Question 1. What is XSL-FO? List the steps to create a XSL-FO document.

Answer:

XSL also allows us to format XML documents. This formatting aspect of XSL is called XSL Formatting Objects (XSL-FO).

XSL-FO is an XML based markup language and specially designed to format XML data. XSL-FO is essentially a typesetting language. It enables us to easily specify page layouts, including setting margins and line spacing; creating headers, footers, and marginalia; and generating endnotes, footnotes, columnar page content, cover sheets, and tables of content.

XSL-FO document does not apply to a single page. It works on multiple pages. This is where the <fo:simple-pagemaster> declaration comes into picture. It defines the layouts for individual pages in terms of the margins, widths, etc. This is the reason why we need an <fo:layout-master-set> element to enclose all the individual page layout definitions.

We have named this page definition as sample. This allows the page formatting definition to be applied where we want, later in the document. The attributes page-height and page-width specify the length and breadth of the page to be outputted. In the above code margin-top, margin-bottom, margin-left, and margin-right defines the margins of page. <fo:region-body margin-top=“0.90in” /> specifies the text area where the text should be displayed. This indicates that the page body has a top margin of 0.90 inches or header can use space of up to 0.90. Finally the sequence <fo:page-sequence> element specifies the sequence descriptions which is header, footer and then page body.

Steps to create a XSL-FO document:

1. Type <?xml version="1.0"?>.

2. Type <fo:root to define the root element of the XSL-FO document. Notice the fo: namespace prefix; it is defined next.

3. Then, type xmlns:fo="http://www.w3.org/ 1999/XSL/Format" to declare the XSL-FO namespace with the fo: prefix.

4. Finally, type > to close the tag.

5. Type <fo:layout-master-set> to start the overall structure part of the document.

6. Type <fo:simple-page-master master-name="master">, where master is the page template name used in Step 10.

7. Type <fo:region-body/> to declare that this page master will contain content in the body region.

8. Type </fo:simple-page-master>.

9. Type </fo:layout-master-set>.

10. Then, type <fo:page-sequence master-reference="master">, where master is the same name defined in Step 6. This begins the page content part of the document.

11. Type <fo:flow flow-name="xsl-region-body">. Here, xsl-region-body refers to the body region declared in Step 7.

12. Enter the page content for your final out-put using fo:block elements.

13. Type </fo:flow>.

14. Type </fo:page-sequence>.

15. Finally, type </fo:root> to complete the XSL-FO document.

Question 2. What is difference between ordered list and definition list?

Answer:

**Ordered List:**

Ordered list begins with<ol> and ends with </ol> and like unordered lists uses <li> tag to mark the beginning of each list item. Ordered lists are suitable for creating simple outlines or step-by-step instructions because the list items are automatically numbered by the browser.

<ol>

<li>ordered list 1</li>

<li> ordered list 2</li>

. . .

<li> ordered list n</li>

</ol>

In case of ordered list type attribute is used to set ‘a’ for lowercase letters, ‘A’ for uppercase letters, ‘i’ for lowercase roman numerals and ‘I’ for uppercase roman numerals. To start ordering from the letter d, you would use <ol type="a" start="4"> because d is the fourth letter. Like unordered list ordered lists can be nested.

The type attribute of the <ol> tag, defines the type of the list item marker:

type="1": The list items will be numbered with numbers (default)

type="A": The list items will be numbered with uppercase letters

type="a": The list items will be numbered with lowercase letters

type="I": The list items will be numbered with uppercase roman numbers

type="i":The list items will be numbered with lowercase roman numbers

**Definition Lists:**

A definition list is a structure which is used in such case where the author wishes to present a single term or phrases followed by content related to the term. Definition lists are enclosed within <dl> and </dl>. Each term being defined within dl element is indicated by a <dt> element, which is derived from "definition term." Each definition itself is defined by <dd>. Neither the dt nor the dd elements require a close tag under traditional HTML. A generic definition list looks like this:

<dl>

<dt>BCA</dt>

<dd> Bachelor of Computer Application</dd>

<dt>MCA</dt>

<dd> Master of Computer Application</dd>

</dl>

Ordered lists are suitable for creating simple outlines or step-by-step instructions because the list items are automatically numbered by the browser. A definition list is a structure which is used in such case where the author wishes to present a single term or phrases followed by content related to the term

Question 3: Explain comparison between XML and Relational database.

