CS8651 Internet Programming – 2017Reg

PHP: An introduction to PHP – Using PHP – Variables – Program control – Built-in functions – Form Validation – Regular Expressions – File handling – Cookies – Connecting to Database; **XML:** Basic XML – Document Type Definition – XML Schema DOM and Presenting XML, XML Parsers and Validation, XSL and XSLT Transformation, News Feed (RSS and ATOM).

Internet Programming – UNIT-IV

Q.No **Questions**

1. **Define** PHP. List the features.

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

PHP is an amazing and popular language!

It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
It is deep enough to run the largest social network (Facebook)!
It is also easy enough to be a beginner's first server side language!

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension "`.php`"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

2.

List the rules for creating variables in PHP.

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the `$` sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (`$age` and `$AGE` are two different variables)
- PHP variable names are case-sensitive

3.

Illustrate a PHP program to determine the type of browser that a web client is using.

Display the Browser – PHP Script

The following PHP function can be used to display the browser:

```
<?php

function get_the_browser()
{

if(strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== false)
return 'Internet explorer';
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Trident') !== false)
return 'Internet explorer';
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Firefox') !== false)
return 'Mozilla Firefox';
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Chrome') !== false)
return 'Google Chrome';
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Opera Mini') !== false)
return "Opera Mini";
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Opera') !== false)
return "Opera";
elseif(strpos($_SERVER['HTTP_USER_AGENT'], 'Safari') !== false)
return "Safari";
else
return 'Other';
```

	} **?>** In the above code, we are checking each possible browser that may be and return the browser name. Here we haven’t checked the Mozilla because of most of the browser using this as the user agent string. Below is how to display the browser name on our web page: Echo get_the_browser();
4.	**Name** any four built-in functions in PHP. PHP Reference The PHP reference contains different categories of all PHP functions and constants, along with examples. Array · Calendar · Date · Directory · Error · File system · Filter · FTP · Libxml · Mail · Math Misc · MySQLi · Network · SimpleXML · Stream String · XML Parser · Zip · Timezones
5.	**Infer** when should the super global arrays in PHP be used? Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes. PHP Global Variables - Superglobals

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

Which super global array in PHP would contain a HTML form's POST data?

PHP Superglobal - $_POST

Super global variables are built-in variables that are always available in all scopes.

PHP $_POST

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:

Example

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
```

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

6. **Classify** the difference between echo() & print() functions.

PHP echo and print Statements

With PHP, there are two basic ways to get output: echo and print.

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small:

The PHP echo Statement	**The PHP print Statement**
➢ echo has no return value ➢ echo can take multiple parameters ➢ echo is marginally faster than print.	➢ print has a return value of 1 so it can be used in expressions. ➢ print can take one argument
The echo statement can be used with or without parentheses: echo or echo().	The print statement can be used with or without parentheses: print or print().
Example-1 <?php echo "<h2>PHP is Fun!</h2>"; echo "Hello world! "; echo "I'm about to learn PHP! "; echo "This ", "string ", "was ", "made ", "with multiple parameters."; ?>	Example-1 <?php print "<h2>PHP is Fun!</h2>"; print "Hello world! "; print "I'm about to learn PHP!"; ?>
PHP is Fun! Hello world!	**PHP is Fun!** Hello world!

Example-2 <?php $txt1 = "Learn PHP"; $txt2 = "W3Schools.com"; $x = 5; $y = 4; echo "<h2>" . $txt1 . "</h2>"; echo "Study PHP at " . $txt2 . " "; echo $x + $y; ?>	Example-2 <?php $txt1 = "Learn PHP"; $txt2 = "W3Schools.com"; $x = 5; $y = 4; print "<h2>" . $txt1 . "</h2>"; print "Study PHP at " . $txt2 . " "; print $x + $y; ?>
Learn PHP Study PHP at W3Schools.com 9	**Learn PHP** Study PHP at W3Schools.com 9

7.

List any two advantages of XML document.

Using XML to exchange information offers many benefits.

Advantages of XML include the following:

- XML uses human, not computer, language. XML is readable and understandable, even by novices, and no more difficult to code than HTML.
- XML is completely compatible with Java™ and 100% portable. Any application that can process XML can use your information, regardless of platform.
- XML is extendable. Create your own tags, or use tags created by others, that use the natural language of your domain, that have the attributes you need, and that makes sense to you and your users.

8.

Give the difference between DTD and XML schema for defining XML document structure with appropriate examples.

DTD vs XSD

There are many differences between DTD (Document Type Definition) and XSD (XML Schema Definition). In short, DTD provides less control on XML structure whereas XSD (XML schema) provides more control.

The important differences are given below:

No.	**DTD**(Document Type Definition)	**XSD**(XML Schema Definition)
1)	DTD stands for **Document Type Definition.**	XSD stands for XML Schema Definition.
2)	DTDs are derived from **SGML** syntax.	XSDs are written in XML.

3)	DTD **doesn't support datatypes**.	XSD **supports datatypes** for elements and attributes.
4)	DTD **doesn't support namespace**.	XSD **supports namespace**.
5)	DTD **doesn't define order** for child elements.	XSD **defines order** for child elements.
6)	DTD is **not extensible**.	XSD is **extensible**.
7)	DTD is **not simple to learn**.	XSD is **simple to learn** because you don't need to learn new language.
8)	DTD provides **less control** on XML structure.	XSD provides **more control** on XML structure.

9.

Analyze about Query String in PHP.

Query string

The information can be sent across the web pages. This information is called query string. This query string can be passed from one page to another by appending it to the address of the page. You can pass more than one query string by inserting the & sign between the query strings. A query string can contain two things: the query string ID and its value. The query string passed across the web pages is stored in $_REQUEST, $_GET, or $_POST variable.

Query string handling in PHP

Query strings

To access the data in a query string you can use the *$_GET* global array. Each element in this array has a key which is the name of the query string variable and a value which is the value of that variable.

```
<a href="mypage.php?variable1=value1&variable2=value2">my link</a>
```

This link loads the page mypage.php with two variables *variable1* and *variable2* with values value1 and value2 respectively.

```
echo $_GET['variable1'];
echo $_GET['variable2'];
// outputs:
//value1
//value2
```

Form data

The get method of forms sends the data to a page via a query string.

```
<form name="form1" id="form1" method="get" action="">
  <input name="textbox" id="textbox" type="text" value="value1" />
  <input name="textbox2" id="textbox2" type="text" value="value2" />
  <input type="submit" name="submitbutton" id="submitbutton" value="Submit" />
</form>
```

This form passes the value of the two text boxes to the page myform.php.

