described for grouped selectors in which rule is written once but it apply to more than one selector. This is way of shorthand writing of css rule. The rule is applied to both the heading elements.

h1, h2 {font-family: arial, book old style, book antiqua;}

2.5.2 Class Selector

Class selector is used to apply same style specifications to the content of more than one kind of elements with HTML class attribute. In style specification definition, the class attribute is defined with a name which is preceded with a period. Then, in the body of pages, you refer to these attribute name to activate the element tag.

The general syntax for a Class selector is:

.ClassSelectorName {Property:Value;}

For example:

Source code for class selector:

```
<!DOCTYPE html>
<HTML><HEAD><title>IGNOU:BCS-053</title>
<style type="text/css">
.bgcolor_tag{ background-color:yellow;}
</style>
</HEAD><BODY>
<h3 class="bgcolor_tag">Heading is display with background color yellow.</h3>
<p class="bgcolor_tag">Paragraph is defined with background color yellow.</p>
</BODY></HTML>
```

Result:

Heading is display with background color yellow.

Paragraph is defined with background color yellow.

In above example, same class selector is used for heading and paragraph element. You have seen both the elements `<h3>` and `<p>` have background-color in yellow. Class selector is also used when you want to specify the style definition to a specific element then you can use element name followed by dot and class selector name. Consider the following example,

```
<!DOCTYPE html>
<HTML><HEAD><title>IGNOU:BCS-053</title>
<style type="text/css">
p.format {color:red; text-decoration:underline;font-size:20px; font-family:algerian; }
</style></HEAD>
<BODY>
<p>This is normal paragraph</p>
<p class="format">This paragraph is defined with style specification</p>
</BODY></HTML>
```

The output of the above program is

This is normal paragraph

THIS PARAGRAPH IS DEFINED WITH STYLE SPECIFICATION

2.5.3 id Selector

The id selector is used to select any element in an html page with specific or unique id. In style definition, id attribute is defined with hash character “#”. ID selectors are similar to class selectors. The difference between an ID and a class is that an ID can be used to identify one element, whereas a class can be used to identify more than one. The general syntax for an id selector is:

```
#idSelectorName{Property:Value;}
```

For example:

Source code for id selector:

```
<!DOCTYPE html>
<html><head><style>
#para1
{
background-color:yellow;
color:red; font-weight:bold;
text-transform:uppercase;
}
</style></head>
<body><p id="para1">This is an example of id selector.</p>
</body></html>
```

Result: The paragraph text is displayed in red colour with yellow background and text-transform property is converted the lowercase letter to uppercase.

THIS IS AN EXAMPLE OF ID SELECTOR.

Check Your Progress 2

1. What are the different possible values in text-align property?

2. What is font-family? How you can define font-family property for the <p> element in Inline style definition?

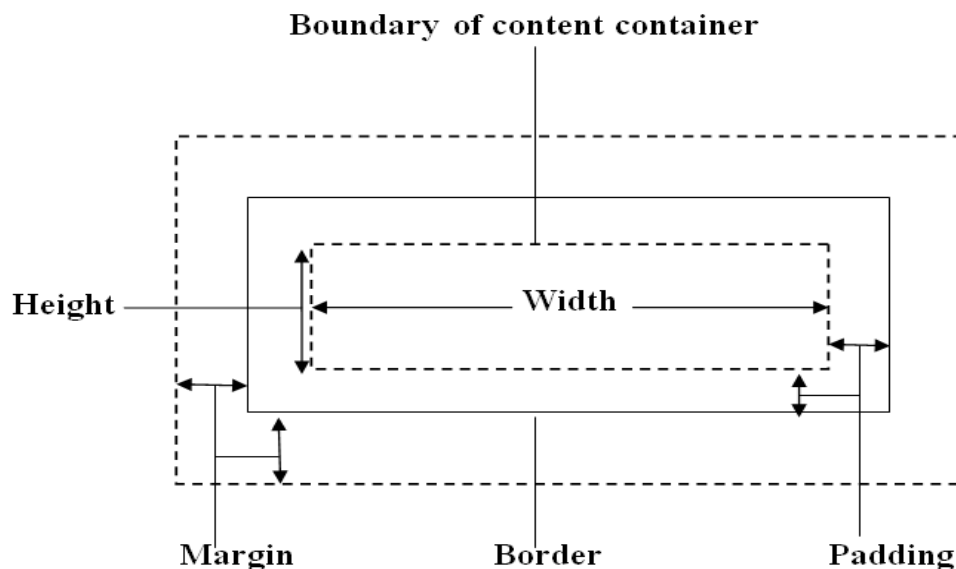
3. Create an HTML document to display the text in the title bar of the browser. “Welcome to 5th semester”.

4. Create a page using BOLD, ITALICS, SMALL, SUPERScript, SUBScript, UNDERLINE element.

5. Explain different forms of color notation in External Style Sheet definition.

2.6 The Box Model

In a document, each element is represented as a rectangular box. In CSS, each of these

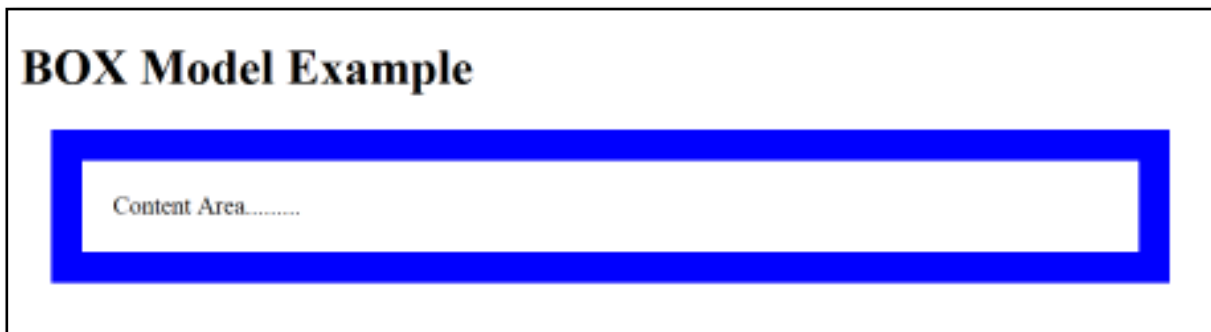


rectangular boxes is described using the box model. Each box has four edges: the **margin** edge, **border** edge, **padding** edge, and **content** edge. Each edge may be broken down into a top, right, bottom, and left edge. This model is shown in the above figure.

The innermost area of box model is content area where the text and images appears. The content area is measured by width and height in terms of pixel and percentages. The padding is the space between the content area of the box and its border which is affected by the background color of the box. The border is surrounding area of padding and content. The border is affected by the background colour of the box. Every element has a property, ***border-style***. You can set the four sides of element with ***border-left-style***, ***border-right-style***, ***border-top-style*** and ***border-bottom-style***. The outermost area of box model is margin which is completely invisible. It has no background color. Like border edge, margin edge has the properties ***margin-top***, ***margin-bottom***, ***margin-left***, ***margin-right*** and padding have ***padding-left***, ***padding-right***, ***padding-top*** and ***padding-bottom*** which is applies to all four sides of an element. You can set these properties in following manner for any element:

```
<!DOCTYPE html>
<html><head><title>BCA:BCS-053</title>
<style>
p { background: white;   border: 20px solid blue;
    margin-top: 20px;    margin-right: 20px;
    margin-bottom: 20px; margin-left: 20px;
    padding-top: 20px;   padding-right: 20px;
    padding-bottom: 20px; padding-left: 20px; }
</style></head><body>
<h1>BOX Model Example</h1>
<p> Content Area.....</p>
</body></html>
```

Output:



You can set all four values using the following shorthand:

padding: [top] [right] [bottom] [left]

For example : **padding: 0 0 0 0;**

p{padding:15px 5px 15px 10px;}

The above css code indicate top padding is 15px, right padding is 5px, bottom padding is 15px and left padding is 10px.

p{padding:10px 5px 15px;}

It shows padding from top(10px), right and left(5px), bottom(15px).

p{padding:20px 15px;} means top and bottom padding are 20px, right and left padding are 15px. In shorthand, you can write **p{padding:15px;}** means all four paddings are 15px.

In the similar way, you can define the margin property also.

2.7 Background Image

Background image, as the name implies, are part of the BACKGROUND of a web page. In HTML page, the most common place to add a background image is the body tag.

The **background-image** property is used to place an image in the background of an element. You can set the background image for a page in CSS like this:

```
body {background-image:url('logo.gif');}
```

By default, the image is repeated so it covers the entire element. This repetition is called tiling. Tiling is controlled by the **background-repeat property** which take the value **repeat** (by default), **repeat-x**, **repeat-y** and **no-repeat**. The repeat-x value means that the image is to be repeated only horizontally, repeat-y means to be repeating vertically and no-repeat specifies that only one copy of the image is to be displayed. You can also set the position for an image by using the **background-position property** which includes values top, center, left, right and bottom. You can set these values in different combinations i.e. right top, top left, bottom right and top center.

Example Source Code for background image:

```
<!DOCTYPE html>
<html><head><title>BCA:BCS-053</title>
<style>
body
{
background-image:url('logo.jpg');
background-repeat:no-repeat;
background-position: top left;
}
h1 {text-align:center}
</style></head><body>
<h1>CSS Background Example</h1></body></html>
```

Outputs:



In the above example, only one copy of image named 'logo.jpg' will be display in top left corner of the screen.

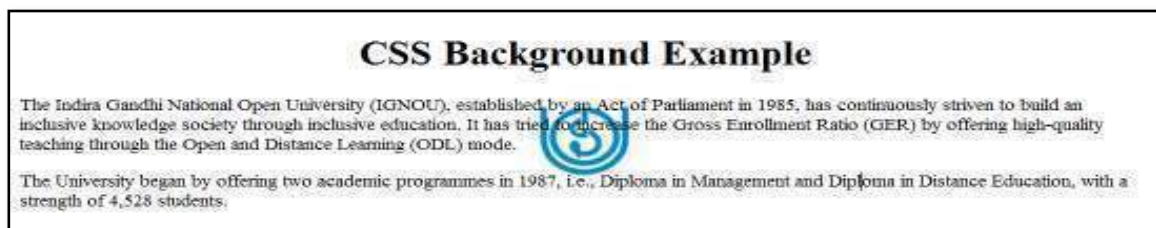
You can also use the shorthand property for the above CSS code:

```
body {background: url('logo.gif') no-repeat top left;}
```

Consider the following example for background image:

```
<!DOCTYPE html>
<html><head><title>BCA:BCS-053</title>
<style>
body
{
background-image:url('logo.jpg');
background-repeat:no-repeat;
background-attachment:fixed;
background-position:center;
}
h1 {text-align:center}
</style>
</head><body>
<h1>CSS Background Example</h1>
<p> The Indira Gandhi National Open University (IGNOU), established by an Act of
Parliament in 1985, has continuously striven to build an inclusive knowledge society
through inclusive education. It has tried to increase the Gross Enrollment Ratio (GER)
by offering high-quality teaching through the Open and Distance Learning (ODL) mode.
</p>
<p>The University began by offering two academic programmes in 1987, i.e.,
Diploma in Management and Diploma in Distance Education, with strength of 4,528
students.
</p>
</body></html>
```

Output of the above program:



In the above example of background, only one copy of image will be display in the centre of the page. The background-attachment property is used to fix the position of the image. If the length of content is large, then page will be scroll but the image position is fixed in the centre.

2.8 The <div> and Tags

Both the elements <div> and are used to group and structure an HTML page and will often be used together with the selector attributes *class* and *id*.

The **div** (short for Division) elements define logical divisions in a web page. The <div> tag is used as a container for a group of elements. It is not seen on a webpage, but works behind the scene to organize the layout of a webpage a certain way. It acts a lot like a <p> element, by placing new lines before and after the division. A division can have multiple paragraphs in it.

The <div> element is used to define the style of whole sections of the HTML page whereas the element is used to style a part of a text. Div is a block-level element, while span is an inline element. Just to remind you that a block level html element is displayed from a new line like <p>, whereas an inline element is appended to the same on going line.

The **span** element has very similar properties to the <div> element, in that it changes the style of the text it encloses. But without any style attributes, the element will not change the enclosed items at all. To use the element, simply surround the text that you want to add styles to with the and tags. For example:

```
<!DOCTYPE html>
<html><head><title>BCA:BCS-053</title>
<style>
  .format{ color:#FF0000; background-color:#99FFCC;}
  #edit{background-color:#9966CC; color:#FFD700;}
</style>
</head><body>
<h2>Div & Span Tags Example</h2>
<div id="edit">
<ul><li> <span class="format">Republic Day </span>
of India which is celebrated on 26<sup>th</sup> January. </li>
<li> India gained its independence on
<span class="format">15<sup>th</sup> August 1947</span>. </li>
</ul></div></body></html>
```

Output of the above example is as follows:



In the above program, the highlighted text ‘*Republic Day*’ and ‘*15th August 1947*’ are displayed in red text color and cyan background by element and rest of the yellow text with purple background by <div> element. Div tags can be used for organising different sections of a web page using different position attributes. A detailed discussion on this is beyond the scope of this Unit. However, we explain the essence of this with the help of example in the next section.

Check Your Progress 3

1. How do you include comments in CSS?

2. Explain the box model and its utility with figure.

3. Write CSS code for including a background image without repeating in a page?

4. What is the difference between <div> and tag?

5. Define the CSS padding property.

2.9 CSS Complete Example

In earlier websites frames and tables were used to organise contents of different webpages. However, div tag and float property provides an alternative to such web page design. The following example explains such a design in more details. Some times it is referred to as table less web layout. It is a method of web design and development without the use of HTML tables for page layout control purposes. Using the DIV-based layout, you can change the entire look of the web site by just changing the definitions in the CSS file. The CSS float property is a very important property for layout. You cannot float every element on a Web page. It can be used only for block-level elements such as images (), paragraphs (<p>), divisions (<div>) and lists ().

The float property is used to specify an element to float either left or right in a web page. It has different property values such as left - the element floats to the left and right - the element floats to the right. Another CSS property, clear is used to clear the floated elements. It has three options such as left – clears all left-floated elements, right – clears all right-floated elements and both - clears all floated elements.

In the given example, the web page contains five sections i.e. Header - display a heading, Left – used for menu like navigation, Content - text of the web page, Right – display events details and Footer - for copyright information. The external style sheet is created for this example named CSSLayout.css. Another file main.html is also created for the web page. The CSSLayout.css file is included in main document using link element. Inside the CSSLayout.css file, all style specifications are written which applicable for the main document.

Source Code for CSSLayout.css

```
body{color:#000; font-size:18pt; margin:0; padding: 0; }
.container{width:100%; }
.left{float:left; background:#ffcc00; width:300px; height:500px; padding-left:
20px;}
.right{float:right; background:#ff00cc; width:300px; height:500px; padding-left:
20px;}
.header{background:#ff0000; text-align:center;font-size:30pt; height:100px;}
.content{ float:left;width:600px; text-align:justify;}
.footer{clear:both; background:lightblue; font-size:15pt; }
```

Please note that .container, .left, .header, .content and .footer are all classes. The property of these classes has been defined in the CSS given above. Please also note how are they referred in the main.html file.

Source Code for main.html

```
<!DOCTYPE html><html><head>
<link type="text/css" rel="stylesheet" href="CSSLayout.css" /></head><body>
<div class="container">
<div class="header">Indira Gandhi National Open University<br>SOCIS</div>
<div class="container">
<div class="left"><menu>
<li><a href="About.html">About Us</a></li>
<li>Programme</li>
<li>Results</li>
<li>Contact Us</li>
</menu></div>
<div class="content"><h3>Table Less DIV Based Layout using Float</h3>
Content within a div tag will take up as much space as it can, namely 100% of
the width of the current location, or the page. So, to affect the location of a
section without forcing it to a fixed width, you can change the padding or the
margin elements. </div>
<div class="right">Event<br>
<details> <summary>BCS:053</summary>
<p>Unit 1: Web 2.0 and HTML5<br>
Unit 2: Style Sheets<br>Unit 3: XML<br>
Unit 4: JavaScript<br>Unit 5: WAP & WML</p></details>
<details> <summary>Datesheet</summary>
<p>Datesheet for BCA students</p></details>
<details> <summary>Prospectus</summary>
<p>Prospectus for BCA students</p></details>
</div></div>
<div class="footer"><footer><hr> &copy; 2013 SOCIS, IGNOU. Design &
developed by Poonam Trikha</footer></div>
</div></body></html>
```

The output screen-1 for the above program is as follows:



In the above output screen-1, header part shows the heading of the page and left column show the navigation item names such as About Us, Programme, Results and Contact Us. The navigation names are displayed by using the menu element. (You can remove the dots in this list using a CSS property; it is left as an exercise for you.) Please note that each of these navigation elements is pointing to another html file. The centre part is for writing the text of the web page and right column is displayed the events & schedule for the students by using details and summary tags. The `<summary>` tag defines a visible heading for the `<details>` element. You can click on the heading to view/hide the details. In this example program, the heading BCS053 and Datesheet is displaying the summary information and prospectus heading hide the details. The second output screen shows the same information regardless the position of the left and right column. For the second program, you can write the same program and just exchange the position of left and right column.

The difference between the following program and the above program is that the position of left column is replaced with the right column. You can run the above and following program to see differences. Please note that the program is run only in Google Chrome due to the `<summary>` and `<details>` tags.

```

<!DOCTYPE html><html><head>
<link type="text/css" rel="stylesheet" href="CSSLayout.css" />
</head><body>
<div class="container">
<div class="header">Indira Gandhi National Open University<br>
SOCIS</div>
<div class="container">
<div class="left">Event<br>
<details> <summary>BCS:053</summary>
<p>Unit 1: Web 2.0 and HTML5<br>Unit 2: Style Sheets
<br>Unit 3: XML<br>Unit 4: JavaScript
<br>Unit 5: WAP & WML</p></details>

<details> <summary>Datesheet</summary>
<p>Datesheet for BCA students</p></details>

<details> <summary>Prospectus</summary>
<p>Prospectus for BCA students</p></details>

</div>
<div class="content"><h3>Table Less DIV Based Layout using
Float</h3>Content within a div tag will take up as much space as it can,
namely 100% of the width of the current location, or the page. So, to affect
the location of a section without forcing it to a fixed width, you can change
the padding or the margin elements. </div>

<div class="right"><menu>
<li><a href="About.html">About Us</a></li>
<li>Programme</li><li>Results</li><li>Contact Us</li></menu></div>

</div>
<div class="footer"><footer><hr> &copy; 2013 SOCIS, IGNOU. Design &
Developed by Poonam Trikha</footer></div>
</div></body></html>

```

The output screen-2 for the above program is as follows:



2.10 Summary

You have learned more about the Cascading Style Sheet. It is a mechanism for adding styles such as font, color, images in background to web pages. The CSS specifications are maintained by the World Wide Web Consortium (W3C). There are three ways to include CSS code in your web pages viz., inline, embedded and external. In inline style definition, you can include the style definition within the tag. Inline styles are easy and quick to add. You do not need to create a whole new document as with external style sheets. The inline style sheet has high precedence over the other embedded and external style sheet. But, Inline style is not the best way to design a web page because it can be more difficult to find out where a style definition or rule is being set. Instead of using Inline style, you can use embedded style definition in your page. It is placed within the head section of the page. But the best way to include style in your web pages is external style sheet. In external style sheet, you can write style rules once and it is reflected in all the web pages in your site. You have also learned about the font, list, color property and box model. In CSS, box model is used for the layout of the web page. The div is most of the important element in html because in combination with CSS properties, it can give more effect in web pages. You can use any element in div tag. By using div tag, you can create a block level section in the web page. Each section can have its own formatting. Span is similar to div element. Both divide the content into individual sections. The difference is that span goes into a finer level, so you can span to format a single character if needed. In this unit, there is more exercise for you in form of questions.

You have learned HTML5 (in previous unit) and CSS rules. Now, you can design simple web page using CSS. Try to develop web sites by linking various web pages.

2.11 Solutions/Answers

Check Your Progress 1

Ans.1: CSS is a web standard that describes style for XML/HTML documents. Cascading style sheet is used to control the layout of multiple pages at once. The CSS code is written in a separate file with extension .css. It is a mechanism to write and modify style definition in css file, and the changes will be reflected in all the web pages in which the css file is linked with link element. The link element should be placed in between <head> and </head> tags. Besides external CSS file, you can use css code at embedded and inline style definition in web page.

Ans.2: The CSS rule is a way of writing a code that allows web designers to define styles definition easily and effectively for HTML pages. For example, <p style="font-weight: bold">some text</p>.

Ans.3: There are three ways to insert CSS into web pages by using inline, document and external style sheet. For more detail, refer to section 2.3 in this unit.

Ans.4: body {background: url('logo.gif') no-repeat top left;}

Ans.5: External style sheet is a file containing only style definition code which can be linked with any number of HTML documents. This is the way of formatting the entire site. External style sheet must have extension .css i.e. style.css. The file is linked with HTML documents using the LINK element inside the HEAD element. The following is the way to use external style sheet in a page. <link rel="stylesheet" href="name_of_file.css" type="text/css" >

Check Your Progress 2

Ans.1: The text-align property is used for horizontal alignment of the text. By default, it is left. Text can either be aligned to the left, right, centred and justify.

Ans.2: The font-family property specifies the font for an element. The font-family property can hold several font names such as arial, Tahoma and many more. If the browser does not support the first font, it tries the next font and so on.

