Q.2 a. What is a frameset? Write HTML code of a frameset to create a table of two rows and a single table. (2+4)

Answer: I (3.3) Page No. 64 – 65

A <FRAMESET ...> defines the general layout of a web page that uses frames. <FRAMESET ...> is used in conjunction with <FRAME ...> and <NOFRAMES>. <FRAMESET ...> creates a "table of documents" in which each rectangle (called a "frame") in the table holds a separate document. In its simplest use, <FRAMESET ...> states how many columns and/or rows will be in the "table". You must use either the COLS or the ROWS attributes or both. For example, this code creates a set of frames that is two columns wide and two rows deep:

```
<HTML>
 <HEAD>
 <TITLE>A Basic Example of Frames</TITLE>
</HEAD>


 <FRAMESET ROWS="75%, *" COLS="*, 40%">
 <FRAME SRC="framea.html">
 <FRAME SRC="frameb.html">
 <FRAME SRC="framec.html">
<FRAME SRC="framed.html">
 </FRAMESET>
 </HTML>
```

<FRAMESET ...> itself only define how many rows and columns of frames there will be. <FRAME ...> defines what files will actual go into those frames. <FRAMESET ...> can be nested within another <FRAMESET ...> to create a "table within a table".

By doing this you can create frames that are not strict grids like in the example above. This set of nested framesets creates the popular "title and sidebar" layout.

b. What is the role of HTML Forms? (4)

Answer: I (3.4) Page No.69 – 70

HTML Forms are required when you want to collect some data from the site visitor. For example during user registration you would like to collect information such as name, email address, credit card, etc.

A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.

There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

The HTML **<form>** tag is used to create an HTML form and it has following syntax:

```
<form action="Script URL" method="GET|POST">
  form elements like input, textarea etc.</form>
```

c. How image is embed in HTML page? Discuss the attributes of <img> tag.(2+4)
Answer: I (2.8) Page No. 43 – 45

Images are very important to beautify as well as to depict many complex concepts in simple way on your web page. Image can be inserted in html code using <img> tag.

Insert Image

You can insert any image in your web page by using **<img>** tag. Following is the simple syntax to use this tag.

```
<img src="Image URL" ... attributes-list/>
```

The <img> tag is an empty tag, which means that it can contain only list of attributes and it has no closing tag.

```
<img src="images/test.png" alt="Test Image" />
```

Will display image test.png with text ” Test Image”

The HTML <img> tag also supports following additional attributes:

Attribute	**Value**	**Description**
align	top bottom middle left right	*Deprecated*-Specifies the alignment for the image.
alt	text	Specifies alternate text
border	pixels	*Deprecated* - Specifies the width of the image border.
crossorigin	anonymous use-credentials	It allows images from third-party sites that allow cross-origin access to be reused with canvas.
height	pixels or %	Specifies the height of the image.
hspace	pixels	*Deprecated* - Amount of white space to be inserted to the left and right of the object.

ismap	URL	Defines the image as a server-side image map.
longdesc	text	*Deprecated*-Specifies a URI/URL of a long description - this can elaborate on a shorter description specified with the alt attribute.
src	URL	the url of an image
usemap	#mapname	Defines the image as a client-side image map and used alongwith <map> and <area> tags.
vspace	pixels	*Deprecated* - Amount of white space to be inserted to the top and bottom of the object.
width	pixels or %	Sets the width of an image in pixels or in %.

Q.3 a. Describe the different ways of styles that can be added to a page. (2+6)
Answer: I (4.1) Page No.88

1. **Inline Styles:** Can be defined within the basic HTML tag.

Inline styles are added directly to the element to be styled with the style attribute:

```
<p style="color: red;">
```

2. **Internal Style Sheets:**

Internal style sheets are added to the <head> of a document inside <style> tags:

```
<style type="text/css">
<!--
p { color: red; }
-->
</style>
```

3. **External styles**

Styles can also be set in an external style sheet which is linked to the page with a <link> tag. For example the style sheet for this site is included like this:

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

The advantage of doing this is that the same style sheet can be used in every page on your site. Your entire site can be updated by changing just this one file. The advantage of doing this is that the same style sheet can be used in every page on your site. Your entire site can be updated by changing just this one file.