Answer:

XML is the abbreviation for Extensible Markup Language. This is an open and popular standard for marking up text in a way that is both machine and human readable. By “marking up text” we mean that the data in the text files is formatted to include meaningful symbols that communicate to a reader what that data is for. The syntax of XML is similar in style to HTML, the markup language of the World Wide Web (WWW).

A relational database is a powerful data storage and retrieval technology where data is stored as rows in tables and the database has one or more tables. Each row of a table has the same columns as every other row in that table. Data is related between tables using the concept of “foreign keys” so that data in a row of one table can be associated with one or more rows of another table.

The data in an XML file can be organized into hierarchies so that the relationships between data elements are visually obvious. The data and the structure are always presented together. XML is intuitive once a few simple syntax rules are understood. Designing a good XML structure is something a non-technical business subject matter expert can do after minimal instruction.

Beyond simple XML text, there are related technologies such as XML Schemas (XSD) and XML Transformations (XSL). These technologies complement XML text documents by adding value in the areas of validation and processing.

Data in a relational database is readable by executing SQL queries in a management tool to extract and present the data in any number of ways. The extraction requires an understanding of the database structure, including the foreign key relationships. Designing a good non-trivial relational database requires significant training and/or significant experience with relational database design techniques.

Data model: Relational data model, consisting of tabular data entities (tables), with rows and columns. XML possessed Hierarchical data model, composed of document structures with element and attribute nodes.

Data types: A wide variety of data types typically are provided, including support for binary data in RDB. XSD schemas are equipped with a comparable set of data types in XML.

Data element relationships: Column definitions can interrelate within and between tables, according to DDL rules in RDB. References can be explicitly or intrinsically defined between elements in XML.

Question 4. Differentiate between Synchronous vs. Asynchronous Data Retrieval with example.

Answer:

**Synchronous data retrieval** :

As you can see in the below code that all the processing is bundled in the method called getXMLDocument. This method expects an URL of web server as a parameter. After this method is invoked, the getXmlHttpRequestObject() method is called and an instance of XMLHttpRequest is set to the variable xmlHttp. Next, this variable is validated to check if it is null or an instance is associated with it. If this variable holds the instance, the open()method is invoked with three parameters. The purposes of these three parameters are already discussed in above section. In the above JavaScript GET is used as first parameter since we are retrieving an XML file from the server. Second parameter is the URL where location.host property is used. Location is the address of the web server, and host is the port number. Since the third parameter is false, so it signifies a synchronous request. send() method performs the actual request. Note that we’ve specified an alert statement to display a message box that is displayed only after the request is processed and returned from the server. So finally we can say, a synchronous operation that is similar to the normal request/response process in any web application.

**Asynchronous data retrieval**:

The main difference with synchronous data retrieval is in third parameter of open() method. As discussed earlier if the value if set to true, it indicates it as asynchronous mode of retrieval. When it is asynchronous mode of data retrieval it implies that the user need not wait until the script has completed execution on the server. After the state has been changed, it triggers the callback method associated with the on ready statechange event. In the above example, the request to url on the server is done asynchronously, and a callback method called processRequest is set to the onreadystatechange event. The processRequestmethod on the client is executed accordingly. Process request method part is highlighted in the above code. In the above example process state is set to 4, which is readyState property; signifies the state as ‘complete’

|  |  |
| --- | --- |
| **Synchronous data retrieval** | **Asynchronous data retrieval** |
| <script language= “javascript” type=”text/javascript” src=xmlhttp.js></script> <script language= “javascript”> function getXMLDocument(url) { getXmlHttpRequestObject(); if(xmlHttp) {  Advance Web Programming Unit 7  Sikkim Manipal University Page No.: 136  xmlHttp.open(“GET”, url, false); xmlHttp.send(null); alert(“Request / Response handled synchronously”); } } //here is the method call for getXMLDocument getXMLDocument (“http://” + location.host + “AJAX/Items.xml”); </script> | function getXMLDocument(url) { getXmlHttpRequestObject(); if(xmlHttp) { xmlHttp.open(“GET”, url, true); xmlHttp.onreadystatechange = processRequest  xmlHttp.send(null); } } function processRequest(url) { if(xmlHttp.readyState == 4) { //Processing…to be done here........ alert(“Request / Response handled Asynchronously”); } |

Question 5: Explain any five new features of HTML5.