<p style="font-family: verdana, tahoma, arial;">

Ans.3: <!DOCTYPE html><html><head><title>Welcome to 5th Semester</title></head><body></body></html>

Ans.4: <!DOCTYPE html><html><head><title>BCA:BCS053</title></head><body>
This line is BOLD
<u>This line is UNDELINE</u>

<i>This line is ITALICS</i>
<small>The font of this line is small </Small>

<p>H₂o</p><p>2³=8</p></body></html>

Ans.5: You can use different color notation for the web page such as color name, hexadecimal form and RGB form. You can use the following program for the color notation. You can write external style sheet for the program as the following named style.css:

```
p.a{color:blue;}
p.b{color:#ff0000;}
p.c{color:rgb(40%,60%,40%);}
p.d{color:rgb(120,60,96);}
```

Now, you can include the above style.css file in the following program:

```

<!DOCTYPE html>
<html><head><title>BCA:BCS-053</title>
<link rel="stylesheet" href="style.css" type="text/css" >
</head><body><h3> Example of Color property</h3>
<b><p class="a">Color property as name</p>
<p class="b">Color property as hexadecimal number</p>
<p class="c">Color property as RGB Form in decimal number</p>
<p class="d">Color property as RGB Form in percentage</p> </b>
</body></html>

```

Check Your Progress 3

Ans.1: You can placed anything between /* and */ in CSS is considered as a comment. Comments are ignored by the web browser.

Ans.2: The term ‘box model’ is used in CSS-based layout and design. Every element in HTML can be considered as a box, so the box model applies to all html and xhtml elements. Each box has four edges: the margin edge, border edge, padding edge, and content edge. Each edge may be broken down into a top, right, bottom, and left edge. For more details, pl. refer to 2.5 section.

Ans.3: `body { background-image:url('logo.jpg'); background-repeat:no-repeat; background-attachment:fixed; background-position:center; }`

Ans.4: The <div> element is used to define the style of whole sections of the HTML page whereas the element is used to style a part of a text. Div is a block-level element, while span is an inline element.

Ans.5: The CSS padding properties define the space between the element border and the element content. In CSS, it is used to specify different padding to all four sides of an element. For example :

```

padding-top:25px;
padding-bottom:25px;
padding-right:50px;
padding-left:50px;

```

2.12 Further Reading

- 1) Beginning Web Programming with HTML, XHTML and CSS by Jon Duckett. Publisher: Wrox.
- 2) HTML & CSS :design and build website by Jon Duckett.
- 3) Programming the World Wide Web by Robert W. Sebesta. Publisher:Pearson
- 4) HTML 4 Unleashed by Rick Darnell. Publisher: Techmedia
- 5) <http://www.w3schools.com>

Unit 3: Introduction to XML

Structure

- 3.0 Introduction
- 3.1 Objectives
- 3.2 What is XML?
 - 3.2.1 Need of XML
 - 3.2.2 Various Terms of XML
 - 3.2.3 XML Document Structure
 - 3.2.4 XML Namespaces
- 3.3 Validating XML Documents
 - 3.3.1 Document Type Definitions (DTD)
 - 3.3.2 XML Schemas
- 3.4 Displaying XML files
 - 3.4.1 XML CSS
 - 3.4.2 XML XSLT
- 3.5 Summary
- 3.6 Answers to Check Your Progress
- 3.7 References

3.0 Introduction

XML is widely used for the definition of platform-independent method of storing and processing texts in electronic form. It is also the interchange and communication format used by many applications on the Internet. In the present chapter we introduce its basic concepts and attempt to make the reader independent to use its various applications.

This Unit will help you to use the various topics covered under Unit1 and Unit2. As we know now, that XHTML is nothing but HTML defined as an XML application. So, to design the cleaner XHTML pages, knowledge of XML is a must. This unit will help you to write the valid xmls, with formatting features using style sheets, as learned in Unit 2.

3.1 Objectives

- Define the use of XML
- Define the basic terminology used in XML
- Create an XML document
- Create XML Schema and DTD
- Verify XML documents

3.2 What is XML?

XML stands for eXtensible Markup Language. Very much like HTML, XML is also a markup language. The essence of XML is in its name:

Extensible

It lets you define your own tags. Tags are displayed in the same order as they are defined in your xml document.

Markup

The most important feature of any xml document is its tags, also called as elements. These are very much similar to the ones which are being used at html.

Language

XML is a language similar to HTML, though more flexible giving you option to create your own custom tags. Basically we can call it as meta-language: it allows us to define or create other languages. For example, we can create RSS, MathML languages and tools such as XSLT.

3.2.1 Need for XML

As we know that HTML is designed to display elements in a web browser. It becomes cumbersome if you want to display the documents in some other language or upgrade the document for dynamic data display. XML is just suited for this purpose. It can be used in a variety of contexts. XML is used to store data in plain text format with strong support via Unicode, for the various languages of the world. This makes it easy to carry data irrespective of any platform. XML is also used as the base language for the communication protocols such as XMPP. It is also very simple and easy to learn and use.

XML provides users the flexibility to define their own tags. Using meaningful tags you can structure the data, which makes it easier to transport later on and define their document structure

XML as a document, does not do anything. It is the software which uses this document to connect with the Database and does the processing of data. XML is recommended by W3C as well, in February 1998 for the purpose of storing and transporting data. XML is so flexible and powerful that it is used in creating new Internet Languages such as XHTML(Extensible HyperText Markup Language now replaced by HTML5), WML(Wireless Markup Language), SMIL(Synchronized Multimedia Integration Language) etc. In HTML, users are supposed to use pre-defined tags only such as <p>, <table>, , <i> etc. thus providing limited flexibility to the users.

Precisely we can say that HTML is a presentation language, and XML is a data-description language.

Lets have a look at the sample program below:

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

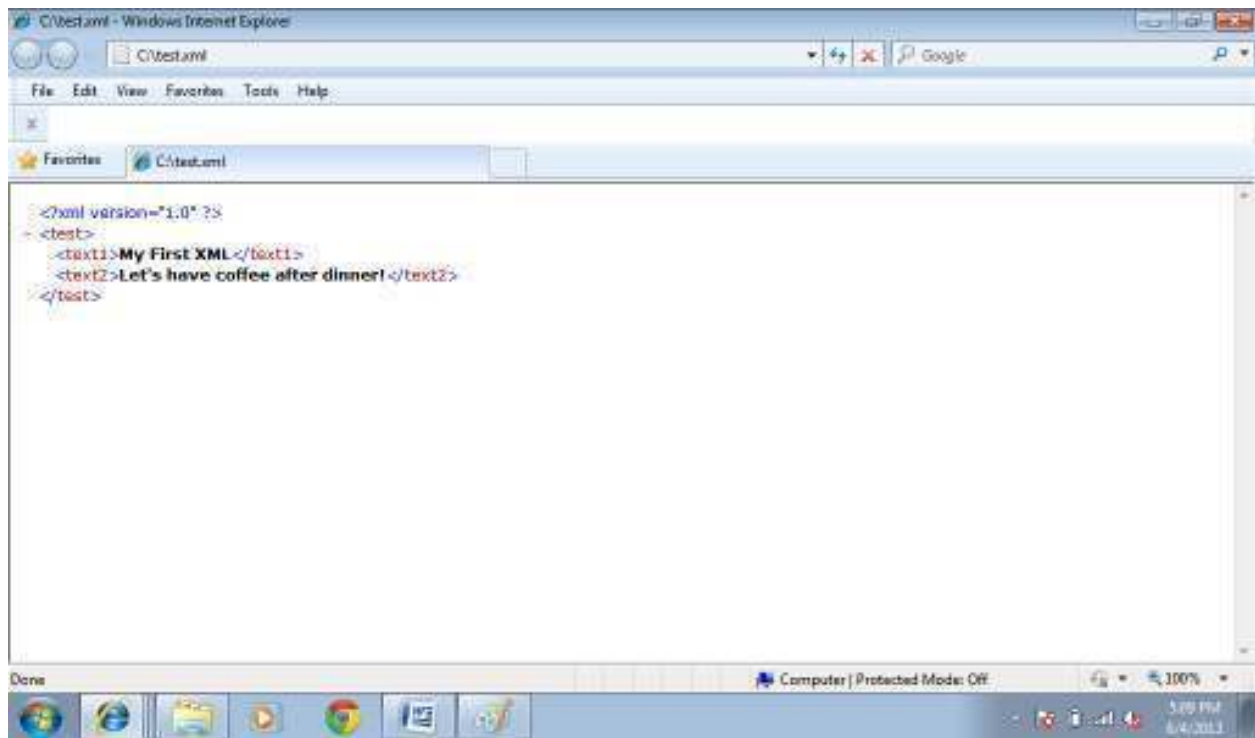
<test>

    <text1>My First XML</text1>

    <text2>Let's have coffee after dinner!</text2>

</test>
```

Write this text in simple notepad and save it with the extension “.xml”. Now, open this file using Internet Explorer (or any other browser) and you will see the output as :



Unlike in HTML, here user has to be particular about missing end tag, use of different capitalization in opening and closing tags, and improperly nested tags. Here `<text1>` tag is different from `<TEXT1>` tag. If opening tag is `<text1>` and closing tag is `</Text1>`, document will be incorrect.

3.2.2 Various Terms of XML

Let us understand the various terms used in this example:

Tags

A markup construct begins with `<` and ends with `>`. Tags are defined as:

- Start tags ; e.g. `<test>` in the example above
- End tags ; e.g. `</test>` in the example above
- Empty-element tags; for example: `<line-break />`

Element

A logical document component that begins with start tag and end with a matching end-tag. It can also contain an empty- element tag. The characters between start-tag and end-tag are the element's content, and may also contain other elements which are called as child elements.

Attribute

A markup which contains a name/value pair existing between start-tag and end-tag.

Now, as we discussed above that XML is extensible as well, if we update the XML by adding or removing some tags, it immediately reflects the same on the click of refresh button on the browser. For example, if we update the example above as:

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

<test>

    <text1>My First extensible XML</text1>

    <text2>Let's have coffee after dinner!</text2>

    <text3>After that, let's go for a long drive.</text3>

</test>
```

And it will look like this, upon refresh:

```
<?xml version="1.0" ?>
- <test>
    <text1>My First extensible XML</text1>
    <text2>Let's have coffee after dinner!</text2>
    <text3>After that, shall we go for a long drive?</text3>
</test>
```

3.2.3 XML Document Structure

Let's understand the XML document structure in detail here. XML documents form a tree-like structure that has root and various branches.

In the following example XML, let us understand its various parts:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<test>
```

```
<text1>My First XML</text1>
```

```
<text2>Let's have coffee after dinner!</text2>
```

```
</test>
```

- The First line, it is the XML declaration. This line defines the XML version and the encoding used. XML version is 1.0 here and the encoding is ISO-8859-1.
- The next line is the root element of the document. It describes that this is the test document.
- Next 2 lines describe the 2 child elements of the root.
- The last line describes the end of the root element.
- Comments can be written in XML in the same way as in HTML: `<!--This is a comment-->`

XML ELEMENTS

All the XML documents must have the ROOT Element to mark the beginning and end of the document tree. Each XML document may have the nested level of the sub-elements, attributes, text. For Example:

```
<Root>
```

```
<Parent>
```

```
    <Node1>
```

```
        <subNode1></subNode1>
```

```
        <subNode2></subNode2>
```

```
    </ Node1>
```

```
    <Node2>
```

```
        <subNode1></subNode1>
```

```
        <subNode2></subNode2>
```

```
    </ Node2>
```

```
</Parent>
```

```
</Root>
```

Attributes

Attributes can be called as adjectives, and are used, wherever additional info is required to be displayed for any element. However the information stored in attributes should not contain any data part. It is very difficult to maintain the data in attributes, as data is dynamic and becomes difficult to maintain as a part of the attribute. Elements are the best option for storing data.

Another important point is that Attribute values shall always be quoted. For Example:

```
<Student name="Ram Kumar Sharma">
</student>
```

Empty-Element tag

Some elements are said to be empty like
, . In all info is contained in its attributes.
 does not signify any value or attributes, it just means a line break. For empty elements, we need not specify their closing tags. E.g.

```
<emptyElementTag/>
```

The / at the end signifies that the element starts and ends here. Its an efficient shortcut method to mark up empty elements.

Lets use elements and attributes in the example below:

If we take Automobile as Parent, we may have various elements like cars, trucks, bikes etc.

```
<Automobile>
  <car type = "SUV">
    <name>Scorpio</name>
    <make>2010</make>
    <price>11,000,00</price>
  </car>
```

```
<car type="MUV">
    <name>Xylo</name>
    <make>2011</make>
    <price>9,000,00</price>
</car>

<car type="SEDAN">
    <name>Honda City</name>
    <make>2010</make>
    <price>9,500,00</price>
</car>

</Automobile>
```

In this example,

<automobile> is the Root Element.

<car> is the Element, because it contains other elements. <car> also has one attribute i.e. "type" having various values such as "SUV", "MUV" & "SEDAN". Based on their attribute values, text contents for the following sub-elements is mentioned:

<name>, <make>, and <price> .

Rules

While designing the XMLs, we shall follow certain rules as mentioned below:

XML Syntax Rules:

- XML Tags are case sensitive
- All XML elements shall have closing tags
- XML Elements shall be properly nested
- XML Attribute values must be in quotes.

- Some Special characters have entity reference. It means that few special characters are already used as HTML pre-defined tags. So If we need to use those characters, we shall write them in a different way as below:

<	<	less than
>	>	greater than
&	&	ampersand
'	'	Apostrophe
"	"	quotation mark

Now, while naming the XMLs also, we shall follow certain rules such as:

- Names can contain letters, numbers, and other characters
- Names cannot start with a number or punctuation character
- Names cannot start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces

Other than these, any name can be used, as no words are reserved.

3.2.4 XML Namespaces

XML allows you to define your own documents thus providing flexibility and scope. But it's a strong possibility that, when combining XML content from different sources, there could be clashes between codes in which the same element name serve very different purposes. For example, if you are using a bookstore, your use of <title> tag in xml may be for the used for tracking the book's title. A insurance agent may use the <title> in a different way – to track the income tax records of its clients. A doctor may use the <title> to track patient's formal titles (like Mr., Mrs., Dr., Ms. etc.) on the medical records. While trying to combine these xmls in one system, problem will arise.

XML namespaces provide a method to avoid element and attribute name conflicts in any XML document. XML namespaces attempt to keep different

semantic usages of the same XML elements separate and unambiguous. In this example, each person could define their own namespace and then prepend the name of their namespace to specific tags: <book:title> is different from <broker:title> and <medrecs:title>.

An XML namespace is declared using the reserved XML pseudo-attribute xmlns or xmlns:prefix, the value of which must be a valid namespace name. We can also use a default namespace, which will be implicit in all the child elements. It has the syntax as follows:

xmlns:prefix="namespaceURI"

A namespace is named as Uniform Resource identifier (URI), identifying an internet resource uniquely, though doesn't point to an actual location of the resource. Uniform Resource Locator (URL) is the most common form of URI used. Another one is Universal Resource Name (URN) which is not so widely used.

Any namespace declaration that's placed in a document's root element becomes available to all the elements in that document. Please note that namespaces have scope. Namespaces affect the element in which they are declared, as well as all the child elements of that document.

For Example:

```
<root xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3schools.com/furniture">

<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

In this example, the xmlns attribute in the <table> tag give the h: and f: prefixes a qualified namespace. So there will not be any name conflict due to same <table> - element name, both having different content and meaning.

When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace. Namespaces can be declared in the elements where they are used or in the XML root element. We can also define default namespaces, which will help us in not prefixing any element in the xml document, e.g.:

```
<root xmlns=http://www.w3.org/TR/html4/
```

Check Your Progress 1

1 What is a well-formed XML document?

2 What is a valid XML document?

3 What are the rules for naming XMLS?

3.3 Validating XML Documents

XML provides two major ways of validating an XML documents- Document Type Definitions and XML Schema. Let us discuss them in more details in the next sub sections.

3.3.1 Document Type Definitions(DTD)

A Document Type Definition (DTD) defines the legal building blocks of an XML document. It defines the document structure with a list of elements and attributes. If DTD is supplied with the XML file, then the XML parser will compare the content of the document with the rules that are set in the DTD. If the document does not conform to the rules specified by the DTD, the parser raises an error and indicates where the processing failed.

A DTD can be declared inline inside an XML document, or as an external reference.

Internal DTD Declaration

If the DTD is declared inside the XML file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<!DOCTYPE root-element [element-declarations]>
```

OR

External DTD Declaration

If the DTD is declared in an external file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<!DOCTYPE root-element SYSTEM "filename">
```

Example DTD with Internal declaration:

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

```
<!DOCTYPE test [
```

```
<!ELEMENT test (text1,text2,text3)>
```

```
<!ELEMENT text1 (#PCDATA)>
```

```
<!ELEMENT text2 (#PCDATA)>
```

```
<!ELEMENT text3 (#PCDATA)>
```

```
] >
```

```
<test>
```

```
    <text1>My First extensible XML</text1>
```

```
    <text2>Let's have coffee after dinner!</text2>
```

```
    <text3>After that, shall we go for a long drive?</text3>
```

```
</test>
```

Now, run this XML in the browser and the result would be displayed like this:

```

<?xml version="1.0" ?>
<!DOCTYPE test (View Source for full doctype...)>
- <test>
  <text1>My First extensible XML</text1>
  <text2>Let's have coffee after dinner!</text2>
  <text3>After that, shall we go for a long drive?</text3>
</test>

```

You can see the source for the doctype definition.

The DTD above is interpreted like this:

- !DOCTYPE test defines that the root element of this document is test
- !ELEMENT test defines that the test element contains three elements: "text1,text2,text3"
- !ELEMENT text1 defines the text1 element to be of type "#PCDATA"
- !ELEMENT text2 defines the text2 element to be of type "#PCDATA"
- !ELEMENT text3 defines the text3 element to be of type "#PCDATA"

Example DTD with External declaration:

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

```
<!DOCTYPE test SYSTEM "test.dtd">
```

```
<test>
```

```
  <text1>My First extensible XML</text1>
```

```
  <text2>Let's have coffee after dinner!</text2>
```

```
  <text3>After that, shall we go for a long drive?</text3>
```

```
</test>
```

And test.dtd is as mentioned below:

<!ELEMENT test (text1,text2,text3)>

<!ELEMENT text1 (#PCDATA)>

<!ELEMENT text2 (#PCDATA)>

<!ELEMENT text3 (#PCDATA)>

DTD defines the main building block of any XML document.

PCDATA here means **parsed character data**. PCDATA is text that WILL be parsed by a parser. The text will be examined by the parser for entities and markup. Tags inside the text will be treated as markup and entities will be expanded.

However, parsed character data should not contain any &, <, or > characters; these need to be represented by the & < and > entities, respectively.

CDATA

CDATA means character data. Think of character data as the text found between the start tag and the end tag of an XML element.

CDATA is text that will NOT be parsed by a parser. Tags inside the text will NOT be treated as markup and entities will not be expanded.

Attribute Declaration in DTD

Attributes are declared with an ATTLIST declaration.

Syntax:

<!ATTLIST element-name attribute-name attribute-type default-value>

For Example, in DTD :

```
<!ATTLIST car type CDATA "SUV" >
```

In XML,

```
<car type= "SUV"></car>
```

The default value of an attribute can be :

REQUIRED

IMPLIED

EMPTY

DTD:

```
<!ELEMENT car EMPTY>  
<!ATTLIST car price CDATA "100000">
```

Valid XML:

```
<car price ="1000000" />
```

Here "car" element is defined to be empty element with price attribute of TYPE CDATA . If no price is specified, it has default value of 1 L.

#REQUIRED

DTD:

```
<!ELEMENT car >  
<!ATTLIST car price CDATA #REQUIRED>
```

Valid XML:

```
<car price ="1000000" />
```

Here "car" element is defined WITH REQUIRED attribute value. If no price is specified, XML parser will throw an error. So, user is forced to specify the value.

#IMPLIED

DTD:

```
<!ELEMENT car >  
<!ATTLIST car price CDATA #IMPLIED>
```

Valid XML:
<car price ="1000000" />

Here "car" element is defined with IMPLIED attribute value. It means user is not forced to specify any price value nor any default value is available for this.

FIXED

Syntax

```
<!ATTLIST element-name attribute-name attribute-type #FIXED "value">
```

Example

DTD:
<!ATTLIST car manufacturer CDATA #FIXED "Mahindra">

Valid XML:
< car manufacturer ="Mahindra" />

Invalid XML:
< car manufacturer ="Hyundai" />

Use the #FIXED keyword when you want an attribute to have a fixed value without allowing the author to change it. If an author includes another value, the XML parser will return an error.

Benefits of using DTD:

1. DTDs allow the declaration of standard public entity sets for publishing characters.
2. DTD support is ubiquitous due to its inclusion in XML 1.0
3. DTDs are terse compared to element-based schema and thus present more information in a single screen.

4. DTDs define a document type rather than those defined by the namespaces. It helps in grouping all document constraint in a single collection.

Limitations of DTDs:

1. They have no explicit support for namespaces. All DTD declarations are global, so you can't define two different elements with the same name, even if they appear in different contexts.
2. DTDs have support only for rudimentary datatypes.
3. DTDs lack readability.
4. DTDs use regular expression syntax to describe schema. This is less accessible to programmers than an element-based syntax may be.

3.3.2 XML Schemas

XML Schema is described as successor to DTD. It also describes the structure of an XML document., and so is also referred to as XML scheme definition (XSD). XSDs are more powerful than DTDs in describing XML language. They use a rich data typing system and allow for more detailed constraints on an XML document's logical structure. The great strength about schemas is that they are written in XML. So, if you know XML, working on XML schema or XSDs doesn't require any special efforts.

The XML schema defines:

- Elements and attributes of an XML document.
- Data types for elements and attributes
- Default or fixed values for elements and attributes
- the child elements
- the number and order of child elements
- empty or non-empty elements

XML Schemas are eXtensible, because they are written in XML. So they can be used in :

- reusing schemas in other schemas
- creating your own datatype derived from the standard type
- reference multiple schemas from the same document.