```
print_r($_GET);
// outputs:
// Array (
//   [textbox] => value1
//   [textbox2] => value2
//   [submitbutton] => Submit
// )

echo $_GET['textbox'];
//outputs: value1
```

10. **Show** an example for XML namespace.

A **Namespace** is a set of unique names. Namespace is a mechanisms by which element and attribute name can be assigned to a group. The Namespace is identified by URI(Uniform Resource Identifiers).

Namespace Declaration

A Namespace is declared using reserved attributes. Such an attribute name must either be **xmlns** or begin with **xmlns:** shown as below −

```
<element xmlns:name = "URL">
```

Syntax

- The Namespace starts with the keyword **xmlns**.
- The word **name** is the Namespace prefix.
- The **URL** is the Namespace identifier.

Example

Namespace affects only a limited area in the document. An element containing the declaration and all of its descendants are in the scope of the Namespace. Following is a simple example of XML Namespace −

```
<?xml version = "1.0" encoding = "UTF-8"?>
<cont:contact xmlns:cont = "www.tutorialspoint.com/profile">
   <cont:name>Tanmay Patil</cont:name>
   <cont:company>TutorialsPoint</cont:company>
   <cont:phone>(011) 123-4567</cont:phone>
</cont:contact>
```

Here, the Namespace prefix is **cont**, and the Namespace identifier (URI) as *www.tutorialspoint.com/profile*. This means, the element names and attribute names with the **cont** prefix (including the contact element), all belong to the *www.tutorialspoint.com/profile* namespace.

11. **Define** XML parse tree.

An XML document is always descriptive. The tree structure is often referred to as **XML Tree** and plays an important role to describe any XML document easily.

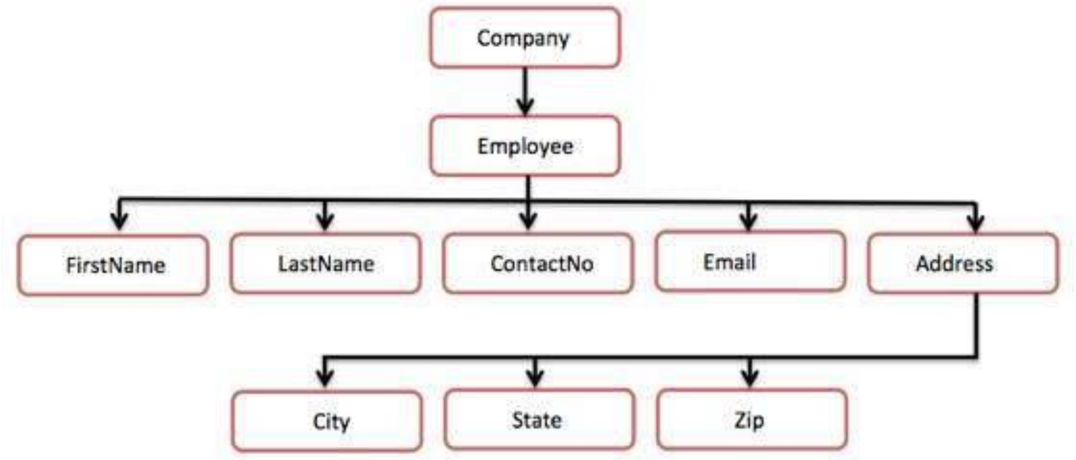
The tree structure contains root (parent) elements, child elements and so on. By using tree structure, you can get to know all succeeding branches and sub-branches starting from the root. The parsing starts at the root, then moves down the first branch to an element, take the first branch from there, and so on to the leaf nodes.

Example

Following example demonstrates simple XML tree structure −

```
<?xml version = "1.0"?>
<Company>
   <Employee>
      <FirstName>Tanmay</FirstName>
      <LastName>Patil</LastName>
      <ContactNo>1234567890</ContactNo>
      <Email>tanmaypatil@xyz.com</Email>
      <Address>
         <City>Bangalore</City>
         <State>Karnataka</State>
         <Zip>560212</Zip>
      </Address>
   </Employee>
</Company>
```

Following tree structure represents the above XML document −



In the above diagram, there is a root element named as <company>. Inside that, there is one more element <Employee>. Inside the employee element, there are five branches named <FirstName>, <LastName>, <ContactNo>, <Email>, and <Address>. Inside the <Address> element, there are three sub-branches, named <City> <State> and <Zip>.

12.

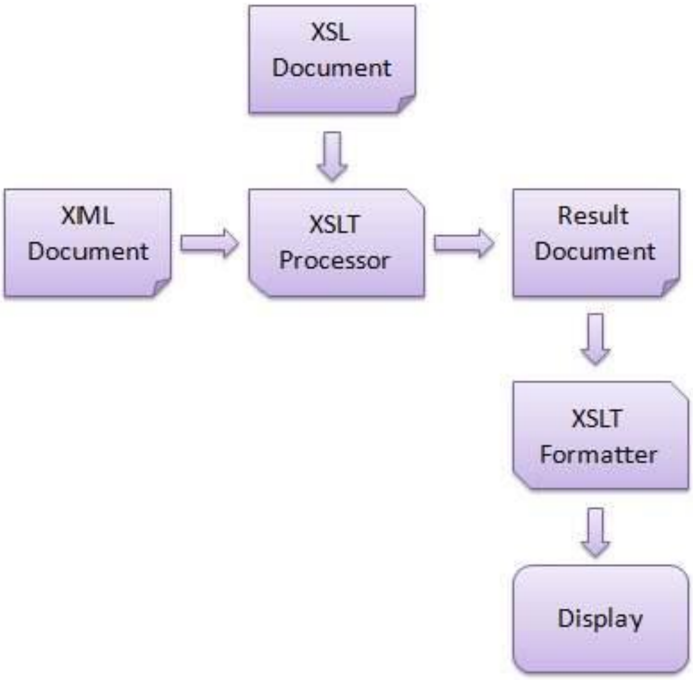
Identify why XSLT is an important tool for development of web applications.

What is XSLT

XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically.

How XSLT Works

An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.



Advantages

Here are the advantages of using XSLT −

- Independent of programming. Transformations are written in a separate xsl file which is again an XML document.
- Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

13.

Assess the data types in XML schema.

You can define XML schema elements in the following ways −

Simple Type

Simple type element is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example −

```
<xs:element name = "phone_number" type = "xs:int" />
```

Complex Type

A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example −

```
<xs:element name = "Address">
   <xs:complexType>
      <xs:sequence>
         <xs:element name = "name" type = "xs:string" />
         <xs:element name = "company" type = "xs:string" />
         <xs:element name = "phone" type = "xs:int" />
      </xs:sequence>
```

```
    </xs:complexType>
</xs:element>
```

14.

Explain DTD for XML Schemas.

A **document type definition** (**DTD**) is a set of *markup declarations* that define a *document type* for a SGML-family markup language (GML, SGML, XML, HTML).

A DTD defines the valid building blocks of an XML document. It defines the document structure with a list of validated elements and attributes. A DTD can be declared inline inside an XML document, or as an external reference.

XML DTD schema example

An example of a very simple external XML DTD to describe the schema of a list of persons might consist of:

```
<!ELEMENT people_list (person)*>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT socialsecuritynumber (#PCDATA)>
```

Taking this line by line:

1. people_list is a valid element name, and an instance of such an element contains any number of person elements. The * denotes there can be 0 or more person elements within the people_list element.
2. person is a valid element name, and an instance of such an element contains one element named name, followed by one named birthdate (optional), then gender (also optional) and socialsecuritynumber (also optional). The ? indicates that an element is optional. The reference to the name element name has no ?, so a person element *must* contain a name element.
3. name is a valid element name, and an instance of such an element contains "parsed character data" (#PCDATA).
4. birthdate is a valid element name, and an instance of such an element contains parsed character data.
5. gender is a valid element name, and an instance of such an element contains parsed character data.
6. socialsecuritynumber is a valid element name, and an instance of such an element contains parsed character data.

An example of an XML file that uses and conforms to this DTD follows. The DTD is referenced here as an external subset, via the SYSTEM specifier and a URI. It assumes that we can identify the DTD with the relative URI reference "example.dtd"; the "people_list" after "!DOCTYPE" tells us that the root tags, or the first element defined in the DTD, is called "people_list":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
  <person>
    <name>Fred Bloggs</name>
    <birthdate>2008-11-27</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

15.

Evaluate the process of displaying XML document in browser.

Display an XML Document in a Web Browser

Displaying XML Using CSS

XML stands for **E**xtensible **M**arkup **L**anguage. It is a dynamic markup language. It is used to transform data from one form to another form.

An XML file can be displayed using two ways. These are as follows :-

1. Cascading Style Sheet
2. Extensible Stylesheet Language Transformation

Displaying XML file using CSS :

CSS can be used to display the contents of the XML document in a clear and precise manner. It gives the design and style to whole XML document.

- **Basic steps in defining a CSS style sheet for XML :**
  For defining the style rules for the XML document, the following things shoulde be done :-

  1. Define the style rules for the text elements such as font-size, color, font-weight, etc.
  2. Define each element either as a block, inline or list element, using the display property of CSS.
  3. Identify the titles and bold them.
- **Linking XML with CSS :**
  In order to display the XML file using CSS, link XML file with CSS. Below is the syntax for linking the XML file with CSS:
  <?xml-stylesheet type="text/css" href="name_of_css_file.css"?>
- **Example 1.**
  In this example, the XML file is created that contains the information about five books and displaying the XML file using CSS.

  XML file :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="Rule.css"?>
<books>
```