<style>

Styles can also be placed in the document using the <style> tag. The <style> tag is usually placed in the head section of the document, where it will apply to the whole document.

When using styles that are not inline styles. A style is defined for a particular tag or for a named *class*. The syntax for creating a style for an HTML tag is to write tag { styles } . Note that you do not need " marks around the styles when written this way. It is a good idea to put <!-- and --> comment tags in between the <style> and </style> tags, just to make sure that their contents aren't accidentally displayed in the page

```
<html>
 <head>
  <title>
   Basic styling!
  </title>
  <style>
   <!--
          p { color:#009900;
          font-family:"comic sans ms",sans-serif; }
          h1 { color:#660000; font-size:12pt; }
   -->
  </style>
 </head>

 <body>
  <h1>   Hello World!</h1>
  <p>I am a very basic page.</p>
  Use your back button to
  get out of here.
 </body>
</html>
```

b. Discuss the following CSS terms with examples: **(2×4)**

(i) CSS Border **(ii) Padding Properties**

(iii) Selectors **(iv) CSS media types**

Answer:

(i) CSS Border

Borders can be used for many things, for example as a decorative element or to underline a separation of two things. CSS gives you endless options when using borders in your pages.

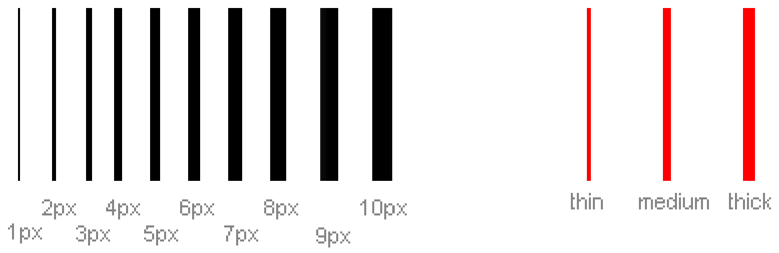
(ii) border-width

(iii)border-color

(iv) border-style

The width of borders [border-width]

The width of borders is defined by the property border-width, which can obtain the values thin, medium, and thick, or a numeric value, indicated in pixels. The figure below illustrates the system:

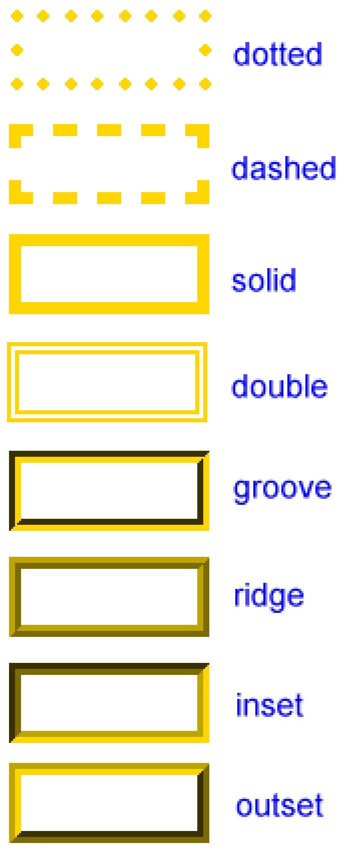


The color of borders [border-color]

The property border-color defines which color the border has. The values are the normal color-values for example "#123456", "rgb(123,123,123)" or "yellow" .

Types of borders [border-style]

The values none or hidden can be used if you do not want any border.



(ii) Padding Properties

The CSS padding properties define the space between the element border and the element content.

Padding

The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element. The top, right, bottom, and left padding can be changed independently using separate properties.

The padding shorthand property sets all the padding properties in one declaration. This property can have from one to four values.Examples:

- **padding:10px 5px 15px 20px;**
  - top padding is 10px
  - right padding is 5px
  - bottom padding is 15px
  - left padding is 20px

- **padding:10px 5px 15px;**
  - top padding is 10px
  - right and left padding are 5px
  - bottom padding is 15px

- **padding:10px 5px;**
  - top and bottom padding are 10px
  - right and left padding are 5px

- **padding:10px;**
  - all four paddings are

(iii) CSS media types – I (5.2.1) Page No. 125

Some CSS properties are designed for a specific type of media. For example the "voice-family" property is designed for aural user agents. Media Types allow you to specify how documents will be presented in different media. The document can be displayed differently on the screen, on the paper, with an aural browser, etc.