XMLs schemas also support data types, so with this support its simple to:

- describe document content
- validate data correctness
- work with databases
- define data facets
- define data pattern
- convert data between different data types

With XML schemas, its easier to send or receive data from a sendor to a receiver with the same expectations. For example, a date is always misinterpreted in different countries based on the date format they follow.

Some follow mm/dd/yyyy or some use dd/mm/yyyy. In XML, we can define date as :

```
<Date type = "date">2013-08-06</date>
```

Here XML date datatype requires the format as "yyyy-mm-dd", thus avoid any chances of confusion.

XML Schema has a lot of built-in data types. The most common types are:

- xs:string
- xs:decimal
- xs:integer
- xs:Boolean
- xs:date
- xs:time

Here are some examples of XML elements:

```
<name>Refsnes</name>
```

```
<age>36</age>
```

```
<dob>1977-08-06</dob>
```

And here are the corresponding simple element definitions:

```
<xs:element name="name" type="xs:string"/>
```

```
<xs:element name="age" type="xs:integer"/>
```

```
<xs:element name="dob" type="xs:date"/>
```

Example XML

XML documents can refer to a DTD or an XML schema, as per example below:

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

```
<test>
```

```
    <text1>My First extensible XML</text1>
```

```
    <text2>Let's have coffee after dinner!</text2>
```

```
    <text3>After that, shall we go for a long drive?</text3>
```

```
</test>
```

DTD File is as follows: **test.dtd**

```
<!ELEMENT test (text1,text2,text3)>
```

```
<!ELEMENT text1 (#PCDATA)>
```

```
<!ELEMENT text2 (#PCDATA)>
```

```
<!ELEMENT text3 (#PCDATA)>
```

Now XML Schema file is as follows: test.xsd

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

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
```

```
<xs:element name="test">
```

```
  <xs:complexType>
```

```

    <xs:sequence>
      <xs:element name="text1" type="xs:string"/>
      <xs:element name="text2" type="xs:string"/>
      <xs:element name="text3" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>

```

In this XSD file, <schema> element is the root element of every XML schema. It has the following attributes.

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

It says that the data types and elements are from this default namespace.

```
targetNamespace="http://www.w3schools.com"
```

It indicates that the elements defined by this schema (test, text1, text2, text3) come from the "http://www.w3schools.com" namespace.

```
xmlns="http://www.w3schools.com"
```

It indicates that the default namespace is "<http://www.w3schools.com>".

```
elementFormDefault="qualified"
```

It says that any element declared in this schema shall be used by only those XML instance documents which are namespace qualified.

A reference to this XML schema file is in the following XML:

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

```
<test xmlns="http://www.w3schools.com"
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
  xsi:schemaLocation="http://www.w3schools.com test.xsd">
```

```
    <text1>My First extensible XML</text1>
```

```
    <text2>Let's have coffee after dinner!</text2>
```

```
    <text3>After that, shall we go for a long drive?</text3>
```

</test>

Here following code fragment means:

```
<test xmlns="http://www.w3schools.com">
```

the default namespace is "http://www.w3schools.com".

This fragment means :

```
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
```

the elements(test, text1, text2, text3) and data types used in the schema are come from the "<http://www.w3.org/2001/XMLSchema-instance>" namespace. It also specifies that elements and attributes that come from the "<http://www.w3.org/2001/XMLSchema-instance>" namespace shall be prefixed with xsi:

This fragment means:

```
xsi:schemaLocation="http://www.w3schools.com test.xsd">
```

First value is the namespace used for all the elements in the xml file and second value is the location of the xml schema used for that namespace.

XML elements can contain values for various datatypes available.

Check Your Progress 2

1. Which are 2 methods to validate XML files?

2. What are the limitations of DTD?

3. Write down the default built-in data types for XML schema.

3.4 Displaying XML files

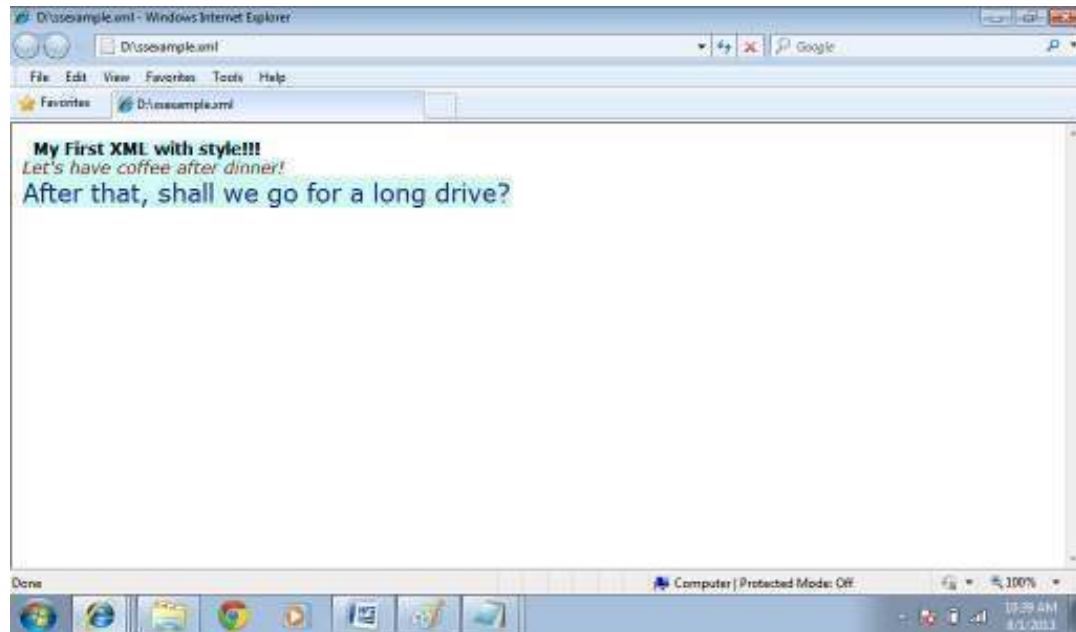
XML documents are plain text files just like HTML files and are displayed with a browser, in a tree-like structure.

As browser has no information how does the tags look like, to enhance the readability, we shall add some style to it. This can be done using css i.e. Cascading Style Sheets. CSS files are applied to HTML as well to enhance the

readability. An XML document can be displayed in a web browser using a CSS and a XSLT. The following sections describe them in details:

3.4.1 XML CSS

If a browser supports css and xml, you can use css and the formatted example xml document look like this:



In this example, please ensure that the xml and css files are stored in the same location on the drive.

Code for the XML file:

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

```
<?xml-stylesheet type="text/css" href="ss.css"?>
```

```
<test>
```

```
  <text1>My First XML with style!!!</text1>
```

```
  <text2>Let's have coffee after dinner!</text2>
```

```
  <text3>After that, shall we go for a long drive?</text3>
```

```
</test>
```

Code for the CSS file:

```
test  
{  
margin:10px;  
background-color:#ccfffc;  
font-family:verdana,sans-serif;  
}
```

```
text1  
{  
display:block;  
font-weight:bold;  
}
```

```
text2  
{  
display:block;  
color:#990000;  
font-size:small;  
font-style:italic;  
}
```

```
text3  
{  
display:block;  
color:#112299;  
font-size:large;  
}
```

You can enhance the css by adding more formatting features, and that shall reflect on the xml document. Similarly you can also try code using embedded or inline style sheets.

3.4.2 XML XSLT

XSLT (eXtensible Stylesheet language Transformations) is the most recommended style sheet language for the XML documents. Using XSLT,

XML document can be transformed as another XML document, or other web objects such as HTML, plain text etc. and looks more presentable as well as it can be converted into PDF, PostScript and PNG files. Original document is never changed, only a new document is created based on the contents of the existing ones.

An XSLT document is a valid XML document and consists of a number of elements/tags/attributes. A transformation can take place in one of the three locations:

- On the server
- On the client (for example, web browser)
- With a standalone program

Lets see how does the transformation takes place on the client, this time instead of css, we will link out xml document with the xsl file. The code for the xml file is as follows:

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="transform.xsl"?>

<test>
  <text1>My First XML with style using XSLT!!!</text1>
  <text2>Let's have coffee after dinner!</text2>
  <text3>After that, shall we go for a long drive?</text3>
</test>
```

Now create the xsl file, with the following code:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<html>
<head>
```



```
<title>XML XSL Example</title>
<style type="text/css">
body
{
margin:10px;
background-color:#ffffaa;
font-family:verdana,helvetica,sans-serif;
}

.test-text1
{
display:block;
font-weight:bold;
}

.test-text2
{
display:block;
color:#990000;
font-size:small;
font-style:italic;
}
.test-text3
{
display:block;
color:#009900;
font-size:small;
font-style:italic;
}
</style>
</head>
<body>
<h2>XML Transformation into HTML</h2>
  <xsl:apply-templates/>
</body>
</html>
</xsl:template>
```

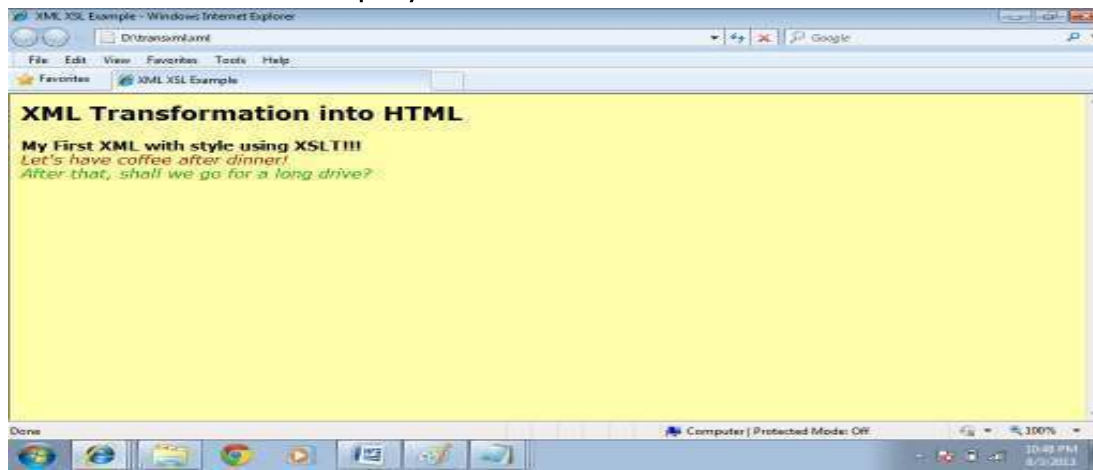
```

<xsl:template match="test">
  <span class="test-text1"><xsl:value-of select="text1"/></span>
  <span class="test-text2"><xsl:value-of select="text2"/></span>
  <span class="test-text3"><xsl:value-of select="text3"/></span>
</xsl:template>

</xsl:stylesheet>

```

The result shall be displayed like this:



Check Your Progress 3

- 1 Modify the Example given under section 3.4.2, as per given specifications:
 Back ground color – RED
 Text Color – White
 Font – Arial Narrow
 Add <p> tag with the content as - A very Interesting example!!!

3.5 Summary

In this unit, you have studied a basic introduction of XML and its document structure. You have seen how to validate and display any XML document and how are namespaces being used in these. Last but not the least, you have understood the concept better by undergoing through the interesting exercises.

3.6 Answers to Check Your Progress

Check Your Progress 1

1 If a document is syntactically correct, it can be called as well-formed XML documents. A well-formed document conforms to XML's basic rules of syntax such as:

- **Every open tag must be closed.**
- **The open tag must exactly match the closing tag as XML is case-sensitive.**
- **All elements must be embedded within a single root element.**
- **Child tags must be closed before parent tags.**

2. If a document is structurally correct then it can be called as valid XML documents. A valid document conforms to the predefined rules of a specific type of document: Please note that a valid XML document is implicitly well-formed, but well-formed document may not be valid

Check Your Progress 2

1. DTD and Schema

2. Limitations of DTDs:

- They have no explicit support for namespaces. All DTD declarations are global, so you can't define two different elements with the same name, even if they appear in different contexts.
- DTDs have support only for rudimentary datatypes.
- DTDs lack readability.
- DTDs use regular expression syntax to describe schema. This is less accessible to programmers than an element-based syntax may be.

3. Common built-in data types are as follows:

- xs:string
- xs:decimal
- xs:integer
- xs:Boolean
- xs:date
- xs:time

Check Your Progress 3

1. Try it yourself.

3.7 References

1. <http://www.w3schools.com>
2. <http://en.wikipedia.org>
3. <http://sitepoint.com>
4. <http://www.w3.org>

UNIT 4 DOCUMENT OBJECT MODEL

Structure

- 1.0 Introduction
- 1.1 Objectives
- 1.2 DOM Levels
 - 1.2.1 HTML DOM
 - 1.2.2 HTML DOM Nodes
 - 1.2.3 HTML DOM Node Tree
 - 1.2.4 Node Parents, Children and Siblings

Check Your Progress 1
- 1.3 Accessing Element in Java script
 - 1.3.1 Method
 - 1.3.2 Property
- 1.4 Traversing a DOM Tree
 - 1.4.1 Node List
 - 1.4.2 Node List Length
 - 1.4.3 Node Relationship
 - 1.4.4 Root Nodes
 - 1.4.5 childNodes and nodeValue
- 1.5 Modifying a DOM Tree
 - 1.5.2 Changing HTML Content
 - 1.5.2 Changing HTML Style
 - 1.5.3 Creating New Elements
 - 1.5.4 Removing Existing Elements
 - 1.5.5 Replacing Elements
- 1.6 DOM Collections and Styles

Check Your Progress 2
- 1.7 Events
 - 1.7.1 How Events Work
 - 1.7.2 Reacting to Events
 - 1.7.3 HTML Event Attributes
 - 1.7.4 Assign Events Using the HTML DOM

Check Your Progress 3
- 1.8 Dynamic Documents using JavaScript
- 1.9 About AJAX

Check Your Progress 4
- 1.10 Summary
- 1.11 Answers to Check Your Progress
- 1.12 Further Readings

1.0 INTRODUCTION

Every web browser window displays a HTML document. The window object that represents the window has a document property which refers to a document object. The document object has objects that represent the content of the document. The HTML documents can contain text, hyperlinks, images, forms etc. JavaScript code can directly access and manipulate the objects that represent the content of the document. A DOM is an API that defines how to access the objects that compose a document.

The DOM originated as a specification to allow JavaScript scripts and Java programs to be portable among Web browsers. "Dynamic HTML" was the immediate ancestor of the Document Object Model, and it was originally thought of largely in terms of browsers. However, when the DOM Working Group was formed at W3C, the vendors in other domains also joined it.

The Document Object Model (DOM) is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- build documents;
- navigate their structures;
- add, modify and delete elements and content;

1.2 DOM LEVELS

There are three DOM levels:

- The Level 0 DOM, supported from Netscape 2 onwards by all browsers.
- The two Intermediate DOMs, supported by Netscape 4 and Explorer 4 and 5.
Note that the use of these DOMs is not necessary any more
- The Level 1 DOM, or **W3C DOM**, supported by Mozilla and Explorer 5 and above.

"The W3C Document Object Model (DOM) is language and platform independent interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for any structured document
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

In this unit, we will be discussing about HTML DOM only.

1.2.1 HTML DOM

The HTML DOM is:

- A standard object model for HTML
- A standard programming interface for HTML
- A W3C standard

The HTML DOM defines the **objects** and **properties** of all HTML elements, and the **methods** to access them. In other words: *The HTML DOM is a standard for how to get, change, add, or delete HTML elements.*

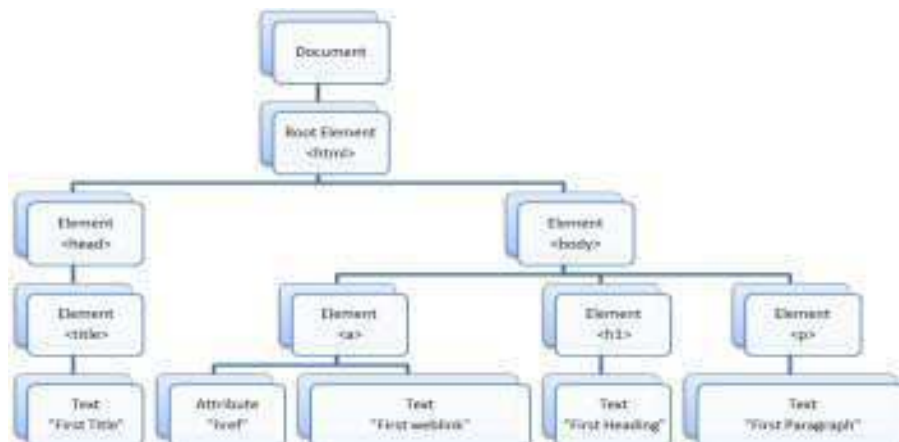
1.2.2 HTML DOM Nodes

According to the W3C HTML DOM standard, everything in an HTML document is a node:

- The entire document is a **document node**
- Every HTML element is an **element node**
- The text inside HTML elements are **text nodes**
- Every HTML attribute is an **attribute node**
- Comments are **comment nodes**

1.2.3 HTML DOM Node Tree

HTML documents are viewed as tree structures by the HTML DOM. The structure is called a **Node Tree**.



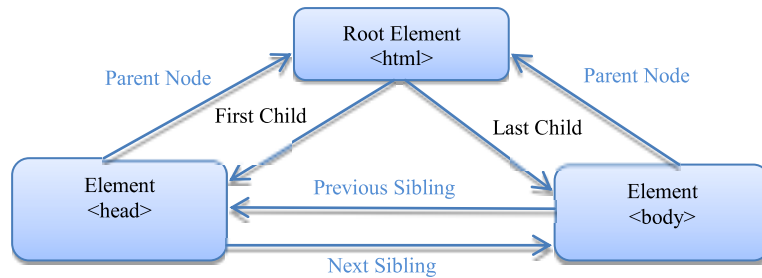
With the HTML DOM, all nodes in the tree can be accessed by JavaScript. All HTML elements (nodes) can be modified, and nodes can be created or deleted.

1.2.4 Node Parents, Children and Siblings

The nodes in the node tree have a hierarchical relationship to each other. The node directly above a node is the parent node of that node. The nodes at the same level and same parent are sibling nodes. The nodes that are one level directly below another node are the children of that node. So, to summarize

- In a node tree, the top node is called the root
- Every node has exactly one parent, except the root (which has no parent)
- A node can have any number of children
- Siblings are nodes with the same parent

The following image illustrates a part of the node tree and the relationship between the nodes:



Look at the following HTML fragment:

```

<html>
  <head>
    <title>Welcome to IGNOU</title>
  </head>
  <body>
    <h1>Courses Offered</h1>
    <p>Computer Science</p>
  </body>
</html>
  
```

From the HTML above:

- The <html> node has no parent node; it is the root node
- The parent node of the <head> and <body> nodes is the <html> node
- The parent node of the "Computer Science" text node is the <p> node
- The <html> node has two child nodes: <head> and <body>
- The <head> node has one child node: the <title> node
- The <title> node also has one child node: the text node "Welcome to IGNOU"
- The <h1> and <p> nodes are siblings and child nodes of <body>
- The <head> element is the first child of the <html> element
- The <body> element is the last child of the <html> element
- The <h1> element is the first child of the <body> element
- The <p> element is the last child of the <body> element

Check Your Progress 1

1. What is DOM and what are the objectives of DOM?

.....

.....

2. What are different levels of DOM?

.....

.....

3. Why was the name Document Object Model chosen?

.....

-
4. Describe the HTML DOM with an example?
-
-

1.3 ACCESSING ELEMENT IN JAVASCRIPT

All HTML elements are defined as objects, and the programming interface is the object methods and object properties.

1.3.1 Method

A **method** is an action, we can take e.g. add or modify an element. Refer to below table for methods used to access an HTML element in different ways:

Method	Purpose
<code>getElementById()</code>	Returns the element with the specified Id
Example: To get the element with id as “university” <code>document.getElementById("university");</code>	
<code>getElementsByTagName()</code>	Returns all elements with a specified tag name
Example 1: to get a list of all <p> elements in the document <code>document.getElementsByTagName("p");</code> Example 2: To get a list of all <p> elements that are descendants (children, grand-children, etc.) of the element with id="university" <code>document.getElementById("university").getElementsByTagName("p");</code>	
<code>getElementsByClassName()</code>	returns a list of all elements with a specified class name
Example : to get a list of all elements with class="university". <code>document.getElementsByClassName("university");</code>	

1.3.2 Property

A **property** is a value that you can get or set (like the name or content of a node).

Property	Description
<code>innerHTML</code>	<ul style="list-style-type: none"> To get the content of an element Useful for getting or replacing the content of HTML element

Following code gets the innerHTML from the <p> element with id="university":

```
< html>
< body>

< p id="university"> Welcome to IGNOU</p>

< script>
var txt=document.getElementById("university").innerHTML;
document.write(txt);
< /script>

< /body>
< /html>
```

nodeName

The nodeName property specifies the name of a node.

- nodeName is read-only
- nodeName of an element node is the same as the tag name
- nodeName of an attribute node is the attribute name
- nodeName of a text node is always #text
- nodeName of the document node is always #document

Note: nodeName always contains the uppercase tag name of an HTML element.

Example: To get the node name of the body element:

Include the following statement in the script
`document.body.nodeName;`

The result will be:
BODY

nodeValue

The nodeValue property specifies the value of a node.

- nodeValue for element nodes is undefined
- nodeValue for text nodes is the text itself
- nodeValue for attribute nodes is the attribute value

Following example retrieves the text node value of the <p id="university"> tag.