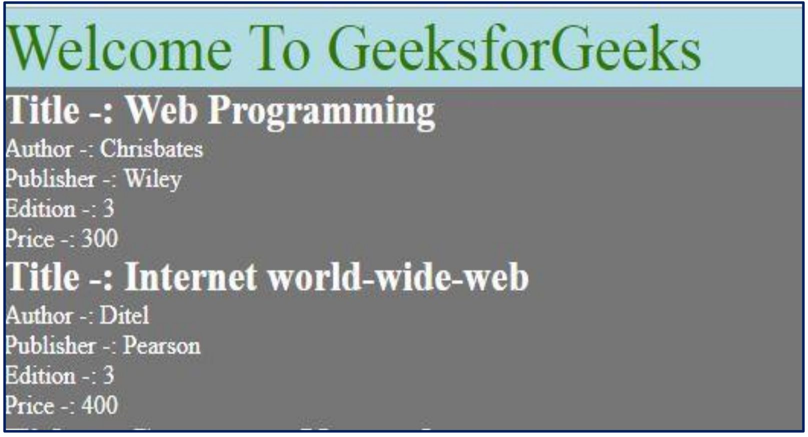
```
  <heading>Welcome To GeeksforGeeks </heading>
  <book>
     <title>Title -: Web Programming</title>
     <author>Author -: Chrisbates</author>
     <publisher>Publisher -: Wiley</publisher>
     <edition>Edition -: 3</edition>
     <price> Price -: 300</price>
  </book>
  <book>
     <title>Title -: Internet world-wide-web</title>
     <author>Author -: Ditel</author>
     <publisher>Publisher -: Pearson</publisher>
     <edition>Edition -: 3</edition>
     <price>Price -: 400</price>
  </book>

  </books>
```

In the above example, Books.xml is linked with Rule.css which contains the corresponding style sheet rules.

CSS FILE :

```
books {
    color: white;
    background-color : gray;
    width: 100%;
}
 heading {
    color: green;
    font-size : 40px;
    background-color : powderblue;
}
 heading, title, author, publisher, edition, price {
    display : block;
}
 title {
    font-size : 25px;
    font-weight : bold;
}
```

- **Output :**



```
Welcome To GeeksforGeeks
Title -: Web Programming
Author -: Chrisbates
Publisher -: Wiley
Edition -: 3
Price -: 300
Title -: Internet world-wide-web
Author -: Ditel
Publisher -: Pearson
Edition -: 3
Price -: 400
```

16. **Summarize** about the need for Namespace in XML.

XML namespaces are used for providing uniquely named elements and attributes in an XML document. They are defined in a W3C recommendation.[1][2] An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a namespace, the ambiguity between identically named elements or attributes can be resolved.

How to get rid of name conflict?

1) By Using a Prefix

You can easily avoid the XML namespace by using a name prefix.

```
<h:table>
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
.  <f:length>120</f:length>
.</f:table>
```

Note: In this example, you will get no conflict because both the tables have specific names.

2) By Using xmlns Attribute

You can use xmlns attribute to define namespace with the following syntax:

```
<element xmlns:name = "URL">
```

Let's see the example:

```
<root>
<h:table xmlns:h="http://www.abc.com/TR/html4/">
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.xyz.com/furniture">
.  <f:name>Computer table</f:name>
.  <f:width>80</f:width>
.  <f:length>120</f:length>
. </f:table>
. </root>
```

In the above example, the <table> element defines a namespace and when a namespace is defined for an element, the child elements with the same prefixes are associated with the same namespace.

17.

Analyze on ATOM in RSS.

What is Atom 1.0 ?

Atom is the name of an XML-based Web content and metadata syndication format, and an application-level protocol for publishing and editing Web resources belonging to periodically updated websites.

Atom is a relatively recent spec and is much more robust and feature-rich than RSS. For instance, where RSS requires descriptive fields such as title and link only in item breakdowns, Atom requires these things for both items and the full Feed.

All Atom Feeds must be well-formed XML documents, and are identified with the *application/atom+xml* media type.

Structure of an Atom 1.0 Feed

A Feed consists of some metadata, followed by any number of entries. Here is a basic structure of an Atom 1.0 Feed.

```
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
   <title>...</title>
   <link>...</link>
   <updated>...</updated>

   <author>
      <name>...</name>
   </author>

   <id>...</id>

   <entry>
      <title>...</title>
      <link>...</link>
      <id>...</id>

      <updated>...</updated>
      <summary>...</summary>
   </entry>

</feed>
```

Atom 1.0 Feed Tags

An Atom 1.0 Feed Document will be constructed of the following two elements:

- <feed> Elements
- <entry> Elements

18. **Summarize** the advantage of RSS documents?

RSS - Advantages

RSS is taking off so quickly because people are liking it. RSS is easy to use and it has advantages for a publisher as well as for a subscriber. Here we have listed out a few advantages of RSS for subscribers as well as for publishers.

Advantages for Subscribers

RSS subscribers are the people who subscribe to read a published Feed. Here are some of the advantages of RSS Feeds for subscribers:

- **All news at one place:** You can subscribe to multiple news groups and then you can customize your reader to have all the news on a single page. It will save you a lot of time.
- **News when you want it:** Rather than waiting for an e-mail, you go to your RSS reader when you want to read a news. Furthermore, RSS Feeds display more quickly than information on web-sites, and you can read them offline if you prefer.
- **Get the news you want:** RSS Feed comes in the form of headlines and a brief description so that you can easily scan the headlines and click only those stories that interest you.
- **Freedom from e-mail overload:** You are not going to get any email for any news or blog update. You just go to your reader and you will find updated news or blog automatically whenever there is a change on the RSS server.
- **Easy republishing:** You may be both a subscriber and a publisher. For example, you may have a web-site that collects news from various other sites and then republishes it. RSS allows you to easily capture that news and display it on your site.

Advantages for Publishers

RSS publishers are the people who publish their content through RSS feed. We would suggest you to use RSS:

- if you want to get your message out and easily,
- if you want people to see what you publish, and
- if you want your news to bring people back to your site.

Here are some of the advantages of RSS if you publish on the Web:

- **Easier publishing:** RSS is really simple publishing. You don't have to maintain a database of subscribers to send your information to them, instead they will access your Feed using a reader and will get updated content automatically.
- **A simpler writing process:** If you have a new content on your web site, you only need to write an RSS Feed in the form of titles and short descriptions, and link back to your site.
- **An improved relationship with your subscribers:** Because people subscribe from their side, they don't feel as if you are pushing your content on them.
- **The assurance of reaching your subscribers:** RSS is not subject to spam filters, your subscribers get the Feeds, which they subscribe to and nothing more.
- **Links back to your site:** RSS Feeds always include links back to a website. It directs a lot of traffic towards your website.
- **Relevance and timeliness:** Your subscribers always have the latest information from your site.

19. **Rewrite** the declaration for elements in XML.

20. How would you **prepare** the steps to get the RSS file on web?

Uploading an RSS Feed

Here are the simple steps to put your RSS Feed on the web.