The @media Rule
The @media rule allows different style rules for different media in the same style sheet.

The style in the example below tells the browser to display a 14 pixels Verdana font on the screen. But if the page is printed, it will be in a 10 pixels Times font. Notice that the font-weight is set to bold, both on screen and on paper:

```
<html>
 <head>
 <style>
@media screen
{p.test {font-family:verdana,sans-serif;font-size:14px;}
 }
 @media print
{ p.test {font-family:times,serif;font-size:10px;}
}
 @media screen,print
{
p.test {font-weight:bold;}
}
```

```
 </style>
 </head>
 <body> ....
</body>
 </html>
```

Media Types

Media Type	**Description**
all	Used for all media type devices
aural	Used for speech and sound synthesizers
braille	Used for braille tactile feedback devices
embossed	Used for paged braille printers
handheld	Used for small or handheld devices
print	Used for printers
projection	Used for projected presentations, like slides
screen	Used for computer screens
tty	Used for media using a fixed-pitch character grid, like teletypes and terminals
tv	Used for television-type devices

(iv) CSS Selectors

CSS selectors allow you to select and manipulate HTML elements.

CSS selectors are used to "find" (or select) HTML elements based on their id, class, type, attribute, and more.

The element Selector

The element selector selects elements based on the element name.

You can select all <p> elements on a page like this: (all <p> elements will be center-aligned, with a red text color)

Example

```
p {
   text-align: center;
   color: red;
}
```

The id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

An id should be unique within a page, so the id selector is used if you want to select a single, unique element.

To select an element with a specific id, write a hash character, followed by the id of the element.

The style rule below will be applied to the HTML element with id="para1":

Example

```
#para1 {
   text-align: center;
   color: red;
}
```

The class Selector

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period character, followed by the name of the class:

In the example below, all HTML elements with class="center" will be center-aligned:

Example

```
p.center {
   text-align: center;
   color: red;
}
```

In the example below, all <p> elements with class="center" will be center-aligned:

Example

```
p.center {
   text-align: center;
   color: red;
}
```

Grouping Selectors

If you have elements with the same style definitions, like this:

```
h1 {
   text-align: center;
   color: red;
}

h2 {
   text-align: center;
   color: red;
}

p {
   text-align: center;
   color: red;
}
```

you can group the selectors, to minimize the code.

Example

```
h1, h2, p {
   text-align: center;
   color: red;
}
```

Q.4 a. Explain the working of break and switch statements in JavaScripts with examples. (2+2)

Answer: I (6.7) Page No. 171 – 172

Switch statement

The basic syntax of the **switch** statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each **case** against the value of the expression until a match is found. If nothing matches, a **default** condition will be used.

```
switch (expression)
{
  case condition 1: statement(s)
                  break;
  case condition 2: statement(s)
                  break;
   ...
  case condition n: statement(s)
                  break;
  default: statement(s)

}
```

The **break** statements indicate to the interpreter the end of that particular case. If they were omitted, the interpreter would continue executing each statement in each of the following cases.

Example:

```
var grade='A';
document.write("Entering switch block<br />");
switch (grade)
{
  case 'A': document.write("grade A<br />");
          break;
  case 'B': document.write("grade B<br />");
          break;
  case 'C': document.write("grade C<br />");
          break;
  case 'D': document.write("grade D<br />");
          break;
  default:  document.write("no grade<br />")
}
```

This will produce following result:

```
Entering switch block
Grade A
```

***break** can also be used to terminate from loops (for, while and do while)

b. Explain the terms 'Exception Handling' with an appropriate example in Javascript. (2+2)

Answer: I (7.3) Page No. 211 – 213

Exceptions

Runtime errors, also called exceptions, occur during execution. Exceptions affect the thread in which they occur, allowing other JavaScript threads to continue normal execution. JavaScript implements the **try...catch...finally** construct as well as the **throw** operator to handle exceptions.