```
< html>
< body>
< p id="university"> Welcome to IGNOU </p>
< script type="text/javascript">
    x=document.getElementById("university");
    document.write(x.firstChild.nodeValue);
< /script>
< /body>
< /html>
```

nodeType

The nodeType property returns the type of node. nodeType is read only.

The most important node types are:

<u>Element type</u>	<u>NodeType</u>
Element	1
Attribute	2
Text	3
Comment	8
Document	9

Example: To convert all Text node (descendants excluded) data to uppercase:

```
if(n.nodeType == 3 /*Node.TEXT_NODE*/
  n.data = n.data.toUpperCase();
```

1.4 TRAVERSING A DOM TREE

With the HTML DOM, we can traverse the DOM tree using node relationships

1.4.1 Node List

The `getElementsByName()` method returns a **node list**. A node list is an array of nodes.

Example: Select all `<p>` element nodes in a document and access the second `<p>`

```
var x=document.getElementsByTagName("p");
y=x[1];      //as the index starts at 0.
```

1.4.2 Node List Length

The “length” property defines the number of nodes in a node-list. We can loop through a node-list by using the “length” property.

Example: Get all `<p>` element nodes. For each `<p>` element, output the value of its text node

```
var x=document.getElementsByTagName("p");
for (i=0;i<x.length;i++) // loop through <p> elements, no. of <p> nodes =x.length
{
    document.write(x[i].innerHTML); // output the value of text node
    document.write("<br />");
}
```

1.4.3 Node Relationship

The three node properties namely parentNode, firstChild and lastChild can be used to navigate in the document structure. Look at the following HTML fragment:

```
< html>
< body>

< p>Welcome to IGNOU</p>
< div>
< p>The courses offered by IGNOU</p>
< p>This example demonstrates node relationships.</p>
< /div>

< /body>
< /html>
```

- The first <p> element is the firstChild of the <body> element
- The <div> element is the lastChild of the <body> element
- The <body> element is the parentNode of the first <p> element and the <div> element

Example: This example demonstrates the use of “firstChild” property to access the text of an element. The output of this script will be Welcome to IGNOU, which is the text of the <p> element (the firstChild of the <body> element).

```
< html>
< body>
< p id="university">Welcome to IGNOU</p>
< script>
x=document.getElementById("university");
document.write(x.firstChild.nodeValue);
< /script>
< /body>
< /html>
```

1.4.4 Root Nodes

There are two special properties that allow access to the full document:

- document.documentElement - The full document
- document.body - The body of the document

Example

```

< html>
< body>

< p>Welcome to IGNOU</p>
< div>
< p>The Courses offered</p>
< p>This example demonstrates the <b>document.body</b> property.</p>
< /div>

< script>
alert(document.body.innerHTML);
< /script>

< /body>
< /html>

```

1.4.5 childNodes and nodeValue

childNodes and nodeValue properties can also be used to get the content of an element, in addition to the innerHTML property.

The following code gets the value of the <p> element with id="university":

Example

```

< html>
< body>

< p id="university"> Welcome to IGNOU </p>

< script>
var txt=document.getElementById("university").childNodes[0].nodeValue;
document.write(txt);
< /script>

< /body>
< /html>

```

In the example above, getElementById is a method, while childNodes and nodeValue are properties.

1.5 MODIFYING A DOM TREE

Modifying the HTML DOM can be many different things

- Changing HTML content
- Changing HTML style

- Creating New Elements
- Removing Existing Elements
- Replacing Elements

1.5.1 Changing HTML Content

We can change the content of an element by using the **innerHTML** property. It is the easiest way to do so.

Example: This example demonstrates how to change the HTML content of a <p> element. Here the content *Welcome to IGNOU* has been changed to *Innovative Learning!*

```
< html>
< body>
< p id="university">Welcome to IGNOU</p>
< script>
document.getElementById("university").innerHTML="Innovative Learning!";
< /script>
< /body>
< /html>
```

1.5.2 Changing HTML Style

With the HTML DOM we can access the style object of HTML elements. In the following example the HTML style of a paragraph has been changed.

```
< html>
< body>
< p id="university">Welcome to IGNOU</p>
< script>
document.getElementById("university").style.color="blue";
< /script>
< /body>
< /html>
```

1.5.3 Creating New Elements

To add a new element to the HTML DOM, we must create the element (element node) first, and then append it to an existing element.

Below HTML document contains a <div> element with two child nodes (two <p> elements):

```
< div id="div1">
< p id="p1">This is first paragraph.</p>
< p id="p2">This is second paragraph.</p>
< /div>
```

```

<script>
var para=document.createElement("p"); //creates a new <p> element
var node=document.createTextNode("This is new paragraph."); //create a text node
para.appendChild(node); //append text node to <p> element

var element=document.getElementById("div1"); // finds an existing element
element.appendChild(para); // appends new element to existing element
</script>

```

The appendChild() method in the above example, appended the new element as the last child of the parent.

If we want that the child should be inserted before a specific node and not as the last child, we can use the insertBefore() method:

```

<script>
var para=document.createElement("p");
var node=document.createTextNode("This is new paragraph.");
para.appendChild(node);

var element=document.getElementById("div1");
var child=document.getElementById("p1");
element.insertBefore(para,child);
</script>

```

1.5.4 Removing Existing Elements

To remove an HTML element, we must know the parent of the element. In this example below HTML document contains a <div> element with three child nodes (three<p> elements).

In this script the child node with id as p1 (*This is first paragraph.*) is removed from the parent i.e. <div> element.

```

< div id="div1">
< p id="p1">This is first paragraph.</p>
< p id="p2">This is second paragraph.</p>
< p id="p3">This is third paragraph.</p>
</div>
<script>
var parent=document.getElementById("div1"); //find the element with id "div1"
var child=document.getElementById("p1"); //find the <p> element with id "p1"
parent.removeChild(child); //remove the child from the parent
</script>

```

OR

Find the child that is to be removed and use its parentNode property to find the parent. The following code lines also achieve the same.

```

var child=document.getElementById("p1");
child.parentNode.removeChild(child);

```

1.5.5 Replacing Elements

To replace an element to the HTML DOM, we use the `replaceChild()` method.

In the script below HTML document contains a `<div>` element with three child nodes (three `<p>` elements). Here we replace the first child with id as `p1` (*This is first paragraph*) with the newly created `<p>` element *para* having a text node *node* as a child (*This is new paragraph.*).

```
< div id="div1">
< p id="p1">This is first paragraph.</p>
< p id="p2">This is second paragraph.</p>
< p id="p3">This is third paragraph.</p>

< /div>

<script>
var para=document.createElement("p");
var node=document.createTextNode("This is new paragraph.");
para.appendChild(node);

var parent=document.getElementById("div1");
var child=document.getElementById("p1");
parent.replaceChild(para,child); // p1 gets replaced by para.
< /script>
```

1.6 DOM COLLECTIONS & STYLES

The groups of related objects on a page are known as Collections in the Document Object Model. DOM collections are accessed as properties of DOM objects such as the document object or a DOM node. The document object has properties containing the images collection, links collection, forms collection and anchors collection. These collections contain all the elements of the corresponding type on the page. , the `length` property of the collection is used to find the number of elements in the collection. The variable `currentLink` (a DOM node representing an a element) has a specialized `href` property to refer to the link's href attribute. An easy access to all elements of a single type in a page is provided by DOM collections. This is useful for gathering elements into one place and for applying changes across an entire page. For example, the forms collection could be used to disable all form inputs after a submit button has been pressed to avoid multiple submissions while the next page loads.

An individual style statement is represented by the Style object. We can access the Style object from the document or from the elements on which that style is applied.

Syntax for using the Style object properties:

```
document.getElementById("id").style.property="value"
```


Example : The following example changes the background color of the <body> element using style object's *backgroundColor* property when a button is clicked

```
< input type="button" onclick="document.body.style.backgroundColor='yellow';"
value="Change background color" />
```

OR

```
< script>
function ChangeBackground()
{
    document.body.style.backgroundColor="yellow";
}
< /script>

< input type="button" onclick="ChangeBackground()"
value="Change background color" />
```

The Style object property has various categories like : Background, Border/Outline, Generated Content, Text, List, Positioning/Layout Printing, Margin/Padding , Table Misc etc.

For a detailed description on the style object property you can visit <http://www.w3schools.com/>

An element's style can be changed dynamically. Often such a change is made in response to user events. Such style changes can create many effects, including mouse over effects, interactive menus, and animations. In general, CSS properties are accessed in the format `node.style.styleproperty`. DOM Style Sheets allow us to step through the rules of each stylesheet, change the selectors, read and write styles, and add new rules. This allows us to create or change CSS that affects several elements at the same time, instead of just one element as with traditional DHTML. It also allows us to take advantage of CSS selectors to target the desired elements, and enter rules into the CSS cascade. DOM stylesheet does not provide an exact copy of what we put in our stylesheet. It produces what the browser sees, and what it interprets. Rules that the browser does not understand are not included. Styles that are not understood are ignored. Comments are not included.

The `document.styleSheets` collection has all stylesheets available. An easy way to check if a browser supports some amount of DOM Style Sheets is by checking for the existence of the `document.styleSheets` collection:

```
if( document.styleSheets ) {
    //DOM stylesheets are available
}
```

Each stylesheet has a number of properties that give details of the stylesheet, such as its URL (`href`), its title (`title`), its type (`type`, usually `'text/css'`), the media types it applies to (`media`), and if it is disabled or not (`disabled`). The `disabled` property can be set to true or false to disable or enable the stylesheet, but the other properties are all read-only. The properties will only be available if they actually apply to the stylesheet in question. For example, if a link element does not have the title attribute set, then its associated `StyleSheet` object will not have a title.

Check Your Progress 2

1. Write a function that converts all the Text node data of a node and its descendants to uppercase?
.....
.....
2. Write a function to find all the tables in a document.
.....
.....
3. Explain How to change HTML Attribute using HTML DOM?
.....
.....
4. Write a function to search for a specific table in a document and count its rows?
.....
.....
5. Write a function to return all name/value pairs of cookies in a document?
.....
.....
6. Write a script to toggle the visibility of a para element on the click of a button.
.....
.....

1.7 EVENTS

Events are generated by the browser when "things happen" to HTML elements. The HTML DOM allows you to execute code when an event occurs. For example clicking of the mouse, loading of the web page / image, moving the mouse over an element or submission of a HTML form etc. are HTML events.

1.7.1 How Events Work

When events happen to an HTML element in a web page, it checks to see if any event handlers are attached to it. If the answer is yes, it calls them in respective order, while sending along references and further information for each event that occurred. The event handlers then act upon the event. DOM elements can be nested inside each other. And somehow, the event handler of the parent works even if we click on its child. The reason for this is *event bubbling*. There are two types of event order:

- Event bubbling
- Event capturing

Event bubbling: It begins by checking the target of the event for any attached event handlers, and then bubbles up through each respective parent element until it reaches the HTML element. The main principle of bubbling states that after an event triggers on the deepest possible element, it then triggers on parents in nesting order.

Event capturing starts with the outer most element in the DOM and works inwards to the HTML element the event took place on and then out again. For example, a click in a web page would first check the HTML element for onclick event handlers, then the body element, and so on, until it reaches the target of the event.

We can choose whether to register an event handler in the capturing or in the bubbling phase. This is done through the `addEventListener()` method . If its last argument is `true` the event handler is set for the capturing phase, if it is `false` the event handler is set for the bubbling phase. Bubbling or capturing can be stopped by `event.cancelBubble=true` for (IE < 9) and `event.stopPropagation()` for other browsers. In all browsers, except (IE < 9), there are two stages of event processing .The event first goes down - that's called *capturing*, and then *bubbles* up. This behavior is standardized in W3C specification.

1.7.2 Reacting to Events

A JavaScript can be executed when an event occurs.

Example 1: Content of the `<h1>` element is changed when a user clicks on it

```
<html>
< body>
< h1 onclick="this.innerHTML='IGNOU'">Click on this text!</h1>
< /body>
< /html>
```

Example 2: In the following example the javascript is executed on `onblur` event. When the user leaves an input field the `onblur` event is triggered and the handler for this event (`changeToUpper()`) gets executed. This function changes the text of the input field to upper case.

```
<html>
<head>
<script>
function changeToUpper()
{
var x=document.getElementById("fname");
x.value=x.value.toUpperCase();
}
</script>
</head>
<body>
Enter your name: <input type="text" id="fname" onblur=" changeToUpper()">
<p>When you leave the input field, the input text will change to upper case.</p>
</body>
</html>
```

Example 3: In the following example we change the text of the `<p>` element from “Let us learn Event Handling” to “I have learnt Event Handling” when a button is clicked.

```
< html>
< body>
< p id="p1">Let us learn Event Handling.</p>
< script>
function ChangeText() {
document.getElementById("p1").innerHTML="I have learnt Event Handling.";
}
< /script>
< input type="button" onclick="ChangeText()" value="Change text">
< /body></html>
```

1.7.3 HTML Event Attributes

To assign events to HTML events we can use event attributes.

Example: Assigning an “onclick” event to a button element

```
<button onclick="displayMonth()">Try it</button>
```

In the example above, a function named *displayMonth* will be executed when the button is clicked.

1.7.4 Assign Events using the HTML DOM

HTML DOM allows us to assign events to HTML elements using JavaScript.

Example: Assigning an “onclick” event to a button element

```
<script>
document.getElementById("myBtn").onclick=function(){displayMonth()};
</script>
```

In the example above, a function named *displayMonth* is assigned to an HTML element with the id=myBtn".

Event	Description
onload onunload	<ul style="list-style-type: none"> These events are triggered when the user enter or leaves the page onload event can be used check browser type/ version and load the proper version of web page onload and onunload can be used to deal with cookies <p>Example: <body onload="checkCookies()"</p>
onchange	<ul style="list-style-type: none"> often used in combination with validation of input fields <p>Example: <input type="text" id="fname" onchange="upperCase()"></p>
onmouseover onmouseout	<ul style="list-style-type: none"> These events can be used to trigger a function when the user mouse over, or out of, an HTML element <p>Example: <html> <body> <h1 onmouseover="style.color='red'" onmouseout="style.color='blue'"> Mouse over this text</h1> </body> </html></p>
onmousedown onmouseup onclick	<ul style="list-style-type: none"> onmousedown, onmouseup, and onclick events are all parts of a mouse-click First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is

	triggered, finally, when the mouse-click is completed, the onclick event is triggered.
--	--

For more information on the events you can visit <http://www.w3schools.com/>

Check Your Progress 3

- Write a program to change the HTML element using events?
.....
.....
- What are the functionalities performed by onload() and onUpload()?
.....
.....
- Execute a javascript before the browser closes the document.
.....
.....
- Write a script to alert the keycode of the key pressed.
.....
.....
- Write a function to execute a script that changes the color of an image when moving the mouse over / out of the image?
.....
.....
- Write a function to return the type of event that occurred?
.....
.....

1.8 DYNAMIC DOCUMENTS USING JAVASCRIPT

Dynamic Document Creation is the creation of a Web document from within the JavaScript. It may be created while an HTML document is being displayed, and may be influenced by features of the current Web page. As a technique, it is very useful for displaying information from Web pages and creating new HTML documents.

Let us create dynamic documents with **document.write()**. This method can

- insert text into the current web page
- create an entirely new document

Let us look at this example that uses write to display today's date to an otherwise static HTML document.

```
<script type="text/javascript">
var today = new Date()
var mon = today.getMonth() + 1
document.write(
"<html><head><title>Dynamic Document</title></head>\n"
```

```

+ "<body bgcolor=yellow>\n <h1 align=center>Today is "
+ mon + "/" + today.getDate() + "/" + today.getYear()
+ "</h1>\n</body></html>")
</script>

```

We can also use the write method in conjunction with the open() and close() methods of the Document object, to create entirely new documents in other windows or frames. The function given below opens a pop up window to display the date the document is accessed on. Invoke this function from an event handler.

```

Function recent() {
    var w = window.open(); // Create a new window with no content
    var d = w.document(); // Get its Document object
    var today = new Date();
    d.open(); //Start a new document
    d.write("<p> Document recently accessed on: " + today.toString());
    d.close(); // Close the document
}

```

1.9 AJAX

AJAX stands for **Asynchronous JavaScript and XML**. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS and Java Script. It uses XHTML for content and CSS for presentation, as well as the Document Object Model and JavaScript for dynamic content display.

In standard Web applications, the interaction between the customer and the server is synchronous. This means that one has to happen after the other. If a customer clicks a link, the request is sent to the server, which then sends the results back.

With Ajax, the JavaScript that is loaded when the page loads handles most of the basic tasks such as data validation and manipulation, as well as display rendering the Ajax engine handles without a trip to the server. At the same time that it is making display changes for the customer, it is sending data back and forth to the server. But the data transfer is not dependent upon actions of the customer.

The Ajax engine works within the Web browser (through JavaScript and the DOM) to render the Web application and handle any requests that the customer might have of the Web server. Because the Ajax engine is handling the requests, it can hold most information in the engine itself, while allowing the interaction with the application and the customer to happen **asynchronously** and independently of any interaction with the server. The key feature of Ajax application is that it uses scripted HTTP to communicate with a web server without causing pages to reload. Since the amount of data exchanged is often small and the browser does not have to parse and render a document. As a result, the response time is greatly improved and this results in making web applications feel more like desktop applications.

Here is the list of famous web applications which are using AJAX

Google Maps (<http://maps.google.com/>)

A user can drag the entire map by using the mouse instead of clicking on a button or something

Google Suggest (<http://www.google.com/>)

As you type, Google will offer suggestions. Use the arrow keys to navigate the results

Gmail (<http://gmail.com/>)

Gmail is a new kind of webmail, built on the idea that email can be more intuitive, efficient and useful.

Check Your Progress 4

1. What is AJAX? List out the differences between AJAX and JavaScript.

.....

2. What are the advantages of AJAX?

.....

3. Name the browsers that support AJAX?

.....

4. What are the limitations of Ajax?

.....

1.10 SUMMARY

In this unit we have learnt how to use the HTML DOM to make our web site more dynamic and interactive. We have also learned how to manipulate HTML elements in response of different scenarios and created dynamic web pages by using scripts on the client (in the browser). We got to know about AJAX which is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS and Java Script.

1.11 ANSWERS TO CHECK YOUR PROGRESS

1.11.1 Check your progress 1

1. The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content.

One important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and applications.

2. There are three DOM levels:

- The Level 0 DOM, supported from Netscape 2 onwards by all browsers.

- The two Intermediate DOMs, supported by Netscape 4 and Explorer 4 and 5. Note that the use of these DOMs is not necessary any more
- The Level 1 DOM, or W3C DOM, supported by Mozilla and Explorer 5

3. The name "Document Object Model" was chosen because it is an "object model" is used in the traditional object oriented design sense: documents are modeled using objects, and the model encompasses not only the structure of a document, but also the behavior of a document and the objects of which it is composed.

4. The HTML DOM defines a standard way for accessing and manipulating HTML documents. The DOM presents an HTML document as a tree-structure.

1.11.2 Check your progress 2

1.

```
function uppercase(n) {  
    if (n.nodeType == 3) /* Text Node */  
        n.data = n.data.toUpperCase();  
    else{  
        var children = n.childNodes;  
        /* loop through children, recursively call  
        for ( var i=0; i<children.length; i++)  
            uppercase(children[i] );  
    }  
}
```
2.

```
var tables = document.getElementsByTagName( "table" );  
alert( " This document contains ( + tables.length + "tables" );
```

3. Example to change HTML Attribute by using HTML DOM

```
<html>  
<body>  
  
<script type=" javascript">  
document.getElementById("image").src = "newclip.jpg"  
</script>  
</body>  
</html>
```

In the above example we load an image on HTML document by using id="image". Using DOM we get the element with id="image". This script will change the src attribute from oldclip.gif to newclip.jpg

4.

```
var table secondtable = document.getElementById ( " COURSES" );  
var rows = secondtable.getElementsByTagName( " tr " );  
var totalrows = rows.length;
```
5.

```
<html>  
<body>
```


Cookies associated with this document:

```
<script>
    document.write(document.cookie);
</script>
</body>
</html>
```

```
6.<html>
  <body>
<p id="p1"> Welcome to IGNOU Welcome to IGNOU Welcome to IGNOU.
Welcome to IGNOU </p>
<input type="button" value="Hide text"
onclick="document.getElementById('p1').style.visibility='hidden'" />
<input type="button" value="Show text"
onclick="document.getElementById('p1').style.visibility='visible'" />
</body>
</html>
```

1.11.3 Check your progress 3

1. Events are usually generated by the browser, when an event on some object takes place like user clicks an element. Event handlers are used to handle the events and execute the code . The example is shown below that is used to change the HTML element values:

```
<html>
<body>
<input type="button" onclick="document.body.bgColor='green';"
value="Change background color" />
</body>
</html>
```

2. onload() and onUpload() are two functions that get activated or the event gets triggered when the user enters or leaves the page. The onload() event is used to verify and check the user's visit according to the browser's type and it loads a version of the event. It provides the web page information that allows easy access to the website and consists of the information regarding the event that is triggered. Onload() and onUpload() events use the cookies to hold down the values given by the users when it enters or leaves the page.