- First decide which version of RSS Feed you are going to use for your site. We would recommend you to use the latest version available.
- Create your RSS Feed in a text file with extension either .xml or .rdf. Upload this file on your web server.
- You should validate your RSS Feed before making it live. Check the next chapter on RSS Feed Validation.
- Create a link on your Web Pages for the RSS Feed file. You will use a small yellow button for the link that says either RSS or XML.

Now, your RSS Feed is online and people can start using it. But there are ways to promote your RSS Feed so that more number of people can use your RSS Feed.

Promote Your RSS Feed

- Submit your RSS Feed to the RSS Feed Directories. There are many directories available on the web, where you can register your Feed. Some of them are given here:
    - Syndic8: Over 300,000 Feeds listed.
    - Daypop: Over 50,000 feeds listed.
    - Newsisfree: Over 18,000 Feeds.
- Register your Feed with the major search engines. Similar to your web pages, you can add your Feed as well with the following major search engines.
    - Yahoo - http://publisher.yahoo.com/promote.php

- Google - http://www.google.com/webmasters/add.html
- Bing - http://www.bing.com/toolbox/submit-site-url

Keeping Up-To-Date Feed

As we have explained earlier, RSS Feed makes sense for the site which are changing their content very frequently, for example, any news or blogging sites.

So now, you have got RSS Feed buttons from Google, Yahoo, and MSN. You must make sure to update your content frequently and that your RSS Feed is constantly available.

PART-B

1. (i) **Describe** about the introduction and installation of PHP.

Introduction to PHP

PHP is one of the most widely used server side scripting language for web development. Popular websites like Facebook, Yahoo, Wikipedia etc are developed using PHP.

PHP is so popular because it's very simple to learn, code and deploy on server, hence it has been the first choice for beginners since decades.

Uses of PHP

To further fortify your trust in PHP, here are a few applications of this amazing scripting language:

1. It can be used to **create Web applications** like Social Networks(Facebook, Digg), Blogs(Wordpress, Joomla), eCommerce websites(OpenCart, Magento etc.) etc.
2. **Common Line Scripting**. You can write PHP scripts to perform different operations on any machine, all you need is a PHP parser for this.
3. **Create Facebook applications** and easily integrate Facebook plugins in your website, using Facebook's PHP SDK. Check this link for more information.
4. **Sending Emails** or building email applications because PHP provides with a robust email sending function.
5. Wordpress is one of the most used blogging(CMS) platform in the World, and if you know PHP, you can try a hand in **Wordpress plugin development**.

Manual Installation

Step 1: Download the files. Download the latest **PHP** 5 ZIP package from www.**php**.net/downloads.**php**. ...

Step 2: Extract the files. ...

Step 3: Configure **php**. ...

Step 4: Add C:\php to the path environment variable. ...

Step 5: Configure **PHP** as an Apache module. ...

Step 6: Test a **PHP** file.

(ii) **Design** simple calculator using PHP.

Calculator.php

```
<!DOCTYPE html>
<html>
        <head>
                <title>Simple Calculator In PHP | Webdevtrick.com</title>
                <meta charset="utf-8">
                <meta http-equiv="X-UA-Compatible" content="IE=edge">
                <meta name="viewport" content="width=device-width, initial-scale=1">

                <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" rel="stylesheet">
        </head>
        <body>

                <div class="container" style="margin-top: 50px">

                <?php

                // If the submit button has been pressed
                if(isset($_POST['submit']))
                {
                // Check number values
                if(is_numeric($_POST['number1']) && is_numeric($_POST['number2']))
                {
                // Calculate total
                if($_POST['operation'] == 'plus')
                {
                $total = $_POST['number1'] + $_POST['number2'];
                }
                if($_POST['operation'] == 'minus')
                {
                $total = $_POST['number1'] - $_POST['number2'];
                }
                if($_POST['operation'] == 'multiply')
                {
                $total = $_POST['number1'] * $_POST['number2'];
```

```
            }
            if($_POST['operation'] == 'divided by')
            {
            $total = $_POST['number1'] / $_POST['number2'];
            }

            // Print total to the browser
            echo "<h1>{$_POST['number1']} {$_POST['operation']}
{$_POST['number2']} equals {$total}</h1>";

            } else {

            // Print error message to the browser
            echo 'Numeric values are required';

            }
            }
            // end PHP. Code by webdevtrick.com
            ?>

        <!-- Calculator form by webdevtrick.com -->
        <form method="post" action="calculator.php">
          <input name="number1" type="text" class="form-control" style="width:
150px; display: inline" />
          <select name="operation">
            <option value="plus">Plus</option>
            <option value="minus">Minus</option>
            <option value="multiply">Multiply</option>
            <option value="divided by">Devide</option>
          </select>
          <input name="number2" type="text" class="form-control" style="width:
150px; display: inline" />
          <input name="submit" type="submit" value="Calculate" class="btn btn-
primary" />
        </form>

            </div>

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

2.

Explain about control statements and data types in PHP with example.

Control Statements in PHP with Examples

Like any other languages, PHP is built out of a series of control statements. The control statement can be an assignment, a function call, a loop, a conditional statement or even a statement that does nothing or an empty statement.

In PHP we have the following conditional statements:

if statement – We use this control statement to execute some code only if a specified condition is true.

if…else statement – We use this control statement to execute some code if a condition is true and another code if the condition is false.

if…elseif….else statement – We use this control statement to select one of several blocks of code to be executed

switch statement – We use this control statement to select one of many blocks of code to be executed

1. The if Statement

Use the if statement to execute some code only if a specified condition is true.
The expression is evaluated to its Boolean value. If expression evaluates to TRUE, PHP will execute statement, and if it evaluates to FALSE – it'll ignore it

Syntax

```
if (condition) {
code to be executed if condition is true;
}
```

The following example would display " A is bigger than B" if $a is bigger than $b:

```
<?php
if ($a > $b)
echo "A is bigger than B";
?>
```

2. The if…else Statement

elseif, as its name suggests, is a combination of if and else. Like else, it extends an if statement to execute a different statement in case the original if expression evaluates to FALSE. However, unlike else, it will execute that alternative expression only if the elseif conditional expression evaluates to TRUE.

```
if (condition)
code to be executed if condition is true;
else
code to be executed if condition is false;
```

For example, the following code would display a is bigger than b, a equal to b or a is smaller than b:

```
<?php
if ($a > $b) {
echo "a is bigger than b";
} elseif ($a == $b) {
echo "a is equal to b";
} else {
echo "a is smaller than b";
}
?>
```

3. The if…elseif….else Statement

4. The Switch Statement

The switch statement is similar to IF statements on the same expression. In many occasions,

Use the if….elseif…else statement to select one of several blocks of code to be executed.
if (condition)
code to be executed if condition is true;
elseif (condition)
code to be executed if condition is true;
else
code to be executed if condition is false;
Note: Note that elseif and else if will only be considered exactly the same when using curly brackets as in the above example. When using a colon to define your if/elseif conditions, you must not separate else if into two words, or PHP will fail with a parse error.
	you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the switch statement is for.
	switch ( )
	{
	case condition1
	break;
	case condition2
	break;
	}
	For example, the following code would display $i matched value as 0 or 1 or 2:
	<?php
	switch ($i) {
	case 0:
	echo "i equals 0";
	case 1:
	echo "i equals 1";
	case 2:
	echo "i equals 2";
	}
	?>

3. (i) **Create** an XML document that marks up various sports and their descriptions. Use XSLT to tabulate neatly the elements and attributes of the document.

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

-<events league="WC Falun" tournament="2010/2011" template="World Cup" sport="Cross Country Skiing" ut="2012-09-05" id="821135">


-<event id="866683" status="Finished" round="8001 - 1/1 (Final)" date="2011-03-20 13:15:00" name="10 km Freestyle Handicap Pursuit">


-<results participantname="Marit Bjoergen" participantid="43427">

<result id="9498426" value="1" type="rank"/>

<result id="9498424" value="27:58.0" type="duration"/>
```

```
<result id="9505038" value="200" type="points"/>

<result id="9498425" value="" type="comment"/>

<result id="9497448" value="1" type="startnumber"/>

</results>


-<results participantname="Justyna Kowalczyk" participantid="43775">

<result id="9498429" value="2" type="rank"/>

<result id="9498427" value="+1:58.0" type="duration"/>

<result id="9505039" value="160" type="points"/>

<result id="9498428" value="" type="comment"/>

<result id="9497454" value="2" type="startnumber"/>

</results>
</event></events>
```

(ii) **Illustrate** a JSP page that enables the user to input the first name and in response outputs the last name.