Answer:

1. Semantic tags :

HTML5 introduces a set of new tags that help to define the semantics, or meaning, of your web page content. Some of such tags are <nav>, <heading>, <footer>. <nav> is used for defining sections that include site navigation, and <heading> is used to define the titles and tag lines at the beginning of web pages. These tags make web pages easier to understand by search engines, enabling them to generate more useful search results. The tags also help web browsers to determine the most important parts of web pages; if there is limited room on the screen, web browsers will know what content to feature more prominently.

1. Richer multimedia : Another very interesting feature of HTML5 is a new <video> tag. It helps to embed video clips into web pages and an <audio> tag to mix sound. In the past, developers used the nonstandard <embed> tag and the generic <object> tag for adding multimedia. The <embed> and <object> tags used to pass the duties to third-party plugins for playing the video or sound. As an added advantage using these tags, web developers can create custom player controls, style the media with CSS, and more. Exact use of these tags will be discussed in detail later.
2. Drawing on canvases: Suppose you want to create an drawing area on your web page. This new <canvas> tag enables you to define a rectangular (or any shape) area on your web page. You can use scripting commands to create shapes, draw straight and curved lines, apply color gradients, and even add images or parts of images within the area. In the past, designers needed to create such visual content in a separate image editor or drawing program and then embed the result as an image. With HTML5, you can create drawings using the <canvas> tag and JavaScript.
3. GeoLocation : By using the GeoLocaton feature web page can retrieve the latitude and longitude of the user's computer. Earlier this feature was not so useful when everyone was using the Internet with their desktop PC. There are not many things that we need the road level location accuracy of the user. We can get the rough location by analyzing the IP address. Webkit and other modern mobile browsers are in everyone's pocket. GeoLocaton lets us design mobile applications and games to play with the location.
4. WebGL : WebGL extends the Canvas element by providing a set of 3D graphics API in the web browser. The API follows the standard of OpenGL ES 2.0. The WebGL provides a real 3D rendering place for 3D HTML5 games. However, all browsers don’t support the WebGL, currently Mozilla Firefox 4, Google Chrome, and a nightly build of WebKit browser support it.

Question 6: Explain the following:

a) Window event attributes. b) Form event attributes.

Answer:

Event-handler attributes are used to define actions that should occur when an event occurs in the browser.

**Window event attributes:**

HTML5 added several attributes to the body element that are used to perform an action on the browser window. Below lists all the window event attributes along with brief explanation of each event. All window event attributes have a script as the attribute value. Among above attributes only onblur, onfocus, and onload was supported by HTML4.

**Form events attributes:**

Several actions are associated with HTML forms. Users might add incorrect data, miss required rows, and so on. Few of form attributes we have used already like, onchange, onfocus, oncontextmenu, etc

Below is the list of Window event attributes and Form event attributes.

|  |  |
| --- | --- |
| **Window event attributes** | **Form events attributes** |
| |  |  | | --- | --- | | Attribute | Explanation | | Onafterprint | After the document is printed | | Onbeforeprint | Before the document is printed | | Onbeforeonload | Before the document is loaded | | Onblur | When the window loses focus | | Onerror | When an error occurs | | Onfocus | When the window gets focus | | onhaschange | When the document has changed | | Onload | When the document is being loaded | | onmessage | When the message is activated | | onoffline | When the document goes offline | | onoffline | When the document comes online | | onpagehide | When the window is hidden | | onpageshow | When the window becomes visible | | Onpopstate | When the window history changes | | Onredo | When the last action is repeated | | Onresize | When the window is resized | | Onstorage | When loading | | onstorage | When the last action is undone | | |  |  | | --- | --- | | Attribute | Explanation | | Onblur | An element loses focus | | Onchange | An element changes | | oncontextmenu | A context menu is activated | | Onfocus | An element receives focus | | onformchange | A form changes | | onforminput | A form receives user input | | Oninput | An element receives user input | | Oninvalid | An element is invalid | | Onselect | An element is selected | | onsubmit | A form is submitted | |