The **try...catch...finally** block syntax:

```
try {
   tryStatements
}
catch(exception){
   catchStatements
}
finally {
   finallyStatements
}
```

The **try** block must be followed by either exactly one **catch** block or one **finally** block (or one of both). When an exception occurs in the **try** block, the exception is placed in **e** and the **catch** block is executed. The optional **finally** block executes unconditionally after try/catch.

Example:

```
function divide()
{
  var a = 5;
  var b = 0;

  try{
    if ( b == 0 ){
      throw( "Divide by zero error." );
    }else{
      var c = a / b;
    }
  }catch ( e ) {
    alert("Error: " + e );
  }
}
```

c. What do you understand by events in JavaScript? Write a code in JavaScript to display the date when (an event) a button is clicked. (4+4)

Answer:

Events in JavaScript

JavaScript's interaction with HTML is handled through events that occur when the user or browser manipulates a page.

When the page loads, that is an event. When the user clicks a button, that click, too, is an event. Another example of events are like pressing any key, closing window, resizing window etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users.

Events are a part of the Document Object Model (DOM) Level 3 and every HTML element have a certain set of events which can trigger JavaScript Code.

Example

Execute a JavaScript when a button is clicked:

```
<button onclick="myFunction()">Click me</button>
```

Click on a <button> element to display the current day, date and time:

```
<button onclick="getElementById('demo').innerHTML=Date()">What is the
time?</button>
```

Q.5 a. What is a Rollover Button? What are the types and uses of Rollover Buttons? (2+2)

Answer: I (8.6) Page No. 252 – 254

A rollover button is a web button that changes its appearance with the move of your mouse. It contains three states: normal, over and down. Normal state applies to when your mouse is off the button, over state applies to when your mouse rolls over the button and down state applies to when you click on the button. Unlike static web buttons, rollover buttons are dynamic in nature, allowing you to change the color or style of the button with the move of your mouse.

Types of Rollover Buttons Rollover buttons come in different shapes, sizes, colors and styles. Although most rollover buttons are rectangular in shape, they can be customized to the shape of a circle, star or polygon. Since images can be converted to rollover buttons, a rollover button can also contain images. A rollover button can be raised, sunken, flat or animated when a user clicks on it.

Uses of Rollover Buttons Rollover buttons are used primarily as navigational buttons on a web page to direct people to other locations. They are also used in drop-down and pop-up menus. Other rollover buttons are used for animated effects and sounds, so an image, color, shape, text or sound can change as the user rolls over the button on the web.

b. Write a code for creating a simple form and also write JavaScript that perform primitive checking of data. (4+8)

Answer:

Example:

```
<script type='text/javascript'>
function notEmpty(elem, helperMsg){
        if(elem.value.length == 0){
                alert(helperMsg);
                elem.focus();
                return false;
        }
        return true;
    }
    </script>
    <form>
```

```
Required Field: <input type='text' id='req1'/>
<input type='button'
  onclick="notEmpty(document.getElementById('req1'), 'Please Enter a Value')"
  value='Check Field' />
</form>
```

check for the following:

- If a text input is empty or not
- If a text input is all numbers
- If a text input is all letters
- If a text input is all alphanumeric characters (numbers & letters)
- If a text input has the correct number of characters in it (useful when restricting the length of a username and/or password)
- If a selection has been made from an HTML select input (the drop down selector)
- If an email address is valid
- How to check all above when the user has completed filling out the form

Checking for All Numbers

/ If the element's string matches the regular expression it is all numbers

```
function isNumeric(elem, helperMsg){
        var numericExpression = /^[0-9]+$/;
        if(elem.value.match(numericExpression)){
                return true;
        }else{
                alert(helperMsg);
                elem.focus();
                return false;
        }
}
```

Checking for All Letters

This function will be identical to *isNumeric* except for the change to the regular expression we use inside the *match* function. Instead of checking for numbers we will want to check for all letters.