POST Method Example Using Form

Below is the **main.jsp** JSP program to handle the input given by web browser using the GET or the POST methods.

```
<html>
  <head>
    <title>Using GET and POST Method to Read Form Data</title>
  </head>

  <body>
    <center>
    <h1>Using POST Method to Read Form Data</h1>

    <ul>
      <li><p><b>First Name:</b>
        <%= request.getParameter("first_name")%>
      </p></li>
      <li><p><b>Last  Name:</b>
        <%= request.getParameter("last_name")%>
      </p></li>
    </ul>
  </body>
</html>
```

Following is the content of the **Hello.htm** file −

```
<html>
  <body>

    <form action = "main.jsp" method = "POST">
      First Name: <input type = "text" name = "first_name">
      <br />
      Last Name: <input type = "text" name = "last_name" />
      <input type = "submit" value = "Submit" />
    </form>

  </body>
</html>
```

Let us now keep **main.jsp** and hello.htm in **<Tomcat-installationdirectory>/webapps/ROOT directory**. When you access ***http://localhost:8080/Hello.htm***, you will receive the following output.

First Name:

Last Name:

Try to enter the First and the Last Name and then click the submit button to see the result on your local machine where tomcat is running.

4. **Create** a webserver based chat application using PHP. The application should provide the following functions Login, Send message (to one or more contacts) and Receive messages (from one or more contacts)

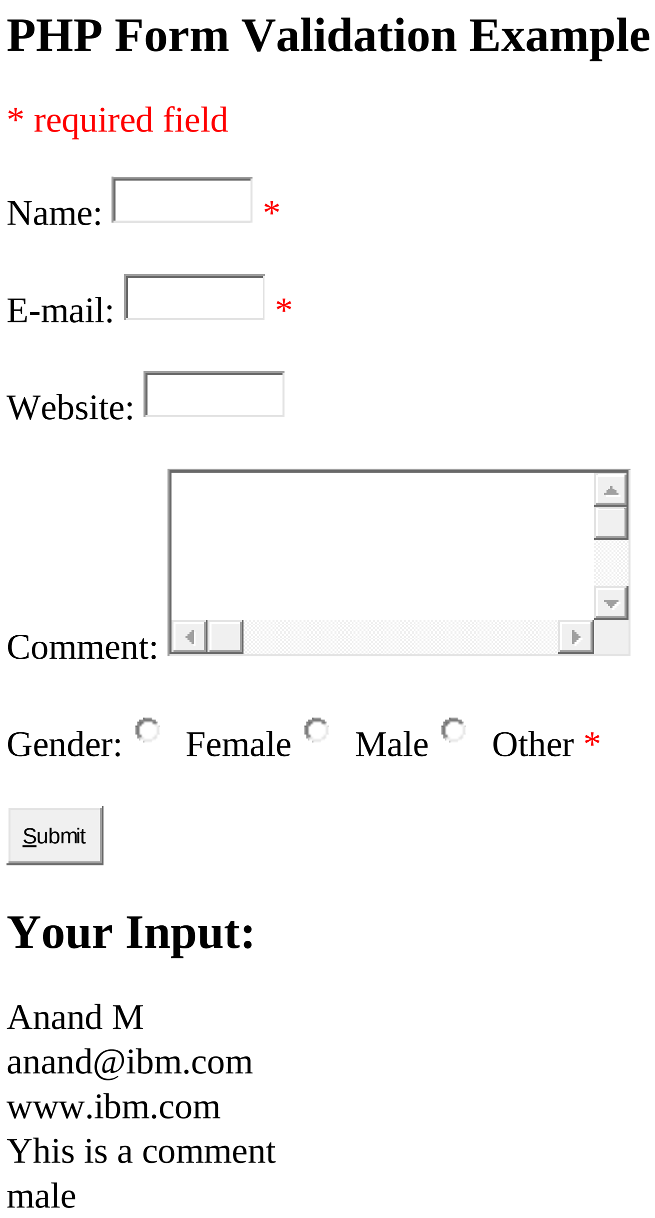
5. (i) **Write** a PHP program that tests whether an email address is input correctly. Test your program with both valid and invalid email addresses.

PHP - Validate Name, E-mail, and URL

Example

```
<?php
// define variables and set to empty values
```

```
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression
also allows dashes in the URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
```

	} ?> OUTPUT  **PHP Form Validation Example** * required field Name: * E-mail: * Website: Comment: Gender: ○ Female ○ Male ○ Other * Submit **Your Input:** Anand M anand@ibm.com www.ibm.com Yhis is a comment male
6.	**Identify** and explain about database connectivity illustrate PHP connectivity with any of the databases. *METHOD FOR: CONNECTING TO MYSQL USING MYSQL* The MySQL Improved extension uses the *mysqli* class, which replaces the set of legacy MySQL functions.

To connect to MySQL using the MySQL Improved extension, follow these steps:

1. Use the following PHP code to connect to MySQL and select a database. Replace *username* with your username, *password* with your password, and *dbname* with the database name:

```
2.    <?php
3.        $mysqli = new mysqli("localhost", "username", "password", "dbname");
?>
```

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

7.

(i) **Discuss** on methods for using cookies in PHP.

Cookies in PHP

Cookies are used to store the information of a web page in a remote browser, so that when the same user comes back to that page, that information can be retrieved from the browser itself.

Uses of cookie

Cookies are often used to perform following tasks:

- **Session management**: Cookies are widely used to manage user sessions. For example, when you use an online shopping cart, you keep adding items in the cart and finally when you checkout, all of those items are added to the list of items you have purchased. This can be achieved using cookies.
- **User identification**: Once a user visits a webpage, using cookies, that user can be remembered. And later on, depending upon the search/visit pattern of the user,

content which the user likely to be visited are served. A good example of this is 'Retargetting'. A concept used in online marketing, where depending upon the user's choice of content, advertisements of the relevant product, which the user may buy, are served.

- **Tracking / Analytics**: Cookies are used to track the user. Which, in turn, is used to analyze and serve various kind of data of great value, like location, technologies (e.g. browser, OS) form where the user visited, how long (s)he stayed on various pages etc.

How to create a cookie in PHP

PHP has a setcookie() function to send a cookie. We will discuss this function in detail now.

```
setcookie(name, value, expire, path, domain, secure, httponly)
```

setcookie() returns boolean.

Example:

Following example shows how to create a cookie in PHP.

```
<?php

$cookie_value = "w3resource tutorials";

setcookie("w3resource", $cookie_value, time()+3600, "/home/your_usename/",
"example.com", 1, 1);

if (isset($_COOKIE['cookie']))

echo $_COOKIE["w3resource"];

?>
```

(ii) **Give** a note on regular expressions.

8. **Summarize** in detail the XML schema, built in and user defined data types.

What is an XML Schema?