```
3. <html>
<head>
<script>
function beforeclose()
{
alert("Welcome to IGNOU!");
}
</script>
</head>
```

```
<body onunload="beforeclose ()">
```

```
<h1>Welcome to IGNOU Home Page</h1>
<p>Press F5 to reload the page OR close this window.</p>
```

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

```
4. <html>
<head>
<script>
function codeKey(event)
{
  alert(event.keyCode);
}

</script>
</head>
```

```
<body onkeyup=" codeKey (event)">
<p>Press a key on your keyboard. An alert box will show the keycode of the key
pressed.</p>
</body>

</html>
```

```
5. <html>
<head>
<script>
function bgChange(bg)
{
  document.body.style.background=bg;
}
</script>
</head>
<body>
<b>Mouse over the squares and the background color will change!</b>
<table width="200" height="100">
<tr>
<td onmouseover="bgChange('red')"
  onmouseout="bgChange('transparent')"
  bgcolor="red">
</td>
<td onmouseover="bgChange('blue')"
  onmouseout="bgChange('transparent')"
  bgcolor="blue">
</td>
<td onmouseover="bgChange('green')"
  onmouseout="bgChange('transparent')"
  bgcolor="green">
</td> </tr>
</table>
</body>
</html>
```

```
6. function getTypeOfEvt(event)
{
  alert(event.type);
}
</script>
```

```

</head>

<body onmousedown=" getTypOfEvt (event)">

<p>Click in the document.
The type of event triggered will be shown in an alert box .</p>

</body>
</html>

```

1.11.4 Check your progress 4

1. Ajax is abbreviated as Asynchronous Javascript and XML. It is new technique used to create better, faster and more interactive web systems or applications. Ajax uses asynchronous data transfer between the Browser and the web server. Here on sending request to the server, one needn't wait for the response. Other operations on the page can be carried out Hence, Asynchronous. On the other hand, Java script sends an HTTPRequest to the server and waits for the XML response e.g. populating State field. Using JavaScript we need to use the "Onchnage" event where as using ajax, the request is just sent to populate the state list. Other operations can be carried out on the page. Ajax is a part of Java Script programming. Java Script is used to manage and control a web page once downloaded. Ajax does not need to wait for the whole page to download.

Use of Ajax can reduce connections to the server since the script has to be requested once.

2. Following are the advantages of Ajax:

- Bandwidth utilization – It saves memory when the data is fetched from the same page.
- More interactive
- Speeder retrieval of data

3. Following browsers support AJAX:

- Internet Explorer 5.0 and above
- Opera 7.6 and above
- Netscape 7.1 and above
- Safari 1.2 and above

4. Limitations of AJAX:

- Back functionality cannot work because the dynamic pages don't register themselves to the browsers history engine
- The page cannot be bookmarked if implemented using Ajax.
- If java script is disabled, Ajax will not work.
- Because different components of the pages are loaded at different times it may create confusion for the user.

1.12 FURTHER READINGS

http://www.w3.org/wiki/Handling_events_with_JavaScript

<http://www.w3schools.com/>

<http://simplehtmldom.sourceforge.net/>

<http://stackoverflow.com/>

<https://developer.mozilla.org/en/docs/DOM>

Deitel, H. M., & J., D. a. (2008). *Internet & World Wide Web How to Program*, 4/e. Pearson Education

Sebesta, R. W. (2011). *Programming the World Wide Web*, 6/E. Addison-Wesley / Prentice Hall

In the previous Block you have gone through the concepts related to web programming scripting languages such as HTML5, XML, JavaScript etc. One common point about these languages is that these languages are interpreted and executed by the browser which results in display of web pages. When you access a web page the client side script files are transferred from the web site which is hosted on a web server to the client side through hypertext transfer protocol. But are these files stored as similar files at the web server or they are generated with the help of a programming language? This block attempts to answer this question. Simply speaking, if the web site displays standard pages to every client, the web server may just store these pages may be as standard HTML tags. Such sites are sometimes referred to as static web sites. However, in practice static web sites has limited uses and Web 2.0 web sites are more dynamic and interactive. This means that the pages that may be created for a client may change as per the need of the client, thus, you need some dynamism at the web server level too. The languages that are used to create client pages at the server, based on clients requirements may be categorized as the Server Side Scripting Languages. Some of the common server side scripting languages are PHP, JSP, SERVLET, ASP.NET, PYTHON and many more. In this Block, we have used JSP as the server side scripting language. In this Block we will be discussing about the server side scripting. This Block consists of four Units: Unit 1 discusses the basic concepts relating to server side scripting. It explains various enterprise level architectures, HTTP methods and the concept of web container.

Unit 2 explains the basic directives and elements of JSP. It also explains the concepts of expressions, and some basic JSP objects.

Unit 3 focusses on exception handling using JSP and use of cookies and sessions in the content of JSP. It also introduces the concept of managing emails using JSP.

Unit 4 presents an example of JSP that uses JDBC and database drivers. It also explains how you can connect to databases and develop applications involving databases.

Unit Continues from Page 7

UNIT 1: THE SERVER SIDE SCRIPTING

Structure	Page Nos.
1.0 Introduction	5
1.1 Objectives	5
1.2 Server side scripting and its need	
1.3 Two-Tier, Three-Tier, N-Tier and Enterprise Architecture	
1.4 Various Languages/ Technologies for server scripting	
1.5 HTTP Methods (such as GET, POST, HEAD, and so on)	
Purpose	
Technical characteristics	
Method selection	
1.6 Use of request and response primitives	
1.7 Web container – Tomcat	
1.8 Summary	
1.9 Answers to Check Your Progress	
1.10 Further Readings	

1.0 INTRODUCTION

In the last Block you have gone through the concepts of client side scripting. A client side script is executed by a browser and displayed as per the stated format. Such code however, in general is static in nature. In order to create dynamic web sites you need to use server side scripting.

This unit explains the concepts of server side scripting. It first defines the need of server side scripting. It explains various types of client server architectural model that are being used to develop flexible dynamic applications. This Unit also describes the role of various HTTP methods which help in transfer of data from the client side to the server side. It also explains the concept of request and response primitives that may be needed in a client server system. Finally the unit explains the concept of a web container with the help of an example.

This Unit thus, provides the basic information about the server side scripting. The remaining three Units of this Block are devoted to one server side scripting language JSP. Please remember that this block only introduces you to the server side scripting. This is one area where tools and technologies are changing at a very rapid rate, therefore, you must keep learning about web programming through suggested further readings on the WWW.

1.1 OBJECTIVES

After going through this Unit you should be able to:

- Define the need and the purpose of server side scripting
- Explain the purpose of tiers of architecture
- Identify several languages for server side scripting
- Work with some of the HTTP methods.

- define request and response primitives and their uses.
- List the features of a web container

1.2 SERVER SIDE SCRIPTING AND ITS NEED

Internet and specially World Wide Web is a major phenomenon of the last two decades. It was first proposed in a paper written by Sir Tim Berner-Lee in the year 1989. The WWW proposal by him was aimed at better communication and was based on the concept of hypertext – a text that can be linked (called hyperlink) to other text document. What should the basic requirements of WWW?

You must have visited WWW and would have visited many web sites. So just try to identify what are the very basic requirements that might be needed for WWW. The first and the foremost is the web address which is a unique address of a resource at a website on WWW. This resource address is also called the Uniform Resource Locator (URL). The second in this context would be a set of rules that allows transfer of the resources (mostly files) from a website to the user. This set of rules was documented as HyperText Transfer Protocol (HTTP). Thirdly, you must agree on some common but simple computer language that allows consistent publishing of the contents (may be in any spoken language). This common computer language is HyperText Markup Language (HTML) which uses standard tags. Now, the question is how does a user reach to a website and where does the website reside?

You as a user of WWW you need an Internet connection which should be running the HTTP and TCP/IP protocols (recollect HTTP is the application layer protocol of TCP/IP). You on your computer use a browser, type a website address which happens to be a URL of a starting page(may be the default index page index.html) of the website. The website responds by sending this page over the Internet and browser interprets and displays this page for you. In the whole process, the browser at your computer acts as a *Client*. The website obviously has to take the responsibility to make sure that client gets the requested page, thus, serves the client. Thus, websites are hosted on a *Server*. Since this server is of WWW commonly called web, the server is called the Web Server. In effect, the basic architecture is the *Client-Server* architecture. Figure 1 shows this architecture. Please note in the figure that there may be many clients for a website.

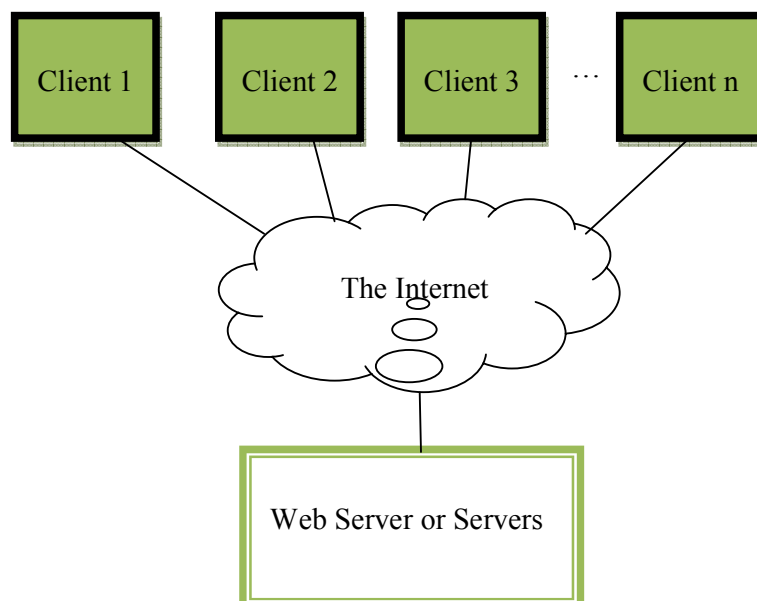


Figure 1: The Client-Server Model

The implication of this architecture is that you can clearly identify three basic components viz. the Client, the Internet cloud and the Server. The technologies of all these three components can be enhanced separately as long as they follow the three basic tenets, those are, the resources should be named using a standard (URL), a standard communication protocol is used (HTTP) and the contents are transferred to the client using a standard tagging language (HTML). Even all these standards over the period of time can be enhanced, but for this purposes newer client and server software needs to be created. You can relate these facts to development of newer secure protocols like HTTPS, creation and enhancement of client side scripting languages and CSS, and server side scripting technologies. The tremendous growth of WWW in terms of technologies can, thus, be attributed to the simple, flexible architecture and basic technologies.

Today WWW consists of dynamic and interactive websites. A website can be made dynamic using both the client-side and server-side scripting. On the client side many technologies, like HTML5, CSS3, JavaScript, JSON, AJAX and many others, have been developed. Some of these technologies have been studied by you in Block 1. You must keep studying about these technologies form the WWW and keep updating yourself. The client side scripting can be used to perform activities at the client site that does not require information from the server. Some such activities may include creating effects such as animations, simple games, changing display, and even checking of information in forms that does not require checking at the server etc. The client-side scripting not only helps in creating some visual effect on the client but also reduces unnecessary communication between the client and server.

However, it is the Server-Side Scripting that has lead to major advances in website design. The basic tenet of server side scripting is that to send a HTML file to a client (browser), you need not store these files on the server as HTML files. Rather, information can be collected from many sources including databases and HTML document can be created and sent to the browser. This makes the websites very dynamic as they can collect information and display formats as per the choices of the clients. The input to server side script is the input obtained from the client. The server-side script is executed at server using the resources of the servers. The final output is created for a client and sent to it for display on the browser. Please note that server-side scripting may generate different output for different browsers, if required. It makes the process of creating web pages simple and programmable. Server side scripting had helped in creating big e-commerce based portals simpler to create and deploy. Can you think of some basic usage of server-side scripting?

Some basic uses of server-side scripting include applications that require users to login to system using a password. However, for such applications you also face problems due to stateless nature of HTTP protocol. What does this stateless protocol means?

It simply means that your current request to a server cannot be related to previous request as in HTTP server is not required to retain information about your previous requests. Therefore, if you are performing a transaction that requires multiple requests, you need to make sure that the information of previous requests is duly recorded. Thus, you need to make special provisions such as creating cookies or sessions that help in performing such tasks. Unit 3 of this Block explains these concepts in more details. However, you should know that the concept of stateless protocol is extremely useful for Internet as storing states take space and time, and given the size of internet and its users it would have made Internet very slow.

Server-side scripting is also very useful in extracting information from the user, may be with the help of a form or otherwise, and processing the information for some useful purpose. Suppose, you are interested in opening a web based email account, then you may fill up an account opening request form of an e-mail service provider

like gmail, yahoo mail, rediffmail, etc. When you are filling up the form and commit some catchable mistake, client side JavaScript caches the error. However, once you submit the form and still an error is encountered, like duplicate user name, it can only be caught by the server-side script.

Server-side scripting is also useful when you want to use a database as a backend to a web based application. Databases are normally stored on a database server which can be directly connected to the web server. Such kind of application has been discussed in the Unit 4 of this Block.

Finally, just some basic advice on when to use client-side scripting and when to use server-side scripting? The answer is very simple – if a user interaction element of a page can be processed without the need of any information from the web server, then you use client-side scripts. It reduces the load on network traffic, but the client hardware should be sufficiently fast to execute the client side script.

form of email service provide for obtaining the email access developing an e-commerce application, then your selection of various products can be stored with the help of login in a session using forms. Once you want to create the final bill, all the information you have selected in various. However, client-side scripts are dependent on browsers and are less secure. For secure applications, you should use server-side scripting.

1.3 WEB APPLICATION ARCHITECTURES

In the previous section, you have gone through the concept of client-server model of computing. The model discussed in Figure 1, defines two simple portion where computing is performed. The first one is the client, which performs the tasks such as displaying a web page of an HTML document using CSS, running JavaScript, performing data conversion on the client side, dealing with client side Graphical User Interface display, communicating the data of the clients over the Internet to the server. While on the server side data may be stored, updated, or retrieved from database, the data may also be processed and a response may be sent to the client. Thus, the model clearly identifies a Client and a Server.

The objective of such a model is to clearly the division of work providing a flexible and scalable model of computing. This model provides flexibility in the sense that a new client can be added to whole system without much effort. The client only needs to know what server to access through the network and how. Now, consider a client-server environment in which you initially had only single digit number of clients, therefore, one single server was able to serve all the clients. However, the application became popular and hundreds of clients joined. In order to serve these clients you may need to deploy more servers. The important point here is that you can scale the number of servers based on the requirements. Thus, client-server model is a scalable model of computing.

1.3.1 N-Tier Architecture

In order to develop client-server based, you need an application development architecture. One such architecture that can be considered is 2-tier architecture. Obviously, in the 2-tier architecture, the first tier belonging to the client, which may be called a client tier, client tier or frontend, and the second tier may be called the server tier or backend. A developer needs to design the basic interactivity for the application, rules of data access and business rules in the client tier. In present day applications interactivity is provided through GUIs. Thus, all the information pages, forms, etc. designed by you are part of the client tier. Thus, web pages of a web application accessed by you using your computer system and browser are part of the

client tier of the web application. The following is an illustration of use of 2-tier architecture.

You (client Computer) using a Browser	Web Server running a Web Application	
	Client Tier/Layer	Server Tier/Layer
Requests index page of a web site by typing the address of the web site		Receives the request and processes it to collect relevant files.
	The presentation tier will use server tier files to create the web page HTML+CSS+JavaScript for communication to client computer.	
The index page is displayed on the browser. Suppose the web site requests you to login. You will enter the user Id and password and press Submit.		The server will receive the data. It will connect to database to determine if the user ID and Password are correct. Suppose it is not then it will create the failure message.
	The failure message will be converted to the required web page in HTML format along with CSS and the page will be communicated to the client computer. Please note this layer may also check the number of login attempts.	
The message page will be displayed. You can try again and the process continues...		

Figure 2: An Illustration of 2-Tier Architecture

As a client, you use the client tier to submit your request for a web page or some data. The client tier also checks, if you have permission to access the requested data. In case, user has the access permission, the requested information details are passed to the server, where the server-tier programs services the request sent by you (as a client). Once the requested data is available, then it is sent back to the client. Please note that display of data on your client computer is processed by the client tier of the web application.

The 2-tier client-server applications are useful for distribution of work, however, they require complex client implementations and number of communications between the client and servers. With the popularity of Internet new web application architecture was developed called n-Tier architecture. One of the most common n-tier architecture is 3-tier architecture. Figure 3 shows 3-tier architecture.

The layers and their basic functions are:

Presentation Layer/Tier: The presentation tier of the three tier architecture interfaces with the client. The presentation tier is responsible for displaying the information to the client as well as extracting information from the client. Technically, for a web based application the presentation tier resides on the web server on request the presentation layer displays interactive web pages in the browser of client computers.

Application Logic Layer/ Tier: This is also called the middle ware. This layer is primarily responsible to provide business rules, sharable components of a web application, access control etc. This layer shields the data layer from direct use of the clients. This layer provides an interface between the presentation and data layer.

Data Layer/Tier: The data for a web application may be hosted on a database management system or a file system. The data layer controls the integrity of data residing on some data storage system. The application logic sends queries to data layer which sends back the query results to the application logic layer.

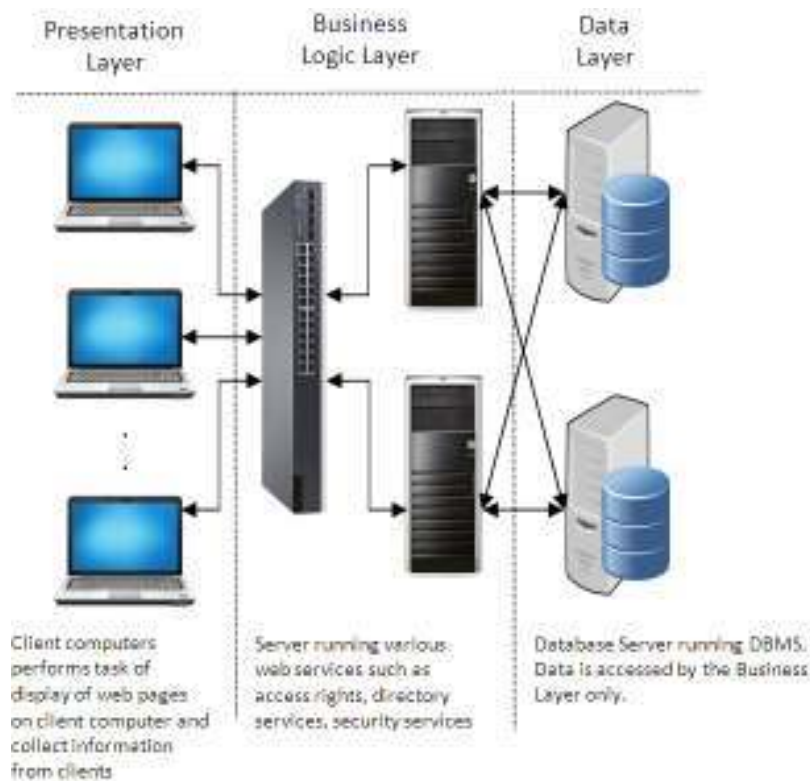


Figure 3: An n-Tier Architecture

Figure 4 shows an illustration of 3-tier Architecture.

Layer / Executed at	Example
Presentation Layer/client computer and web server	User logs in and on Successful login requests the list of students for BCA programme using a GUI based interface created by the web server.