If we wanted to see if a string contained only letters we need to specify an expression that allows for both lowercase and uppercase letters: /^[a-zA-Z]+$/ .

Example
// If the element's string matches the regular expression it is all letters

```
function isAlphabet(elem, helperMsg){
        var alphaExp = /^[a-zA-Z]+$/;
        if(elem.value.match(alphaExp)){
                return true;
        }else{
                alert(helperMsg);
                elem.focus();
                return false;
        }
}
```

Checking for Numbers and Letters

By combining both the *isAlphabet* and *isNumeric* functions into one we can check to see if a text input contains only letters and numbers.

Example

```
// If the element's string matches the regular expression it is numbers and letters
function isAlphanumeric(elem, helperMsg){
        var alphaExp = /^[0-9a-zA-Z]+$/;
        if(elem.value.match(alphaExp)){
                return true;
        }else{
                alert(helperMsg);
                elem.focus();
                return false;
        }
}
```

Q.6 a. Explain the scalar. Define the different functions which operate on scalars. Also define arrays and hash data types of Perl. (2+4+4)

Answer: I (9.4.1) Page No.286 – 287

Scalars

Variables can be used as placeholders for data. The simplest kind of placeholder in perl is called a scalar. Scalar variables can hold any kind of data: strings of text, integers, floating point numbers, and even binary data like images, if you want. Perl determines the type of data from context.

Arrays

For collections of variables, Perl offers two options: arrays and hashes. Arrays are the simplest kind of collection, so we will start with them.

First, define the array like this:

```
@status = ("leaving", "arriving", "staying", "whatever");
```

Note that instead of a $ sign, arrays are preceded by an @ sign.

So now you could add something like the following to your hello program:

```
@status = ("leaving", "arriving", "staying", "whatever");

foreach $status (@status) {

        if ($status eq "leaving") {

                $greeting = "Goodbye";
        }
        elsif ($status eq "arriving") {

                $greeting = "Hello";
        }
        else {

                $greeting =  "Nice day if it don't rain";
        }

        print $greeting . "\n";

}

print "All done.";
```

Variables can be referred to by number, for example:

```
@status = ("leaving", "arriving", "staying", "whatever");
$status = $status[1];
print $status;
```

The output of this simple script might surprise you: it prints 'arriving', the second value in the array. The reason is that arrays start their numbering at zero, not 1. To reference the first element in the array, you would need to use:
$status = $status[0];

One of the more confusing aspects of perl is that in this context, the array is preceded by a dollar sign. The reason is that $status[0] refers to a single element of the array, so it's really a scalar.

Hashes

There is another type of collection, called a hash. A hash is like an array, except that instead of referring to its elements by their number, in a hash you refer to them by an assigned key.

An example might be:

```
%names = (
          'Durno' => 'John',
          'Jordan' => 'Mark',
          'Hacker' => 'J.R.'
          );
```

Note that hashes are indicated by the percent sign. To find out the first name of the person named Durno, you could do something like:

```
$firstName = $names{'Durno'};
print $firstName;
```

b. What are the different control structures in PERL? Explain in detail. (6)
Answer: I (9.7) Page No. 299 – 305

Control Structures

Control structures allow the use of conditional logic in your scripts. The most common types of control structures are **'if', 'else', 'elsif', 'while', 'for', and 'foreach'** structures.

If, else, and while structures use the concept of "truth" to execute whatever code is defined in the applicable block.

if structure

```
if ($a eq $b) {
  print "a and b are equal";
} else {
  print "a and b are not equal";
}
```

This example tests to see if two variables, $a and $b, are the same; if they are, (that is to say, if the comparison is "true") the first block (the code between he first set of braces) is executed, and if they are not (i.e., if the comparison is not true), then the block of code after the 'else' is executed.