An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

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

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

XSD Simple Elements

XML Schemas define the elements of your XML files.

A simple element is an XML element that contains only text. It cannot contain any other elements or attributes.

The text can be of many different types like boolean, string, date, etc.), or it can be a custom type that you can define yourself.

You can also add restrictions (facets) to a data type in order to limit its content, or you can require the data to match a specific pattern.

The syntax for defining a simple element is:

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

where xxx is the name of the element and yyy is the data type of the element.

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

Example

Simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Here are some XML elements:

```
<lastname>Ronald</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

Simple elements may have a default value which is automatically assigned to the element when no other value is specified as shown below:

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

How to Define a Complex Element

We can define a complex element in an XML Schema two different ways:

1. The "employee" element can be declared directly by naming the element, like this:

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2. The "employee" element can have a type attribute that refers to the name of the complex type to use:

```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

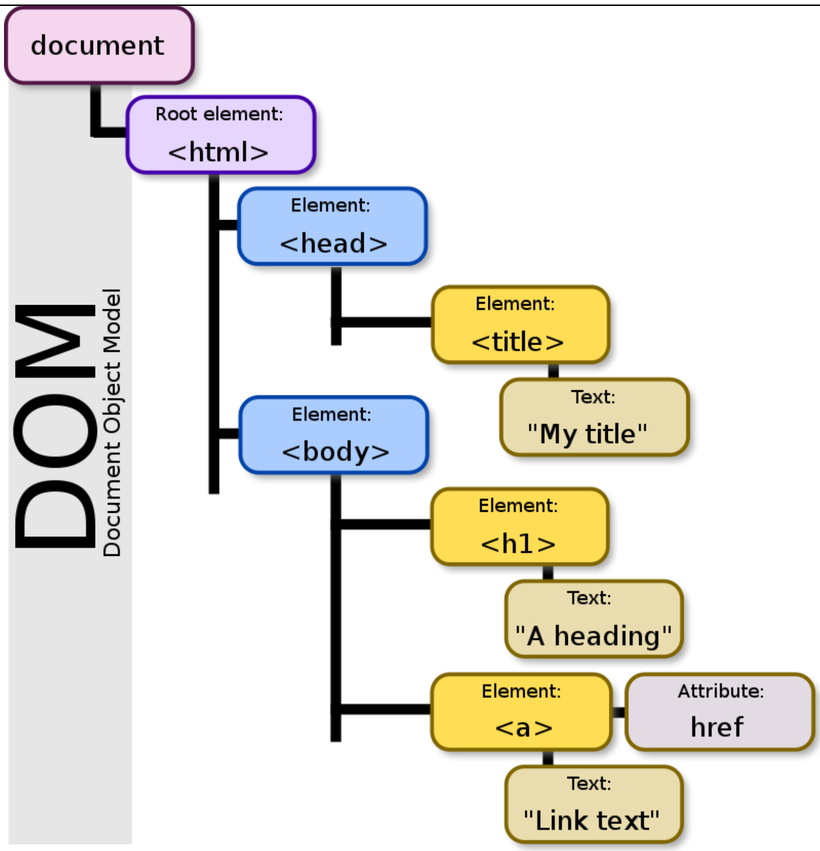
If you use the method described above, several elements can refer to the same complex type, like this:

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

9.

(i) **Demonstrate** the building blocks of DOM.

The **Document Object Model** (**DOM**) is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document. The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document. Nodes can have event handlers attached to them. Once an event is triggered, the event handlers get executed.



(ii) **Classify** the types of DTD.

XML DTD

An XML document with correct syntax is called "Well Formed".

An XML document validated against a DTD is both "Well Formed" and "Valid".

DTD stands for Document Type Definition.

A DTD defines the structure and the legal elements and attributes of an XML document.

A "Valid" XML document is "Well Formed", as well as it conforms to the rules of a DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
```

```
<body>Don't forget me this weekend!</body>
</note>
```

The DOCTYPE declaration above contains a reference to a DTD file. The content of the DTD file is shown and explained below.

The purpose of a DTD is to define the structure and the legal elements and attributes of an XML document:

Note.dtd:

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

The DTD above is interpreted like this:

- !DOCTYPE note - Defines that the root element of the document is note
- !ELEMENT note - Defines that the note element must contain the elements: "to, from, heading, body"
- !ELEMENT to - Defines the to element to be of type "#PCDATA"
- !ELEMENT from - Defines the from element to be of type "#PCDATA"
- !ELEMENT heading - Defines the heading element to be of type "#PCDATA"
- !ELEMENT body - Defines the body element to be of type "#PCDATA"

Tip: #PCDATA means parseable character data.

10.

How do you **infer the significant** differences between DID and XML schema for defining XML document structures with appropriate examples.

Difference Between XML Schema and DTD

XML Schema vs. DTD

DTD, or Document Type Definition, and XML Schema, which is also known as XSD, are two ways of describing the structure and content of an XML document. DTD is the older of the two, and as such, it has limitations that XML Schema has tried to improve.

Summary:

1. XML Schema is namespace aware, while DTD is not.

2. XML Schemas are written in XML, while DTDs are not.

3. XML Schema is strongly typed, while DTD is not.

4. XML Schema has a wealth of derived and built-in data types that are not available in DTD.

5. XML Schema does not allow inline definitions, while DTD does.

11.

(i) **List** out data types data types of XML

XML Schema Data Types

XML Schema data types can be generally categorized a "simple type" (including embedded simple type) and "complex type." The "embedded simple type" is already defined, but can be used to create a new type through restriction or extension.

Table : XML Schema Data Types

Simple Type	User can independently define. This type is used when a restriction is placed on an embedded simple type to create and use a new type.
Complex Type	User can independently define. This type is used when the type has a child element or attribute.

A simple type is a type that only contains text data. This type can be used with element declarations and attribute declarations. On the other hand, a complex data type is a type that has a child element or attribute structure.

●Simple Type Example

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

Here, the section described together with "xs:string" is an embedded simple type according to XML Schema. In this example, we have established the definition that the data type for the element called "Department" is a text string.

●**Complex Type Example**