Business Layer/Web Server	On user request the business layer verifies the access constraints of the user, suppose he is only authorized to see a group of students of BCA, then the query to database is suitably modified.
Data Layer/Web Server and Database	The query is executed using a connection

Server	with the database, and the results are returned to the web server.
Business Layer/Web Server	The web server checks if the returned data set is empty or not. If not passes the data set to the Presentation Layer, else returns an error message.
Presentation Layer/ web server and client computer	The presentation layer converts the data into a HTML table or error message to an error page. This page then is sent to client computer and displayed in the browser.

Figure 4: An illustration of 3-Tier Architecture

An n-tier architecture in addition to the above three layers may have many more application layers such as client presentation layer, entity class layer, persistence layer etc. The more are the number of layers the more complex the system is, however, more layers may bring better application flexibility.

In general, n-tier architectures results in more scalable, more secure (database access is hidden) and better integrity based applications. However, they are more complex in nature.

1.3.2 MVC Architecture

In the previous sub-section, you have gone through the concept of n-tier architecture. In this section, we discuss an important architecture that is used by Java for web application development. This architecture is known as Model-View-Controller (MVC). This architecture can also be used for the development of a web application. Figure 6 shows the component of web applications as per this architecture.

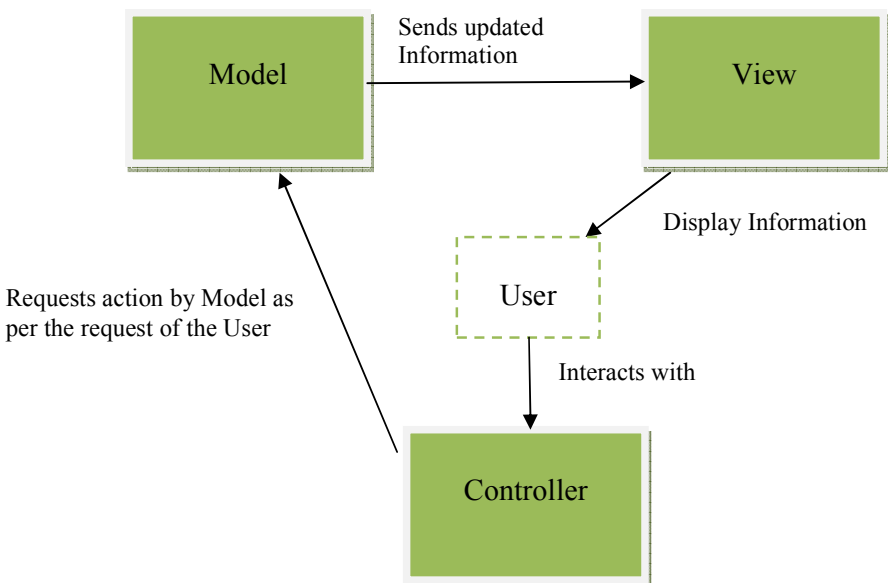


Figure 6: MVC Architecture

The MVC architecture has three functional component of a web application that communicates through a user as well as among them. The following are the basic components of MVC architecture:

Model: In the context of MVC a Model defines the data model and its access controls. The main role of this component is to represent data and perform updates on it as per the defined rules. The main responsibility of this component is to accept user requests and the data entered by the user and perform the necessary data related function.

View: The function of the view will be to accept the data from the Model and convert it to form that can be seen by a user in a user friendly way. Thus, view is responsible for displaying web pages for the user. Any change in the model should also change the view. This can be achieved using two processes -

Push processes allows views to register with a model and change in the data of the model result in pushing the current data to view.

Pull process requires the view to get data current data from the model when needed.

Controller: A user may be allowed to interact with the web pages created by the View component. This interaction may be in the form of selection of options of Menu, pull down lists, check boxes etc., filling up data in text boxes etc. It is the controller which accepts this data from the user and initiates suitable actions that should be carried out by the models.

Let us explain the MVC with the help of an illustration as shown in figure 7.

Model	You decided to create a website for storing information of registered students for various courses on an e-learning website. The model may be the student registration database.
View	As a first time user, you want to register into the website, one of the view would be the registration page. Another view may be the list of courses on offer. Please note that in the view pages the course list will be created from the model. In case a new course is added, this list needs to be updated. How? Not manually, but by a program. You will learn about this in the subsequent Units.
Controller	You may register to a course by submitting the online registration form, the data of this form will be accepted by the controller and sent to model for necessary actions.

Figure 7: An Illustration of MVC

Check Your Progress 1

- 1) What is the need of Server side scripting? How is it different to client side Scripting?

.....

.....

.....

2) What are the various components of 3-Tier architecture? Why is it needed?

.....

.....

.....

.....

3) Compare 3-Tier Architecture with MVC?

.....

.....

.....

.....

1.4 TOOLS FOR SERVER SIDE SCRIPTING

In this section we will explain about the basic features needed for the server side scripting languages. As discussed in section 1.2, purpose of server-side scripting is program the response of a server to the client side requests. Therefore, the input to server side scripting would be the interactive data input sent by the client using a web browser, and the output is the displayable web page which are displayed at the client's browser.

The question that you need an answer is: What should be the elements of a server side programming language? In order to answer this question, you need to first look at the functions that may be performed by the server side scripting. These are:

- Some mechanism for accessing the data that has been transferred from the client.
- Creation of dynamic web pages in which server side scripting may change the contents of a specific portion of a webpage
- Processing of form data obtained from the client side, this data sometimes may be used to store, update, and retrieve information from the database. All these activities are performed by the server-side scripting
- Handling errors that may occur due to several reasons, such as database access errors, data related errors, or other events.
- Enforcing the data security and integrity and handling constraints.
- Handling creation of sessions between client and server despite HTTP is stateless protocol.
- Creating output pages in a HTML or other client side language form.

Thus, a server side scripting language support following constructs:

- The basic programming constructs like variables, data types, assignment statements, if statement, looping statements, arrays, etc.

- Most of the web programming languages are object oriented programming languages, therefore, they use classes, objects, inheritance hierarchy, containers, collections of classes, error handling etc.
- To communicate with the clients, they support some predefined methods, primitives and protocols. This ensures that data entered by a user is duly sent to a program that handles it. Most of the server side programming languages call these data items as parameters and have various ways to refer to these parameters.
- A good website requires access to a database system to store, update and retrieve data based on user's request. A database is maintained under the control of a DBMS. You are required to use a database driver that makes possible the exchange between the web server and the database server. These drivers must be connected through some connection protocols (some of these protocols are Open DataBase Connectivity (ODBC) and Java DataBase Connectivity (JDBC)). Thus, a server side script must have classes for driver specification and connection establishment. In addition, server-side scripting support constructs to create and execute SQL commands through these connections and obtain the resulting data tables. But how to access individual data records and attributes? Thus, every server-side scripting must support tools to access data records or rows and attributes or columns.
- Since HTTP is a stateless protocol, therefore, you need to establish sessions or create cookies (These will be discussed in Unit 3 of this Block). Thus, server side scripting must have classes/mechanisms to deal with sessions and cookies.
- Server side script produces output for the browser, therefore, such classes or commands must exist in such languages.

Thus, server-side scripting requires extensive features on part of its tools. Some of the most popular server-side scripting languages include – ASP.NET, PHP, JSP and SERVLETS, Perl and many more. There are a number of tools that integrate complete web development environment including web servers. Some such IDEs are Eclipse, Netbeans, and Visual Studio Express etc. A detailed discussion on these tools is beyond the scope of this Unit. You will be learning more about JSP in this Block. However, you may learn and use other languages also as many of the concepts are of similar nature.

1.5 HTTP METHODS

In the previous sections, you have been explained about the client-server architectures and the features of the language required for web application development. In these sections, you have been told that data from the client is passed to the server using a protocol. In this section, we define some of the basic methods that make this communication possible.

HyperText Transfer Protocol (HTTP) is the protocol that is used for WWW. HTTP is an application level protocol. What is an application level protocol? Please find out answer to this question from BCS041: Fundamentals of Computer Networks. Present version that is in use is HTTP/1.1

HTTP uses a request-response method of communication. As a client, you type in the URL of a website, for example, <http://www.ignou.ac.in>, you are initiating a request from the web server

1.6 REQUEST AND RESPONSE PRIMITIVES

below:

1.7 WEB CONTAINERS

The Chi-square Distribution is defined in the BCS-040/Block 2/Unit 4/ Section 4.5 on page 22.

Check Your Progress 2

- 1) Create t -distribution for the data given in Figure 7 and Figure 8 of this section using a spreadsheet package.

.....

.....

.....

- 2) Create chi square-distribution for the data given in Figure 9 of this section using a spreadsheet package.

.....

.....

.....

- 3) Create F -distribution for the data given in Figure 10 of this section using a spreadsheet package

.....

.....

.....

Check Your Progress 3

- 1) Develop all the spreadsheets as shown in Figure 11 to Figure 14. Can you now make effective use of names for ranges instead of cell references?

.....

.....

.....

1.8 SUMMARY

This section has been an attempt to provide you details on some of the basic concepts

1.9 ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 1:

Three-

Comparison with the MVC architecture[\[edit source\]](#) | [\[editbeta\]](#)

At first glance, the three tiers may seem similar to the [model-view-controller](#) (MVC) concept; however, topologically they are different. A fundamental rule in a three tier architecture is the client tier never communicates directly with the data tier; in a three-tier model all communication must pass through the middle tier. Conceptually the three-tier architecture is linear. However, the MVC architecture is triangular: the view sends updates to the controller, the controller updates the model, and the view gets updated directly from the model.

From a historical perspective the three-tier architecture concept emerged in the 1990s from observations of distributed systems (e.g., web applications) where the client, middle ware and data tiers ran on physically separate platforms. Whereas MVC comes from the previous decade (by work at [Xerox PARC](#) in the late 1970s and early 1980s) and is based on observations of applications that ran on a single graphical workstation; MVC was applied to distributed applications later in its history (see [Model 2](#)).

Today, MVC and similar [model-view-presenter](#) (MVP) are [Separation of Concerns](#) design patterns that apply exclusively to the presentation layer of a larger system. In simple scenarios MVC may represent the primary design of a system, reaching directly into the database; however, in most scenarios the Controller and Model in MVC have a loose dependency on either a Service or Data layer/tier. This is all about Client-Server architecture.

Web development usage[\[edit source\]](#) | [\[editbeta\]](#)

In the [web development](#) field, three-tier is often used to refer to [websites](#), commonly [electronic commerce](#) websites, which are built using three tiers:

1. A front-end [web server](#) serving static content, and potentially some [cached](#) dynamic content. In web based application, Front End is the content rendered by the browser. The content may be static or generated dynamically.
2. A middle dynamic content processing and generation level [application server](#), for example [Ruby on Rails](#), [Java EE](#), [ASP.NET](#), [PHP](#), [ColdFusion](#), [Perl](#) platform.
3. A back-end [database](#) or [data store](#), comprising both data sets and the [database management system](#) or [RDBMS](#) software that manages and provides access to the data.

Other considerations[\[edit source\]](#) | [\[editbeta\]](#)

Data transfer between tiers is part of the architecture. Protocols involved may include one or more of [SNMP](#), [CORBA](#), [Java RMI](#), [.NET Remoting](#), [Windows Communication](#)

[Foundation](#), [sockets](#), [UDP](#), [web services](#) or other standard or proprietary protocols. Often [middleware](#) is used to connect the separate tiers. Separate tiers often (but not necessarily) run on separate physical servers, and each tier may itself run on a [cluster](#).

Traceability[[edit source](#) | [editbeta](#)]

The end-to-end traceability of data flows through n -tier systems is a challenging task which becomes more important when systems increase in complexity. The [Application Response Measurement](#) defines concepts and [APIs](#) for measuring performance and correlating transactions between tiers.

Comments[[edit source](#) | [editbeta](#)]

Generally, the term "tiers" is used to describe physical distribution of components of a system on separate servers, computers, or networks (processing nodes). A three-tier architecture then will have three processing nodes. The term "layers" refer to a logical grouping of components which may or may not be physically located on one processing node.

Check Your Progress 2:

Please use any spreadsheet package and enter all the data as shown in

Check Your Progress 3:

download the file containing all these figures. Find out what formulas have been entered. Also find the errors in the entry of formula. You may define cell ranges and use in formulas.

1.10 FURTHER READINGS

Web link:

- www.wikipedia.org

UNIT 2 JSP - Basic

Structure

- 2.0 Introduction
- 2.1 Objective
- 2.2 JSP: An Introduction
- 2.3 JSP Life Cycle
- 2.4 Elements of JSP
 - 2.4.1 Directives
 - 2.4.1.1 page Directive
 - 2.4.1.2 include Directive
 - 2.4.1.3 taglib Directive
 - 2.4.2 Scripting
 - 2.4.2.1 Declarations
 - 2.4.2.2 Expressions
 - 2.4.2.3 Scriptlets
 - 2.4.3 Action Elements
 - 2.4.3.1 jsp:useBean
 - 2.4.3.2 jsp:setProperty
 - 2.4.3.3 jsp:getProperty
 - 2.4.3.4 jsp:param
 - 2.4.3.5 jsp:include
 - 2.4.3.6 jsp:forward
 - 2.4.3.7 jsp:plugin
 - 2.4.3.8 jsp:fallback
- 2.5 Comments and Template Data
- 2.6 JSP Implicit Object
 - 2.6.1 request
 - 2.6.2 response
 - 2.6.3 session
 - 2.6.4 application
 - 2.6.5 page
 - 2.6.6 pageContext
 - 2.6.7 out
 - 2.6.8 config
 - 2.6.9 exception
- 2.7 Complete Example
- 2.8 Summary
- 2.9 Solutions/Answers

2.0 Introduction

In the previous block, you have learned that how to create HTML web pages. It is static pages. When you will include java code inside the html page, it becomes Java Server Pages (or JSP). Java Server Pages are simple but powerful technology used to generate dynamic web pages. Dynamic web pages are different from static web pages in that web server will create a web page when it is requested by a client or user. For example, your online results on IGNOU website, the page for every student instead IGNOU web server dynamically creates a page depending on your roll number.

JSP is released in 1999 by SUN Microsystems. JSP is a technology for developing web pages. JSP is similar to PHP, but it uses the Java programming language. It follows the characteristics of Java 'write once and run anywhere. JSP pages are platform independent means it can run in any platform. It enables you to add dynamically generated content with static html. In addition to html, JSP page are built using different components viz., directives, scripting elements, standard actions and implicit objects. This unit covers how to create a JSP page. It also provides a basic understanding of Java Bean, custom tag and life cycle of Java Server Page.

2.1 Objectives

In this Unit, you will learn the following:


- how to create a JSP pages
- how to include HTML or plain text files at the time the client requests the page
- how to create and access Java Bean within JSP page
- how to create a custom tag
- how to define JSP predefined variables that can be used within scriptlets and expression
- how to forward request from JSP page to other resource
- how to include applet within JSP page

2.2 JSP: An Introduction

Java Server Pages (JSP) is a web technology that helps software developers to create dynamic content based web pages. Unlike a plain HTML page, which contains static content that always remains the same but in JSP; you can change content dynamically with the help of Java Bean and JSP elements. JSP is an extension of Java Servlet because it provides more functionality than servlet. Servlet are server side components that services requests from a web server. It is basically a class that run inside the Java Virtual Machine. Servlet is very useful for writing server side code but it suffer from some disadvantages. In particular, writing HTML code with plenty out.println() statements (println() is a method of system.out object to display string which is passed to it), it is very tedious and error prone and also software developers has to take on dual roles of developing application logic and designing web pages. JSP is designed to address these disadvantages.

A Java Server Page contains HTML tags as well as JSP elements. The JSP elements are basic building blocks of the page. The JSP elements are easier to maintain than the servlet. A JSP page contains a very simple structure that makes it easy to developers to write JSP code and also easy to servlet engine to translate the page into a corresponding servlet. In addition to html tags, a JSP page consists of directives, scripting elements, scriptlets and action elements. Each of these elements can use either JSP syntax or they can be expressed in XML syntax but you cannot intermix the two. For this problem, you can use the include mechanism to insert file that may use different syntax.

The Java Server Pages has more advantages over the servlet which are as follows:



Web Servers are computers that using client/server model and connected to internet for serving web pages.

- It allows programmers to insert the Java code directly into the JSP file that makes the development process easier.
- JSP support element based dynamic content that allows programmers to develop custom tags libraries to satisfy application needs.
- Content and display logic are separated
- JSP pages can be used in conjunction with servlet that handle business logic.

You simply write the regular html with the extension *.jsp. The following code contains a simple example of a JSP page:

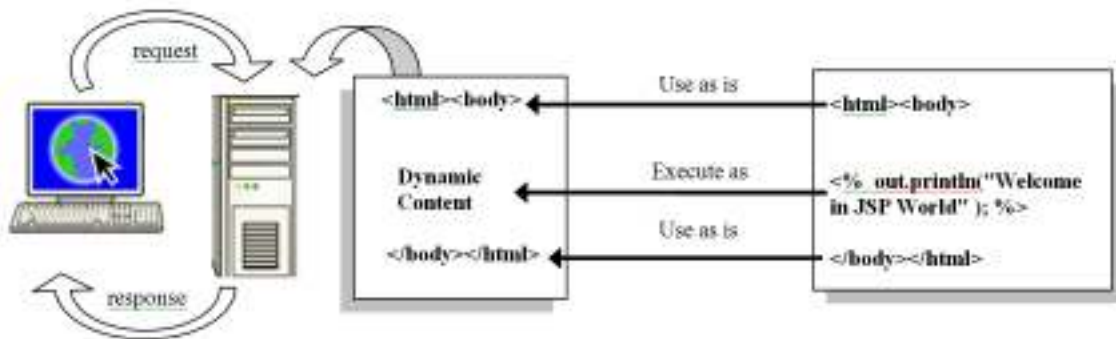


Figure 1: Generating dynamic content with JSP page.

The output of the above program is as follows:

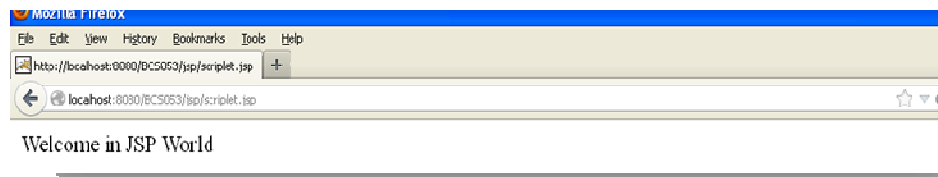


Figure 2: JSP Page

You can see the above program which looks like any other HTML page with some added JSP elements that allow the server to insert dynamic content in the page. When client send a request for a JSP page, the server executes a JSP page elements merges with static contents and sends the dynamically page back to the client browser, as illustrated in figure-1.

The JSP technology is based on JSP API (Application Programming Interface) that consists of two packages i.e. javax.servlet.jsp and javax.servlet.jsp.tagext packages. In addition to these two packages, JSP also needs two packages of servlet such as javax.servlet and javax.servlet.http. Apart from these interfaces and classes, the two exception classes: JspException and JspError are also defined in JSP API. The javax.servlet.jsp package has two interfaces such as HttpJspPage and JspPage and four classes: JspEngineInfo, JspFactory, JspWriter and PageContext. The jspInit() and jspDestroy() methods are defined in JspPage interface and _jspService() method is in HttpJspPage interface. The javax.servlet.jsp.tagext contains classes and interfaces for the definition of Java Server Pages Tag Libraries.

2.3 JSP Life Cycle

In this section, you will go through the life cycle of JSP and see how a JSP page is displayed. When the JSP is first accessed, it is translated into corresponding servlet (i.e. java class) and compiled, then JSP page services request as a servlet. The translation of JSP page is done by the JSP engine of the underlying web container/servlet container (eg. Tomcat). The following figure-3 shows that how a JSP page is processed.

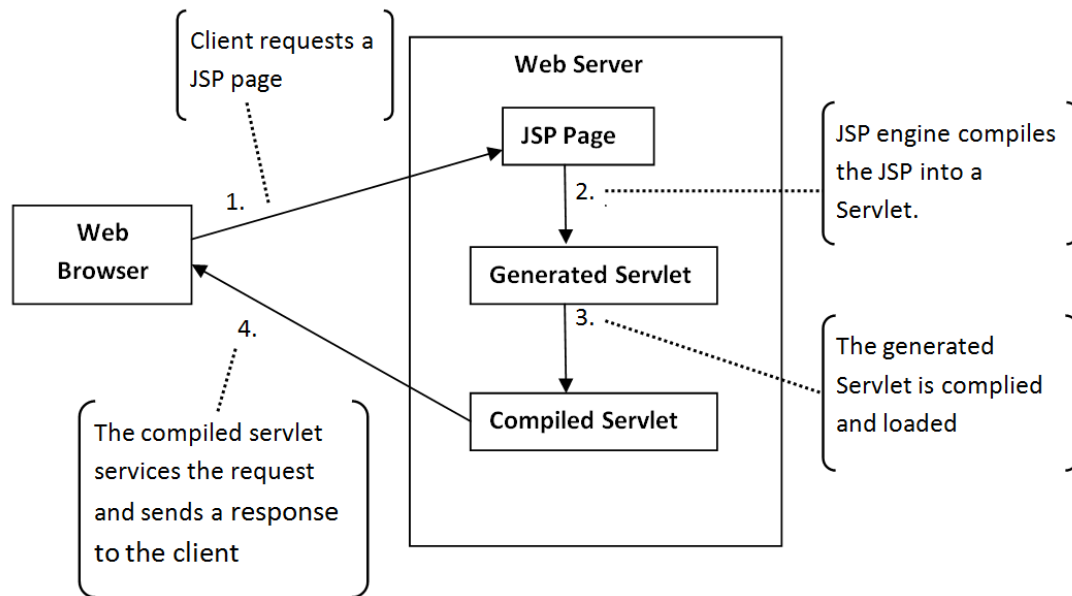


Figure 3: The steps of a JSP Page processing.

The life cycle of JSP page is controlled by three methods i.e. `jspInit()`, `_jspService()` and `jspDestroy()`.

`jspInit()` - The `jspInit()` method is same as the `init()` method of servlet and it is called only once during life cycle of a JSP/Servlet. It is used to initialize objects and variables that are used throughout the life cycle of JSP. This method is defined in `JspPage` interface. This method is invoked when the JSP page is initialized. Once the JSP page is initialized, the `getServletConfig()` method will return the desired value. It has no parameters, return no value and thrown no exceptions. The signature of the method is as follows:

```
public void jspInit() { // Initialization code }
```

`_jspService()` - `_jspService()` is the method which is called every time the JSP page is requested to serve a request. This method is defined in the `javax.servlet.jsp.HttpJspPage` interface. This method takes `HttpServletRequest` and `HttpServletResponse` objects as an arguments. The `_jspService()` method corresponds to the body of the JSP page. It is defined automatically by the processor and should never be redefined by the JSP author. It returns no value. The underscore ('_') signifies that you can not override this method. The signature of the method is as follows:

```
public void _jspService(
    javax.servlet.http.HttpServletRequest request,
    javax.servlet.http.HttpServletResponse response)
    throws javax.servlet.ServletException, java.io.IOException
{
    // services handling code
}
```


jspDestroy()- The jspDestroy() is invoked when the JSP page is to be terminated. It is synonymous with the destroy() method of a servlet. It has no parameters, return no value and thrown no exceptions. Override jspDestroy() when you need to perform any cleanup, such as releasing database connections or closing open files. The signature of the method is as follows:

```
public void jspDestroy(){ // cleanup code }
```

The following figure-4 shows the life cycle of JSP page.

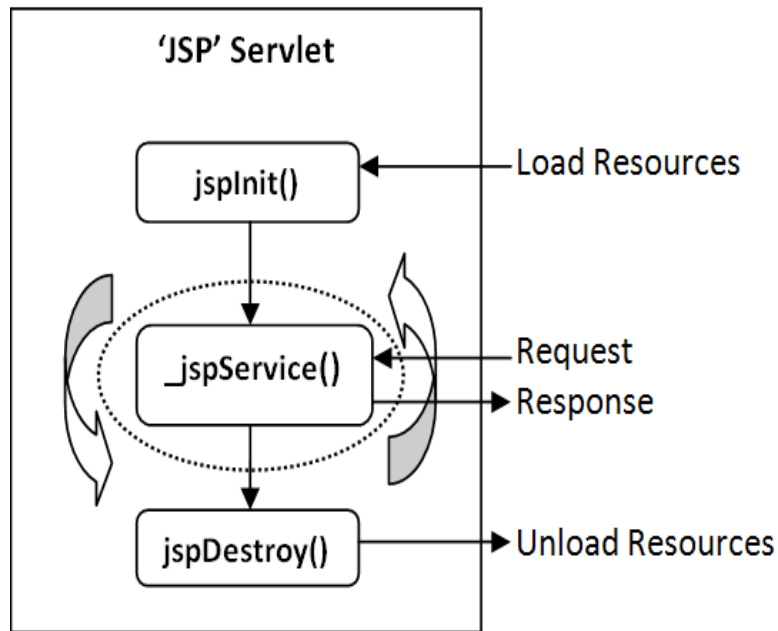


Figure 4: Life Cycle of JSP Page

Check Your Progress 1

1. Explain the term JSP page.

2. What are the advantages of JSP over Servlet?

3. Describe the _jspService() method.

4. Write the basic steps for processing JSP request.

5. Explain the life cycle of JSP.

2.4 Elements of JSP

In this section, we are going to introduce the elements of JSP that make up a JSP page. There are three types of JSP components such as directives, expressions and scriptlets. A directive affects the overall structure of the JSP page. They are discussed in detail in the following sections:

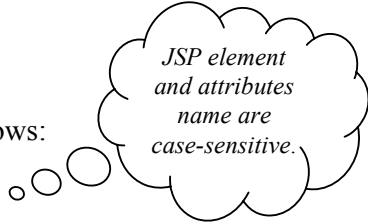
2.4.1 Directives

Directives are used as guiding the JSP container for translating and compilation of JSP page. It appears at the top of the page. Using directives, the container translate a JSP page into corresponding servlet. They do not directly produce any output. A directive contains one or more attribute name/value pairs. Directives are defined by using `<%@` and `%>` tags. Directives have the following syntax:

`<%@ directive attribute = "value" %>`

You can write XML equivalent of the above syntax as follows:

`<jsp:directive. directivename attribute = "value" />`



*JSP element
and attributes
name are
case-sensitive.*

There are three types of directives currently used in JSP document: *page*, *include* and *taglib*. Each one of these directives and their attributes are defined in following sections:

2.4.1.1 page Directive

The page Directive is used to specify the attributes of JSP page such as declaration of other resources i.e. classes and packages, page buffering requirements and name of page that should be used to report run time errors, if any. The page Directive is a JSP element that provides global information about an entire JSP page. This information will directly affect the compilation of the JSP document. Following is the syntax of page directive:

`<%@ page attribute = "value" %>`

The page directive contains many attributes, you can set these attributes. The attribute may be one or more of the following:

Attribute	Description
language="scripting language"	This attribute define the language that will be used to compile the JSP document. By default, it is java language.
import="import list"	This attribute defines the names of packages.
session="true false"	It specifies whether or not the JSP document participate in HTTP session. The default is <i>true</i> .
extends="classname"	This attribute define the name of parent class that will be inherited by generated servlet. It is rarely used.
buffer="none size in kb"	The default value is 8kb. It specifies the size of <i>out</i> buffer.

autoFlush="true false"	The default is <i>true</i> . It means that the <i>out</i> buffer will be automatically flushed when full. If it is <i>false</i> , it will be raised an exception when buffer is full.
Info='text'	If this attribute is used, the servlets will override the <code>getServletInfo()</code> method.
errorPage="error_page_url"	This attribute defined the relative URL to JSP document that will handle exception.
isErrorPage="true false"	This attribute indicates whether the current page can act as an error page for another JSP page. The default is <i>false</i> .
isThreadSafe="true false"	The default value is <i>true</i> . It indicates that the page can service more than request at a time. When it is <i>false</i> , the <code>SingleThreadModel</code> is used.

An example of the use of page directive is as follows:

```
<%@ page import="java.io.*, java.util.Date" buffer="16k" autoFlush="false" %>
```

This page directive instructs the web container to import `java.io` package and `java.util.Date` class. It also instructs the web container to set buffer size to 16k and turn off autoflushing.

You can specify multiple page directives in your JSP document, such as the following:

```
<%@ page import="java.util.Date" info="Example Page" %>
<%@ page errorPage="MyErrorPage.jsp" buffer="16kb"%>
```

In the above examples of page directives, the first directive statement tells the web container to import `java.util.Date` class and to set information about the JSP page which is retrieved by `getServletInfo()` method. The value of the `info` attribute will be a text string. The second page directive statement define a name of error page which is used for handling errors and set the buffer size in 16 kb.

2.4.1.2 include Directive

Include directive is an important JSP directive. This is used to insert text and code in the form of file such as `html`, `JSP` into a current JSP document at the translation time. It means that it enables you to import the content of another static file into a current JSP page. This directive can appear anywhere in a JSP document. The syntax of include directive is as follows:

```
<%@ include file = "relative URL" %>
```

For example: The following example will demonstrate the physically inclusion of header file. In this way, you can create a single header file only once and use it many times for your website.

```
<html><body>
<h2>Example of include directive</h2>
<%@ include file='header.html' %>
</body></html>
```

Source code for header.html:

```
<html><body>
<h2>This is from Header File</h2>
</body></html>
```

***Relative URL** is only designates the filename or resource name. not fully qualified; while **Absolute URL** is designates the protocol, host, path, and name of the resource name.*

The output of the above program is as follows:

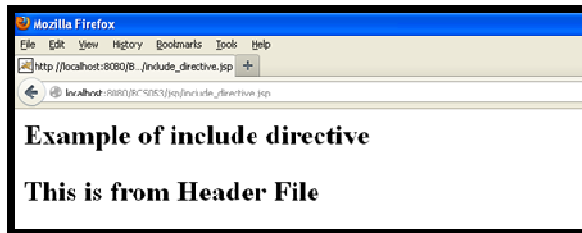
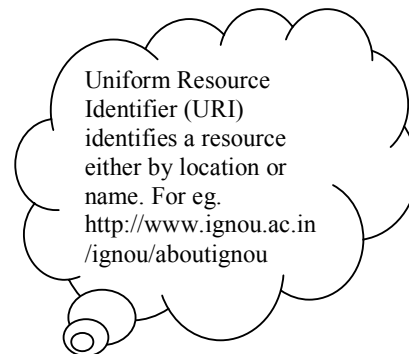


Figure 5: Example of include directive



2.4.1.3 taglib Directive

As yet, you have learned the basic elements of JSP. In this section, you will learn about the creation of custom tag libraries in Java Server Pages.

A custom tag is user defined tag. It is a reusable code in a JSP page and tag library is a collection of custom tags.

Custom Tag Syntax -

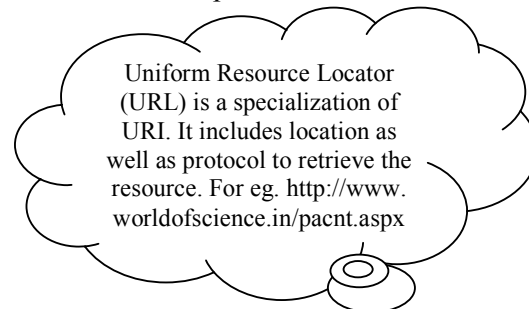
To use the customs tags in a JSP page, JSP technology provide a taglib directive to make use of the tag. The taglib directive has the following syntax:

```
<%@ taglib uri="tagLibraryURI" prefix="tagPrefix" %>
```

The uri attribute defines an absolute or relative uri of tag library descriptor (TLD) file and the prefix attribute defines the string that will identify a custom tag instance.

To use customs tags in a JSP page, you need to know four components that make use of the custom tags. These are:

- 1) Tag handler class
- 2) Tag library descriptor file
- 3) JSP file
- 4) Deployment descriptor file (web.xml)



Tag Handler Class -

It is a java class that defines the behaviour of the tags. This class must implement the javax.servlet.jsp.tagext package.

Tag Library Descriptor (TLD) file -

A tag library descriptor file is an xml document. It defines a tag library and its tags. The file extension of this file is .tld. It contains one <taglib> root element and tlibversion, jspversion, shortname, tag are sub elements of the taglib element. The tag is the most important element

in TLD file because it specifies the name of the tag and class name of the tag. You can define more than one tag element in the same TLD file.

Deployment Descriptor file (web.xml) -

The deployment descriptor is an xml file that specifies the configuration details of the tag. The most important element for custom tag in web.xml file is <taglib-location>. Using the web.xml, the JSP container can find the name and location of the TLD file.

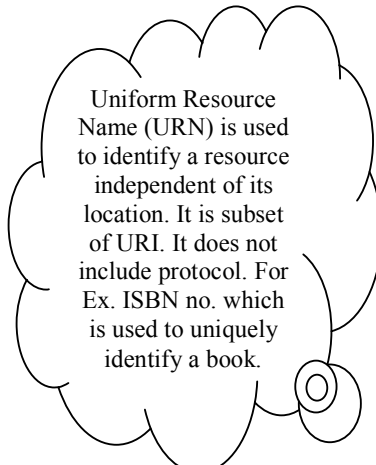
A JSP file -

Once, you have created tag handler java class, a tag descriptor file and define configuration details in deployment descriptor file and then you have to write a JSP file that makes use of the custom tag.

Here are some steps for generating a simple custom tag. Now, follows the following five easy steps:

Step-1: write and compile a java class called MyCustomTag.java which is given in following program source. The class file must be placed in the directory say 'customTag' under the WEB-INF/classes directory.

```
package customTag;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
public class MyCustomTag extends TagSupport {
public int doEndTag() throws JspException {
JspWriter out = pageContext.getOut();
try { out.println("Hello from custom tag"); }
catch(Exception e) {}
return super.doEndTag();
} //doEndTag()
} //main class
```



Uniform Resource Name (URN) is used to identify a resource independent of its location. It is subset of URI. It does not include protocol. For Ex. ISBN no. which is used to uniquely identify a book.

Step-2: Create a TLD file named taglib.tld as shown in following program source and save it in WEB-INF directory.

```
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library
1.2//EN" "http://java.sun.com/j2ee/dtd/web-jsptaglibrary_1_2.dtd">
<taglib><tlib-version>1.0</tlib-version><jsp-version>1.2</jsp-version>
<short-name></short-name> <tag> <name>myTag</name>
<tagclass>customTag.MyCustomTag</tagclass></tag> </taglib>
```

Step-3: Create a JSP file named CustomTag.jsp that contains the following code:

```
<html><body><%@ taglib uri="/myTLD" prefix="easy" %>
<easy:myTag /></body></html>
```

Step-4: Place the following code in web.xml under the <web-app> root element.

```
<taglib> <taglib-uri> /myTLD </taglib-uri>
<taglib-location>/WEB-INF/taglib.tld </taglib-location>
</taglib>
```

Step-5: Start server say Tomcat (if you are using this). Open web browser and run the JSP page. The following screen comes as an output for a simple custom tag.

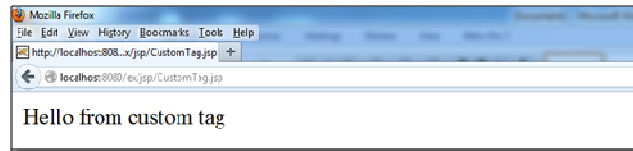


Figure 6: A simple JSP Custom Tag page

When the user requests the JSP page, the JSP container first sees the taglib directive and gets the taglib uri. On the basis of this, JSP container looks into the web.xml file to find taglib location and continues the processing and get the name of tag. After getting the name and location of TLD file, it obtains the java class. Now, the JSP container loads the class for custom tag and continues the processing.

2.4.2 Scripting Elements

Scripting elements allow you to insert java code fragments directly into an HTML page. You can specify three ways to include java code into your JSP document, such as declarations, expression and scriptlets. Each of these scripting elements has an appropriate location in the generated servlet. These are discussed in more detail in following sections:

2.4.2.1 Declarations

As its name implies, declarations are used to declare the variables and methods that can be used in the JSP document. The declaration part is initialized when the JSP document is initialized. After the initialization, they are available to other expressions, declaration and scriptlets. A declaration is start with a `<%!` and end with a `%>`. The syntax of the declaration is as follows:

```
<%! declaration %>
```

You can write XML equivalent of the above syntax as follows:

```
<jsp:declaration> code fragment </jsp:declaration>
```

For example: simple declaration of variables

```
<%! int i = 0; %>    // i is an integer type variable
<%! int a, b, c; %>  // a, b, c is an integer type variable
<%! Square a = new Square (4.0); %>
```

Following is a variable and method declaration:

```
<%! String name = new String("SOCIS"); %>
<%! public String getName() { return name; } %>
```

When the above code is automatically compiled and converted from JSP code to servlet, the declaration part of JSP page is included in the declaration section of the generated servlet.

The first declaration statement is used to defined a variable as string type with initial value 'socis' and second statement is defined a string type method named 'get Name()'.

2.4.2.2 Expressions

An Expression is an instruction to the web container to execute the code within the expression and to display the resulting data at the expression's referenced position in the JSP document. Expressions are evaluated at request time i.e. run time. The syntax of the expression is as follows:

```
<%= expression %>
```

You can write XML equivalent of the above syntax as follows:

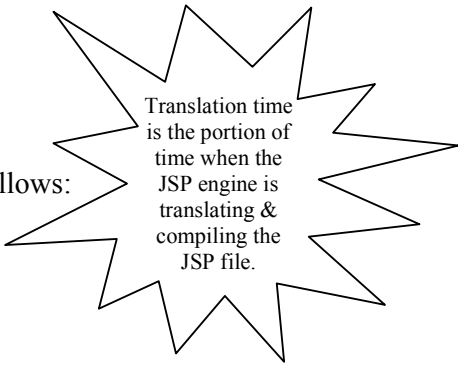
```
<jsp: expression > code fragment </jsp: expression >
```

For example, to show the current date and time:

```
<%= new java.util.Date() %>
```

Consider another example:

```
<html><body>
<%! String name = new String("SOCIS"); %>
<%! public String getName() { return name; } %>
Hello <b><%= getName()%></b><br/><%= new java.util.Date() %>
</body></html>
```



Translation time is the portion of time when the JSP engine is translating & compiling the JSP file.

The output of the above program is as follows:

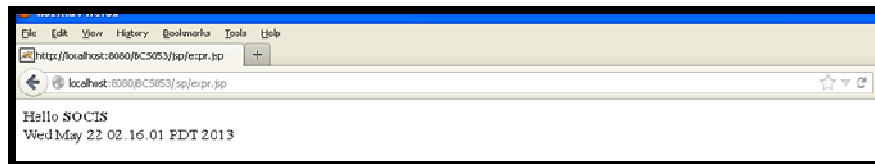


Figure 7: Expression example

When the above code is compiled and converted to servlet code and then expression part is placed in its referenced position of the generated servlets' _jspService() method.

2.4.2.3 Scriptlets

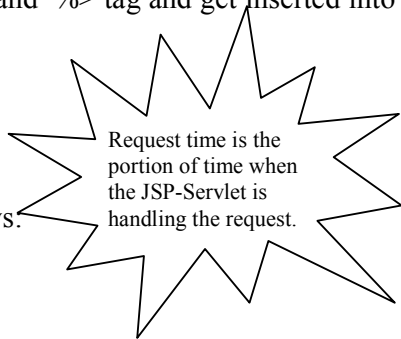
In scriptlets, all the scripting elements are brings together. It is executed at the request time and makes use of declaration, expressions and JavaBeans. You can write scriptlets anywhere in a page. It contains a valid java statements within the <% and %> tag and get inserted into the _jspService() method of generated servlet.

The syntax of the scriptlet is as follows:

```
<% scriptlet code %>
```

You can write XML equivalent of the above syntax as follows:

```
<jsp:scriptlet > code fragment </jsp:scriptlet >
```



Request time is the portion of time when the JSP-Servlet is handling the request.

For example: The following example will demonstrate the number series from 1 to 3.

```
<html><body><p>Counting to 1- 3 :</p>
<% for (int i=1; i<=3; i++) {%><p>This number is <%= i %>.</p>
<% } %></body></html>
```

Output of the above program is as follows:

```
Counting to 1- 3 :
This number is 1.
This number is 2.
This number is 3.
```

Figure 8: scriptlets Example

2.4.3 Actions Elements

Action elements (or standard actions) are tags that can be embedded in a JSP document. At the compile time, they are replaced by java code.

Before going to start learning about how you can add Java Bean in JSP page, you must take a look at what a bean is. A Java Bean is nothing more than a java class. It is reusable component that work on any Java Virtual Machine. For the creation of Java Bean, you must create a java class that implements `java.io.Serializable` interface and uses public `get/set` methods to show its properties.

2.4.3.1 <jsp:useBean>

The first JSP standard action (identified by *jsp* prefix) is `<jsp:useBean>`. This action is used to include an instance of java bean within java server pages. The syntax of the `<jsp:useBean>` action is as follows:

```
<jsp:useBean id="name"
             scope="page | request | session | application"
             typeSpecification>
    body
</jsp:useBean>
```

typeSpecification can be represented in the following syntax:

```
typeSpecification ::= class="className" |
                     class="className" type="typeName" |
                     type="typeName" beanName="bean_name" |
                     type="typeName"
```

Following table contains the attributes of the `<jsp:useBean>` action:

Attribute	Description
-----------	-------------

id	It represents name of the object. This attribute is required.
scope	It defined the scope of the object. It may be <i>page</i> , <i>request</i> , <i>session</i> or <i>application</i> . This attribute is optional. By default, it is <i>page</i> .
Class	This attribute represents the fully qualified class name of the object. The class name is case sensitive.
type	It specifies the type of object. The value of this attribute is equal to <i>class</i> , a super class of <i>class</i> or an interface implemented by the <i>class</i> . If it is not specified then same as class attribute.
beanName	It is the name of bean.

The **scope** attribute of Java Bean means how long the object is available and if it is available than only for a single user or all application users JSP provides different scope for sharing data between web pages. These are:

- **Page** - 'page' scope means, the JSP object can be accessed only from within the same page where it is created. By default, it is page. JSP implicit objects out, exception, response, pageContext, config and page have 'page' scope.
- **Request** – The object is available to current JSP and to any JSP or Servlet that control is forward to and included from. Only Implicit object request has the 'request' scope.
- **Session** – JSP object is accessible from any JSP within the same session. Implicit object session has the 'session' scope.
- **Application** - JSP object is accessible from any JSP within the same web application. Implicit object application has the 'application' scope.

2.4.3.2 <jsp:setProperty>

This action is used to sets the Java Beans property value. The syntax for this action is as follows:

```
<jsp:setProperty name="beanName" property_expression />
```

property_expression can be represented in the following syntax:

```
property="*" |
property="propertyName" |
property="propertyName" param="parameterName" |
property="propertyName" value="propertyValue"
```

Following table contains the attributes of the <jsp: setProperty > action:

Attribute	Description
name	It represents name of the bean instance defined in <jsp:useBean> action.
property	It specifies the name of property being set. This attribute is required. If it is an asterisk (*) then it is used to set all properties of bean. You can also specify the specific bean property.
param	This attribute represents the name of parameter.
value	It specifies the value assigned to the named bean's property.

2.4.3.3 <jsp:getProperty>

Once you have defined a bean and given it a name using the `<jsp:useBean>` action. Now, you can get the beans property values with another standard action `<jsp:getProperty>`. This action is used to get the property value from a Java Beans and add to the response body. The syntax for this action is as follows:

```
<jsp:getProperty name="beanName" property="propertyName" />
```

Following table contains the attributes of the `<jsp: getProperty >` action:

Attribute	Description
name	It represents name of the bean instance defined in <code><jsp:useBean></code> action.
property	This attributes represents the specific property within bean.

Following is the simple Java Bean example that store student name.

Let's create a Java Bean named `student.java` and place class file under `WEB-INF/classes/bean1` directory.

```
package bean1;
public class student {
    private String name;
    public String getName() { return name;}
    public void setName(String name) { this.name=name;}}
```

Now, you can access the properties of the Java Bean from a JSP.