```
<xs:complexType name="EmployeeType">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Name" />
    <xs:element ref="Department" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Name" type="xs:string" />
<xs:element name="Department" type="xs:string" />
```

(ii) **Explain** about the attributes of XML.

XML Elements vs. Attributes

Take a look at these examples:

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

In the first example gender is an attribute. In the last, gender is an element. Both examples provide the same information.

There are no rules about when to use attributes or when to use elements in XML.

12. **Summarize** on the following

(i) DOM based Parsing.

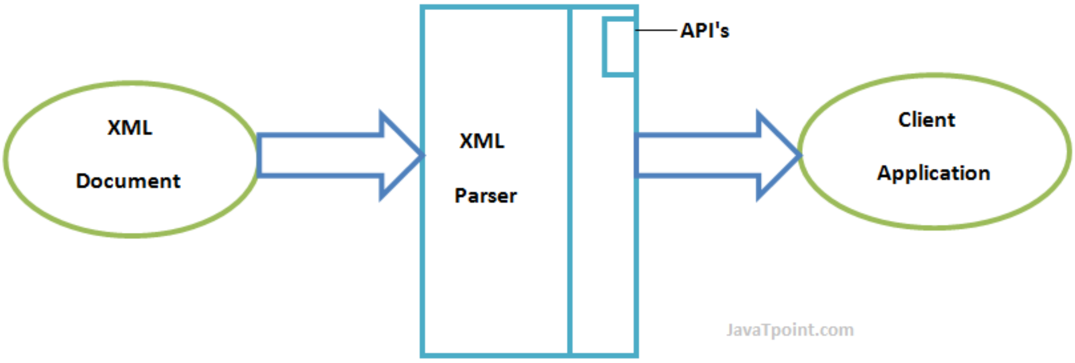
(ii) SAX based Parsing.

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

Advantages

1) It supports both read and write operations and the API is very simple to use.

2) It is preferred when random access to widely separated parts of a document is required.

Disadvantages

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).

2) It is comparatively slower than other parsers.

SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

Features of SAX Parser

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

It is an event based parser, it works like an event handler in Java.

Advantages

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

Disadvantages

1) It is event-based so its API is less intuitive.

2) Clients never know the full information because the data is broken into pieces.

13.

(i) **Compare and contrast** RSS & ATOM.

Difference Between RSS and ATOM

RSS vs ATOM

- Really Simple Syndication or RSS has been the standard for web feeds for a considerable time.
- Web feeds contains either a summary or the full text content of a web page.
- The problem with RSS is the often confusing and non standard conventions used by RSS due in part to its scattered development.
- The advent of the ATOM syndication standard was a response to the design flaws of the RSS standard.
- The primary advantage of the ATOM is its adaptation as the IETF standard.
- Being an IETF standard, each atom feed contains an explicit declaration of the format of the content along with what language is used.
- RSS feeds do not declare its content, but since it only contains plain text or escaped HTML, it is rather easy for the browser to distinguish which is which.

A major flaw of RSS is in its code. RSS code isn't really very usable in other XML vocabularies since it wasn't really intended to do so at the very beginning. ATOM code has been built from the ground with modularity in mind. Therefore, a great majority of its code is reusable even with other XML vocabularies like RSS.

Summary:

1. ATOM is an IETF standard while RSS is not
2. ATOM feeds explicitly indicates the content while the browser is left to figure out whether the RSS feed contains plain text or escaped HTML
3. ATOM code is modular and reusable while RSS code is not
4. RSS still holds dominance in the syndication format due to its head start and popularity

(ii) **Explain** in detail about XSL elements.

XSLT <xsl:element>

Definition and Usage

The <xsl:element> element is used to create an element node in the output document.

Syntax

```
<xsl:element
name="name"
namespace="URI"
use-attribute-sets="namelist">

  <!-- Content:template -->

</xsl:element>
```

Attributes

Attribute	Value	Description
name	name	Required. Specifies the name of the element to be created (the value of the name attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{$country}" />
namespace	URI	Optional. Specifies the namespace URI of the element (the value of the namespace attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{$country}" namespace="{$someuri}"/>
use-attribute-sets	namelist	Optional. A white space separated list of attribute-sets containing attributes to be added to the element

Example 1

Create a "singer" element that contains the value of each artist element:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <xsl:for-each select="catalog/cd">
    <xsl:element name="singer">
      <xsl:value-of select="artist" />
    </xsl:element>
    <br />
  </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

EXAMPLE OUTPUT FILES

XML File

```
<catalog>
<cd>
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<country>USA</country>
<company>Columbia</company>
<price>10.90</price>
<year>1985</year>
</cd>
<cd>
<title>Hide your heart</title>
<artist>Bonnie Tyler</artist>
<country>UK</country>
<company>CBS Records</company>
<price>9.90</price>
<year>1988</year>
</cd>
</catalog>
```

XSL File

```
xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version=
"1.0">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<xsl:for-each select="catalog/cd">
```

```
<xsl:element name="singer">
<xsl:value-of select="artist"/>
</xsl:element>
<br/>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

RESULT

My CD Collection

Bob Dylan
Bonnie Tyler
Dolly Parton
Gary Moore

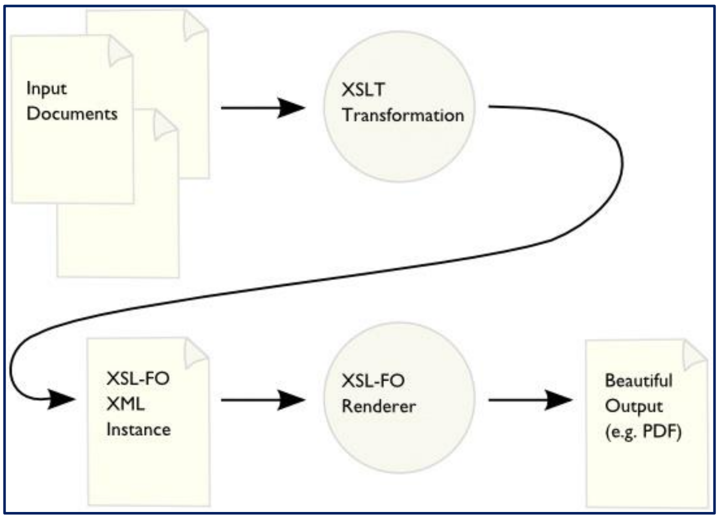
14.

Explain in detail about
(i) XSL and XSLT transformation

XSLT - Transformation

What is XSLT?

XSL Transformations (XSLT 2.0) is a language for transforming XML documents into other XML documents, text documents or HTML documents. You might want to format a chapter of a book using XSL-FO, or you might want to take a database query and format it as HTML.



Wildly Popular

XSLT has become the language of choice for a very wide range of XML applications. It is of course still used to produce XSL-FO documents for printing, but it is also used to integrate back-end software for Web sites. We can find XSLT inside most modern Web browsers, so that XML can be transformed on the fly without the user even noticing; you will find XSLT on the desktop, in servers, in network appliances.

What is XSLT Used For?

If you make a purchase on eBay, or buy a book at Amazon, chances are that pretty much everything you see on every Web page has been processed with XSLT. Use XSLT to process multiple XML documents and to produce any combination of text, HTML and XML output. XSLT support is shipped with all major computer operating systems today, as well as being built in to all major Web browsers.

XSLT – Transformation : STEPS

1) Start with a Raw XML Document
2) Create an XSL Style Sheet
3) Link the XSL Style Sheet to the XML Document

1) Start with a Raw XML Document

We want to **transform** the following XML document ("cdcatalog.xml") into XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.</catalog>
```

2) Create an XSL Style Sheet

Then you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template:

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

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

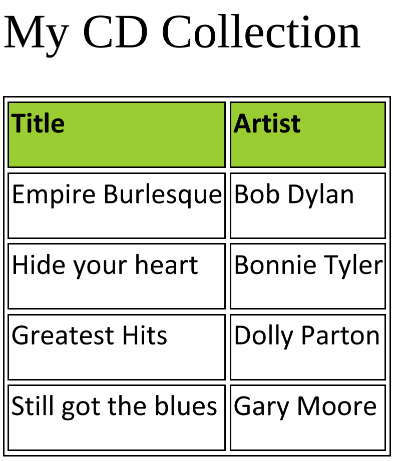
3) Link the XSL Style Sheet to the XML Document

Add the XSL style sheet reference to your XML document ("cdcatalog.xml"):

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>


    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.</catalog>
```

If you have an XSLT compliant browser it will nicely **transform** your XML into XHTML.