```
<html><body>
<jsp:useBean id="myBean" class="bean1.student" scope="session" />
<jsp:setProperty name="myBean" property="name" value="Poonam" />
Welcome to this Unit, <jsp:getProperty name="myBean" property="name" />
, Please visit egyankosh.ignou.ac.in to get more.
</body></html>
```

The following output screen is as follows for the above program.

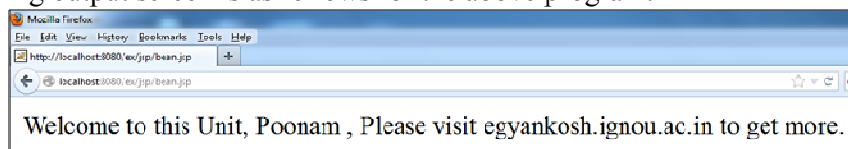


Figure 9: A simple Java Bean Example

You have learned the standard actions specific to a Java Bean. The remaining actions are defined in the following sections. Using the following standard actions, you can dynamically insert a resource into a current JSP page, forward the current request to another page or add bean/applet in a page.

2.4.3.4 `<jsp:param>`

The `<jsp:param>` action tag defines the name/value pair of parameter to be passed to an included or forwarded JSP document or a page that uses the `<jsp:plugin>` tag.

Following is the syntax of the above action tag:

```
<jsp:param name="paramName" value="paramValue" />
```

In the above syntax, name attribute defines the name of parameter being referenced and value represents the value of named parameter.

2.5.2 <jsp:include>

JSP supports two types of include: static and dynamic. In the previous section, you have already been studied the static include i.e. include directive. In static include, the content of included JSP code is inserted into the including JSP (or current JSP Document) at the translation time. After the content of included JSP is processed, the content included in JSP file does not change until the included file is changed and server is restarted. While in dynamic include, the content is included at the request time. This means that it is a mechanism for including static as well as dynamic content in the current JSP document such as static HTML, JSP or Servlet. This is done by <jsp:include> action element.

Dynamic includes are performed using the following syntax:

```
<jsp:include page="relativeURLSpecification" flush="true" />
or
<jsp:include page="relativeURLSpecification" flush="true" >
    <jsp:param ...../>
</jsp:include>
```

In the above syntax, page attribute defines the path of the resource to be included and flush attribute indicates whether the buffer is flushed. It is an optional parameter. The first syntax is used when <jsp:include> does not have a parameter name/value pair. If you want to pass the parameter to the included resource, use the second syntax.

For example:

```
<html><head><title>Student Information</title></head><body>
<h2>JSP Include Example</h2><table><tr><td>
<jsp:include page="head.jsp" flush="true">
<jsp:param name="student" value="Poonam"/>
<jsp:param name="prg" value="BCA"/>
</jsp:include>
</td></tr><tr><td>Course Name:BCS-053</td></tr></table>
</body></html>
```

Source code for **head.jsp** file:

```
<% // Get the Student Name from the request
    out.println("<b>Student Name:</b>" + request.getParameter("student"));
    out.println("<br>");
    // Get the Course Name from the request
    out.println("<b>Programme Name:</b>" + request.getParameter("prg")); %>
```

Output screen for the above program:

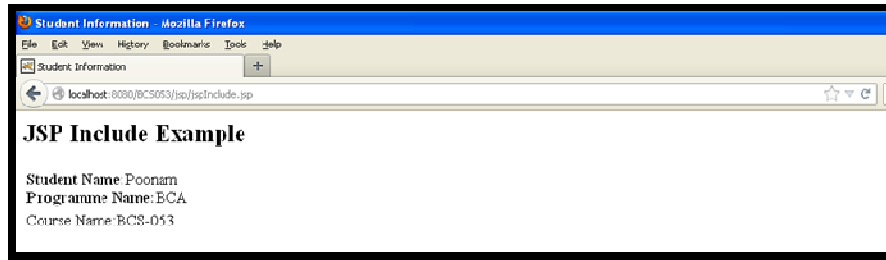


Figure 10: JSP Include Example

2.5.3 <jsp:forward>

The <jsp:forward> action is used to terminate the current execution of JSP page and transfer the control to the another JSP page within the same application. It may be static or dynamic page or resource. It can be used with <jsp:param> tag. The <jsp:param> is used for providing the values for parameters in the request to be used for forwarding.

The syntax for <jsp:forward> action tag is as follows:

```
<jsp: forward page="relativeurlSpecification" flush="true" />
                        or
<jsp: forward page="relativeurlSpecification" flush="true" >
    <jsp:param ...../>
</jsp: forward >
```

For example:

```
<html><head><title>Example: JSP Forward</title></head><body>

<% if((request.getParameter("city")).equals("delhi")) { %>
    <jsp:forward page="head.jsp" >
    <jsp:param name="student" value="Poonam"/>
    <jsp:param name="prg" value="BCA"/></jsp:forward>

<% } else { out.println("Your value is incorrect"); } %>

</body></html>
```

Source code for head.jsp file is same as in the above <jsp:include> example. Enter the following URL into your browser to see the result.

<http://localhost:8080/BCS053/jsp/jspForward.jsp?city=delhi>

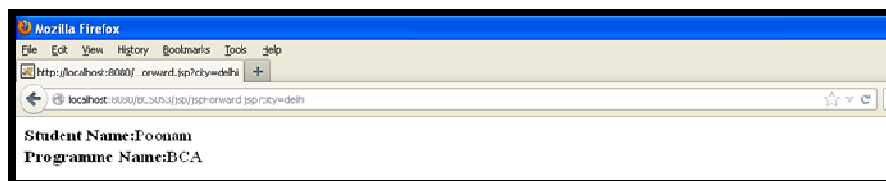


Figure 11: JSP forward example

2.5.4 <jsp:plugin>

The <jsp:plugin> action element is used to insert java component such as Applets and JavaBean in JSP page. The <jsp:param> is also used with action element to send parameters to Applet or Bean. Following is the syntax of <jsp:plugin> action:

```
<jsp:plugin type="pluginType" code="classFile" codebase="relativeURLpath">  
  <jsp:param ...../>  
</jsp: plugin >
```

In the above syntax, type attribute indicates the type of plugin to include in JSP page, code attribute represent the name of class file and codebase attribute is the path of where the code attribute can be found.

For example: An Applet program named JavaApplet.java

```
import java.applet.*;  
import java.awt.*;  
  
public class JavaApplet extends Applet{  
  public void paint(Graphics g){  
    g.drawString("Welcome in Java Applet.",40,20);  
  } }  
}
```

Plugin source code:

```
<html><body><p>Plugin Example :</p>  
<jsp:plugin type="applet" code="JavaApplet.class" width = "200" height = "200">  
  <jsp:fallback>  
    <p>Unable to load applet</p>  
  </jsp:fallback>  
</jsp:plugin>  
</body></html>
```

The output of the above plugin program is as follows:

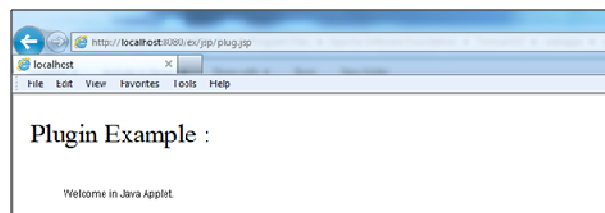


Figure 12:plugin example

2.5.3 <jsp:fallback>

The <jsp:fallback> action is used only with the <jsp:plugin> action element. It provides alternative text to the browser.

```
For example: <jsp:fallback> Unable to see plugin </jsp:fallback>
```

Check Your Progress 2

1. Write the names of JSP components with their syntax.

2. What are Standard actions in JSP? Explain the standard action specific to Java Bean.

3. What is difference between JSP include and include directive?

4. What is declaration scripting element? Explain with example.

5. Write a program which will demonstrate the use of <jsp:include> with <jsp:param> action.

2.6 Comments and Template Data

JSP Comments

Commenting is a part of good programming practice. Comments help in understanding what is actually code doing. You can write comments in JSP like the following way:

JSP comment syntax:

<%--jsp comment text --%>

This comments tag is used within a Java Server Page for documentation purpose. It tells the JSP container to ignore the comment part from compilation, so it does not appear in the resulting java code.

<!-- HTML comment --> This HTML comment is treated by JSP container as just like another html tag. It is not a JSP comment. It can be viewed through the web browser's "view source" option.

/* java comment */ or **//** used inside the scriptlets tag is also not a JSP comment. This is a java comment. This comments are not translated by JSP container and therefore do not appear on the client side web document.

For example:

```
<html><body>
<%-- This is JSP comment – not visible in the page source --%>
<!-- This HTML Comment - visible only in the page source
<% out.println("Welcome in JSP World" ); %> -->
<% //This is single line comment under the scriptlet %>
<% /* out.println("This comment is used for multiple lines."); */ %>
```

Program output:

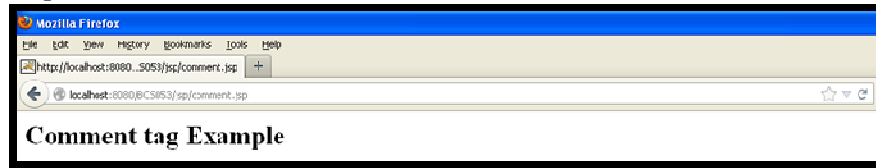


Figure 13: JSP comment example

Program output is visible only in page source code of web browser.

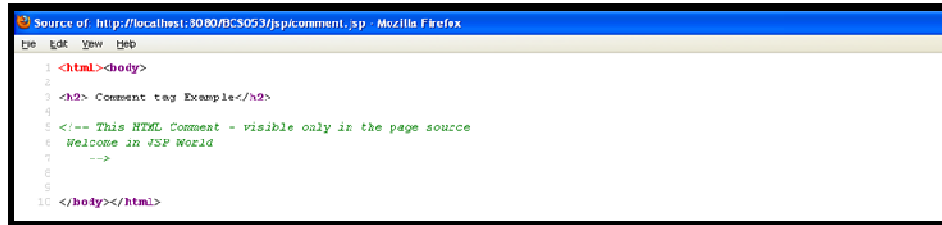


Figure 14: HTML comment's view in web browser

JSP XML Text Element

The `<jsp:text>` is an XML tag. It is used to enclose template data in JSP document. Template(static) data consists of any text that is not interpreted by the JSP translator. You cannot intermix JSP data with XML data in single file even you can include directive. The XML syntax for text element is as follows:

```
<jsp:text> text </jsp:text>
```

2.7 JSP Implicit Object

JSP Implicit objects are the java objects that are available for the use in JSP document to developers and they can call them directly without being declared first. These objects are also called predefined variables. These variables are parsed by JSP engine and inserted into generated servlet as if you defined them yourself. These objects are used within the scriptlets and expressions of the JSP page. These implicit objects are defined in the servlet specification's `javax.servlet.http` package, two are part of the JSP `javax.servlet.jsp` package and some is in Java core API.

JSP supports implicit objects which are listed below:

Implicit Object	Type
request	<code>javax.servlet.HttpServletRequest</code>
response	<code>javax.servlet.HttpServletResponse</code>
session	<code>javax.servlet.http.HttpSession</code>
application	<code>javax.servlet.ServletContext</code>
page	<code>javax.servlet.jsp.HttpJspPage</code>
pageContext	<code>javax.servlet.jsp.pageContext</code>
out	<code>javax.servlet.jsp.JspWriter</code>
config	<code>javax.servlet.http.servletConfig</code>
exception	<code>java.lang.throwable</code>

2.7.1 request Object

The request object is an instance of `HttpServletRequest` interface. This object is used to retrieve the values that the client browser passed to the server during HTTP request such as cookies, headers or parameters associated with the request. The most common use of request object is to access query string values. You can do this by calling the `getParameter()` method. You have already seen the example of `request.getParameter()` method in `<jsp:include>` and `<jsp:forward>`.

For example:

```
<html><body><h2>Example of request Object</h2>
<% // Get the Programme Name from the request Query
    String name=request.getQueryString();
    out.println("Hello: "+name); %>
</body></html>
```

When you run this program, it looks for your name in the query and returns the value, if it is found. Enter the following URL into your browser to see the result.

<http://localhost:8080/BCS053/jsp/reqObject.jsp?poonam>

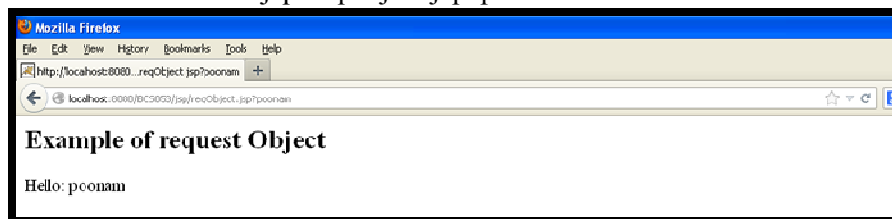


Figure 15: Example of Request Object

In the above program, `getQueryString()` method of request object is used to return the query string from the request.

2.7.2 response Object

As you know, response is a process to responding against it request. Using response object, reply is sent back to the client browser. Through this object, response parameter can be modified or set. This object is used to handles the output of client. The response object is an instance of `HttpServletResponse` interface.

```
<% response.sendRedirect("http://www.ignou.ac.in");%>
```

When the above code is run in Tomcat server, the `sendRedirect()` method of the `javax.servlet.HttpServletResponse` to redirect the user to a different URL. In this case, control transfer to IGNOU website.

2.7.3 session Object

The session object is represented by the `javax.servlet.http.HttpSession` interface. This object is behaves in same way as under the java servlet. For each user, servlet can create an `HttpSession` object that is associated with a particular single user. This object is used to track the user information in a same session. Session tracking allows servlet to maintain

information about a series of request from the same user. This can be done by URL rewriting, cookies, hidden form fields and session.

2.7.4 application Object

The application object is an instance of `javax.servlet.ServletContext`. This object has application scope which means that it is available to all JSP pages until the JSP engine shut down. The `ServletContext` is the environment where the servlet run. The servlet container creates a `ServletContext` object that we can use to access the servlet environment. There is one `ServletContext` object for each web application per Java Virtual Machine.

2.7.5 page Object

It represents the `javax.servlet.jsp.HttpJspPage` interface. This object is reference to the current instance of the JSP page. The page object is a synonym for *this* object and is not useful for programming language.

2.7.6 pageContext Object

The `pageContext` object is an instance of `javax.servlet.jsp.PageContext`. It is used to represent the entire JSP page. The `pageContext` object is used to set, get and remove attribute of the JSP page.

It provides a single point of access to many of the page attribute such as directives information, buffering information, `errorPageURL` and page scope. It is also provides a convenient place to store shared data. This object stores references to the request and response objects for each request. The `PageContext` class defines several fields, including `PAGE_SCOPE`, `REQUEST_SCOPE`, `SESSION_SCOPE`, and `APPLICATION_SCOPE`, which identify the four scopes.

2.7.7 out Object

Out is a very simple and frequently used implicit object in scriptlet. JSP expression is automatically placed in output stream. So, out object is rarely needed to refer in expression. It is an instance of `javax.servlet.jsp.JspWriter` class which extends `java.io.writer`. You call either `print()` or `println()` method to send output to the client.

For example: `<% out.println(" Hello! Out Object"); %>`

2.7.8 config Object

The `config` object is an instance of `javax.servlet.http.servletConfig`. This object is used to get the configuration information of the particular JSP page.

2.7.9 exception Object

This object is available only on pages that are marked as an error page using the page directive `isErrorPage` attribute. This object is assigned to the `Throwable` class which is the super class of all errors and exceptions in the java language. It contains reference to uncaught exception that caused the error page to be invoked.

Check Your Progress 3

1. What are the implicit objects?

2. What is request object? Explain with example.

3. Which implicit object allows the storage of values in an associated hash table?

4. Write a JSP program for odd numbers.

5. Write a JSP program for displaying a Remote address using scriptlets.

2.8 Complete Example

The following program is displayed a JSP login page which accepted login information from the user and count the number of times user visit the page.

Source code for header.html

```
<html><head><title>BCA:BCS-053</title>
<style>h1 {text-align:center}</style>
</head><body><table><tr><td></td><td>
<h1>Indira Gandhi National Open
University</h1></td></tr></table></body></html>
```

Source code for visitCount.java

```
package bean1;
public class visitCount {
int count=0;
public visitCount(){}
public int getCount(){ count++;
return this.count;}
public void setCount(int count)
{this.count=count;}}
```

Source code for footer.jsp

```
<html><head><title>BCA:BCS-053</title></head><body>
<%@ page language="java" %>
<jsp:useBean id="counter" scope="session" class="bean1.visitCount" />
<jsp:setProperty name="counter" property="count" param="count" />
<footer><hr> &copy; 2013 IGNOU</footer>
Visitor Count:
<jsp:getProperty name="counter" property="count" />
</body></html>
```

Source code for main.jsp

```
<html><head><title>BCA:BCS-053</title></head><body>
<%@ include file="header.html" %>
<form action="actionJsp.jsp">
<fieldset ><legend>Login Details</legend>
Username <input type="text" name="uname"/><br>
Password <input type="password" name="pwd"/> <br></fieldset>
<input type="submit" value="submit"> </form><br>
<jsp:include page="footer.jsp" />
</body></html>
```

Source code for actionJsp.jsp

```
<html><head><title>BCA:BCS-053</title></head><body>
<%@ include file="header.html" %>
<% // Get the User Name from the request
    out.println("<b>User Name:</b>" + request.getParameter("uname"));
    out.println("<br>");
    // Get the Password from the request
    out.println("<b>Password:</b>" + request.getParameter("pwd")); %>
</body></html>
```

The outputs of the above program are as follows:



When you will click on the submit button of the above form, the following screen will be displayed with the login information.



The above example contains many small program files such as header.html, visitCount.java, footer.jsp, main.jsp and actionJsp.jsp. The header file is used for displaying a heading for the

program page. In `visitCount.java`, `visitCount` is a Java Bean class that acts as a visitor counter. It has a single `int` type property, `count` that holds the current number of times the beans property has been accessed. This program also contains the methods for getting and setting the count property. In `footer.jsp`, the `visitCount` bean is integrate with JSP page using the `<jsp:useBean>` standard action. Other two action elements `<jsp:setProperty>` and `<jsp:getProperty>` are also defined for the count property. In `main.jsp`, header and footer files are included by using the `include` directive and `include` action. Two input form control is defined for accepting the user's login information. Another file `actionJsp.jsp` is also included in main program. When you will run main program named `main.jsp`, it will show you a form, in which you fill information and submit the form.

2.9 Summary

In this unit, you have covered the basics of Java Server Pages. Now, you have known that the JSP technology is used to create web application. It is an extension of the servlet because it provides more functionality then servlet. Aside from the regular HTML, there are three types of JSP component that you embed in your document i.e. directives, scripting elements and actions. Directives are used to control overall structure of the JSP page. You have learned three types of scripting elements such as declaration, expression, and scriptlets. Some Standard actions are specific to Java Bean and other actions are used to include or forward request, add applets or plugin within JSP page. You have also learned about the objects that are available for use in scriptlet or expression without being declared. Now, you have able to create a web page in JSP and also able to create/access a Java Bean within the JSP.

2.10 Solutions/Answers

Check Your Progress 1

Ans1: JSP page contains JSP technology specific elements as well as static HTML or XML tags. A JSP page has extension `.jsp` which tells the web server that the JSP engine will process elements on this page.

Ans2: Both Servlet and JSP are server side technology but the development process of JSP page is easier than Servlet. The content and display logic is separated in a JSP page and you can insert directly java code inside the JSP page while in servlet, you can write java code with plain html using plenty of `out.println()` statement for front end display of the page which is very tedious work. Another advantage of JSP page over the Servlet is that the JSP page can use custom tags while in Servlet, it is not possible. The power of JSP is to provide a framework for Web application development.

Ans3: The `_jspService()` method is defined in `javax.servlet.jsp.HttpJspPage` interface and it is invoked every time a new request comes to a JSP page. This method takes `HttpServletRequest` and `HttpServletResponse` objects as an arguments. It returns no value. A page author cannot override this method. It is defined automatically by the processor.

Ans4: When a client requests a JSP page, the browser sends a request to web server which is forwarded to the JSP engine. If it is a new request from the client then it translated and compiled by the JSP engine into a servlet. Now, servlet is ready for servicing the client

request and generates response which returns back to the browser via web server. For the second time same request from the client including browser and web server request to JSP engine, the JSP engine determines the request that the JSP-Servlet is up to date, it immediately passes control to the respective JSP-Servlet.

Ans5: Pl. refers to 2.3 section of this unit.

Check Your Progress 2

Ans1: pl. refers to 2.4 sections of this unit.

Ans2: The standard actions are tags that are embedded in a JSP page such as forward, include and many more. It is also called as action elements. You can dynamically insert a file, reuse a Java Bean, forward or include a request to/from the other page. There are three action elements specific to Java Bean i.e. <jsp:useBean>, <jsp:setProperty> and <jsp:getProperty>. For more detail, pl. refers to 2.4.3 section.

Ans3: This include directive is used to insert text and code at the translation time. You can not pass the parameter in this directive. The syntax is: <%@ include file="relativeURL" %>

The include action is used to include static as well as dynamic content in the current JSP page at the request or run time. You can pass the parameter using param action. The syntax is :

```
<jsp:include page="relativeURL"/> or  
<jsp:include page="relativeURL"/> <jsp:param ../></jsp:include>
```

Ans4: The declaration tag is used to declare variables and methods which are placed inside the declaration part of the generated servlet. In JSP page, the declaration tag is start with <%! and end with %>. The code inside this tag must end with semicolon. For example:

```
<html><body><p>Example of declaration tag </p><%! private int i = 4; %>  
<%! private int squire(int i) { i = i * i; return i; }%> <%= squire(i) %> </body></html>
```

Ans5: Source code for <jsp:include> action :

```
<html> <head><title>JSP:Include example</title></head><body>  
  <jsp:include page="faculty.jsp">  
    <jsp:param name="name1" value="Sh. Shashi Bushan" />  
    <jsp:param value="Director" name="desig1"/>  
    <jsp:param name="name2" value="Sh. Akshay Kumar" />  
    <jsp:param value="Associate Professor" name="desig2"/>  
  </jsp:include>  
</body></html>
```

Source code for faculty.jsp file

```
<html><head><title>faculty.jsp</title></head><body>  
  <b><i><% out.print("Name : "+request.getParameter("name1")); %></i></b> <br>  
  <% out.print("He is a "+request.getParameter("desig1")+ " Dept.: SOCIS"); %> <br>  
  <b><i><% out.print("Name : "+request.getParameter("name2")); %></i></b><br>  
  <% out.print("He is a "+request.getParameter("desig2")+ " Dept.: SOCIS "); %>
```

</body></html>

Check Your Progress 3

Ans1: Pl. refers to 2.7 sections.

Ans2: pl. refers to 2.7.1 section.

Ans3: The pageContext object is an instance of javax.servlet.jsp.PageContext. It is used to represent the entire JSP page. The pageContext object is used to set, get and remove attribute of the JSP page. It provides a single point of access to many of the page attribute such as directives information, buffering information, errorPageURL and page scope. It is also provides a convenient place to store shared data.

Ans4: <%@ page language="java"%><html><head><title>Odd number example written in JSP</title></head> <body><p>Odd number are:</p>
 <% for(int i=0;i<=100;i++) {
 if((i%2)!=0) { out.println(i); out.println(""); } }%></body></html>

Ans5:<html><body><% out.println("Remote Address is : " + request.getRemoteAddr());%>
</body></html>