Sample OUTPUT



My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore

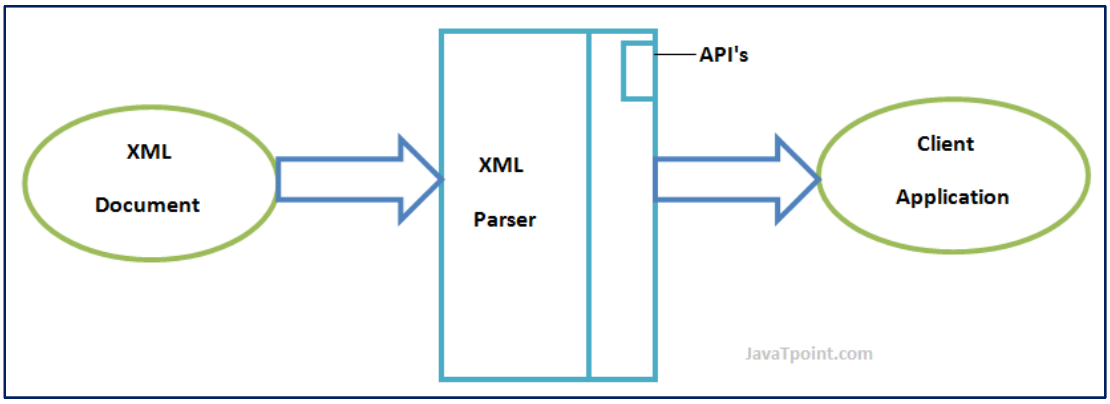
(ii) Comparison of DOM & SAX

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



Types of XML Parsers

These are the two main types of XML Parsers:

3. DOM
4. SAX

DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

Advantages

1) It supports both read and write operations and the API is very simple to use.

2) It is preferred when random access to widely separated parts of a document is required.

Disadvantages

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).

2) It is comparatively slower than other parsers.

SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

Features of SAX Parser

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

It is an event based parser, it works like an event handler in Java.

Advantages

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

Disadvantages

1) It is event-based so its API is less intuitive.

2) Clients never know the full information because the data is broken into pieces.

	PART-C
1.	**Explain** how you shall carry out String Manipulations using a PHP Program.

Manipulating PHP Strings

PHP provides many built-in functions for manipulating strings like calculating the length of a string, find substrings or characters, replacing part of a string with different characters, take a string apart, and many others.

Here are the examples of some of these functions.

Calculating the Length of a String

The `strlen()` function is used to calculate the number of characters inside a string. It also includes the blank spaces inside the string.

Example

Run this code »

```
<?php
$my_str = 'Welcome to Tutorial Republic';

// Outputs: 28
echo strlen($my_str);
?>
```

Counting Number of Words in a String

The `str_word_count()` function counts the number of words in a string.

Example

Run this code »

```
<?php
$my_str = 'The quick brown fox jumps over the lazy dog.';

// Outputs: 9
echo str_word_count($my_str);
?>
```

Replacing Text within Strings

The `str_replace()` replaces all occurrences of the search text within the target string.

Example

Run this code »

```php
<?php
$my_str = 'If the facts do not fit the theory, change the facts.';

// Display replaced string
echo str_replace("facts", "truth", $my_str);
?>
```

The output of the above code will be:

If the truth do not fit the theory, change the truth.

You can optionally pass the fourth argument to the `str_replace()` function to know how many times the string replacements was performed, like this.

Example

Run this code »

```php
<?php
$my_str = 'If the facts do not fit the theory, change the facts.';

// Perform string replacement
str_replace("facts", "truth", $my_str, $count);

// Display number of replacements performed
echo "The text was replaced $count times.";
?>
```

The output of the above code will be:

The text was replaced 2 times.

Reversing a String

The `strrev()` function reverses a string.

Example

Run this code »

```php
<?php
$my_str = 'You can do anything, but not everything.';

// Display reversed string
echo strrev($my_str);
?>
```

The output of the above code will be:

.gnihtyreve ton tub ,gnihtyna od nac uoY

2.

Design a PHP application for College Management System with appropriate built-in functions and database.

3.

Design application to send an email using PHP.

PHP mail() Function

Example

Send a simple email:

```
<?php
// the message
$msg = "First line of text\nSecond line of text";

// use wordwrap() if lines are longer than 70 characters
$msg = wordwrap($msg,70);

// send email
mail("someone@example.com","My subject",$msg);
?>
```

Syntax

```
mail(to,subject,message,headers,parameters);
```

Parameter Values

Parameter	**Description**
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. **Note:** This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. **Windows note:** If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php

	$txt = str_replace("\n.", "\n..", $txt); ?>
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n). **Note:** When sending an email, it must contain a From header. This can be set with this parameter or in the php.ini file.
parameters	Optional. Specifies an additional parameter to the sendmail program (the one defined in the sendmail_path configuration setting). (i.e. this can be used to set the envelope sender address when using sendmail with the -f sendmail option)

Technical Details

Return Value:	Returns the hash value of the *address* parameter, or FALSE on failure. **Note:** Keep in mind that even if the email was accepted for delivery, it does NOT mean the email is actually sent and received!
PHP Version:	4+
PHP Changelog:	PHP 7.2: The headers parameter also accepts an array PHP 5.4: Added header injection protection for the *headers* parameter. PHP 4.3.0: (Windows only) All custom headers (like From, Cc, Bcc and Date) are supported, and are not case-sensitive. PHP 4.2.3: The *parameter* parameter is disabled in safe mode PHP 4.0.5: The *parameter* parameter was added

More Examples

Send an email with extra headers:

```
<?php
$to = "somebody@example.com";
$subject = "My subject";
$txt = "Hello world!";
$headers = "From: webmaster@example.com" . "\r\n" .
"CC: somebodyelse@example.com";

mail($to,$subject,$txt,$headers);
?>
```

Send an HTML email:

```
<?php
$to = "somebody@example.com, somebodyelse@example.com";
$subject = "HTML email";

$message = "
<html>
<head>
<title>HTML email</title>
</head>
<body>
<p>This email contains HTML Tags!</p>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
</tr>
</table>
</body>
</html>
";

// Always set content-type when sending HTML email
$headers = "MIME-Version: 1.0" . "\r\n";
$headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";

// More headers
$headers .= 'From: <webmaster@example.com>' . "\r\n";
$headers .= 'Cc: myboss@example.com' . "\r\n";

mail($to,$subject,$message,$headers);
?>
```

4.

Summarize about XML schema and XML Parsers and Validation.

XML Schema

What is an XML Schema?

An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

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

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

Why Learn XML Schema?

In the XML world, hundreds of standardized XML formats are in daily use.

Many of these XML standards are defined by XML Schemas.

XML Schema is an XML-based (and more powerful) alternative to DTD.

XML Parser

All major browsers have a built-in XML parser to access and manipulate XML.

The XML DOM (Document Object Model) defines the properties and methods for accessing and editing XML.

However, before an XML document can be accessed, it must be loaded into an XML DOM object.

All modern browsers have a built-in XML parser that can convert text into an XML DOM object.

Parsing a Text String

This example parses a text string into an XML DOM object, and extracts the info from it with JavaScript:

Example

```
<html>
<body>

<p id="demo"></p>

<script>
var text, parser, xmlDoc;

text = "<bookstore><book>" +
"<title>Everyday Italian</title>" +
"<author>Giada De Laurentiis</author>" +
"<year>2005</year>" +
"</book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>

</body>
</html>
```

OUTPUT

```
Everyday Italian
```

XML - Validation

Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration(DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are −

- Well-formed XML document
- Valid XML document

Well-formed XML Document

An XML document is said to be **well-formed** if it adheres to the following rules −

- Non DTD XML files must use the predefined character entities for **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)**.
- It must follow the ordering of the tag. i.e., the inner tag must be closed before closing the outer tag.
- Each of its opening tags must have a closing tag or it must be a self ending tag.(<title>....</title> or <title/>).
- It must have only one attribute in a start tag, which needs to be quoted.
- **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)** entities other than these must be declared.

Example

Following is an example of a well-formed XML document −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address
[
  <!ELEMENT address (name,company,phone)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT company (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
]>

<address>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</address>
```

The above example is said to be well-formed as −

- It defines the type of document. Here, the document type is **element** type.
- It includes a root element named as **address**.
- Each of the child elements among name, company and phone is enclosed in its self explanatory tag.
- Order of the tags is maintained.

Valid XML Document

If an XML document is well-formed and has an associated Document Type Declaration (DTD), then it is said to be a valid XML document.