"elsif" is used if there are more than two possible outcomes of the test for truth:

```
if ($string eq "high") {
  print "String is 'high'";
} elsif ($string eq "low") {
  print "String is 'low'";
} elsif ($string eq "medium") {
  print "String is 'medium'";
} else {
  print "String is $string"; # $string is none of the above
}
```

"while" structures are loops. The test returns true while the defined condition is met. For example, this code will increment a number until it gets to 100 and then return false (i.e., stop incrementing):

```
while ($number < 100) {
 $number++;
 print $number;
}
```

"for" statements are similar but they set an initial value for the variable that is tested for truth. For statements are useful when you want to loop a specific number of times. For example, if you want to print out all numbers between 0 and 100, you would use the following code:

```
for ($num = 0; $num <= 100; $num++) {
  print $num . "\n";
}
```

"foreach" structures iterate through lists. For example, the following code will print out each of the members of an array:

```
@mylist = ('a','b','c','d','e');
foreach $letter (@mylist) {
 print $letter;
}
```

Q.7 a. What is CGI? Explain. (4)

Answer: I (10.1) Page No. 346

CGI, or Common Gateway Interface, is the standard programming interface between web servers and external programs. It is almost one of the most exciting and fun areas of programming today. The CGI standard lets web browsers pass information to programs written in any language. If you want to create a lightning-fast search engine, then your CGI program will most likely be written in C or C++. However, most other applications can use Perl. The CGI standard does not exist in isolation, it is dependent on the HTML and HTTP standards. HTML is the standard that lets web browsers understand document content. HTTP is the communications protocol that, among other things, lets web servers talk with web browser.

b. How to save a cookie after the header has been sent in Perl CGI? Explain. (4)

Answer:

HTTP provides no way to set them elsewhere. Cookies are only set in the header. If you want to set a cookie based on form data, then you do it in the response to the form submission request. If you want to use data both for generating a cookie and generating a form, then get that data into a variable before you send the header and use it in both locations. You could generate JavaScript to set the cookie in the HTML body but that would be unnecessarily complex and unreliable.

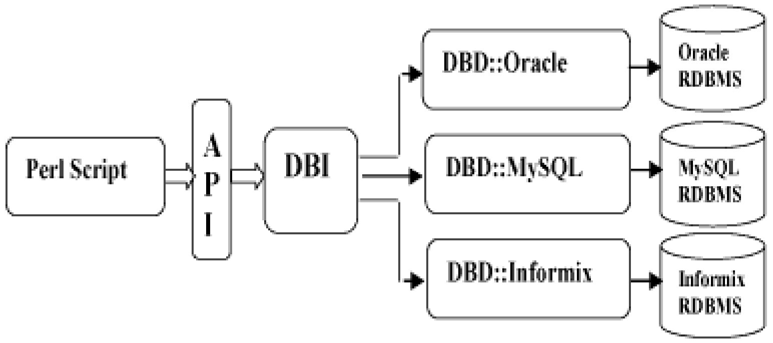
c. Explain PERL DBI though a diagram. (4+4)

Answer: I (11.3.2) Page No. 380 – 381

DBI stands for **Database Independent Interface** for Perl which means DBI provides an abstraction layer between the Perl code and the underlying database, allowing you to switch database implementations easily.

The DBI module enables your Perl applications to access multiple database types transparently. You can connect to PostgreSQL, SQLite, MySQL, MSSQL, Oracle, Informix, Sybase, ODBC and many more without having to know the different underlying interfaces of each. The API defined by DBI will work on all these database types and many more.

You can connect to multiple databases of different types at the same time and easily move data between them. The DBI layer allows you to do this simply and powerfully. Here's a diagram that demonstrates the principle:



Q.8 a. What are advantages of PHP sessions? How is a PHP session created?(4+4)

Answer: I (13.1.2) Page No. 482 – 483

A session is basically a way of storing variables and making them available across multiple pages on your web site. A PHP session allow you to store user information on the server for later use (i.e. username, shopping cart items, etc). However, this session information is temporary and is usually deleted very quickly after the user has **left** the website that uses sessions.

Starting a PHP Session

A PHP session is easily started by making a call to the **session_start()** function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to **session_start()** at the beginning of the page.

PHP code to start up a PHP session is

```
<?php
session_start(); // start up your PHP session!
?>
```

This small piece of code will register the user's session with the server, allow you to start saving user information and assign a UID (unique identification number) for that user's session.

Example: this example page will start a session, then it will echo the value of the session variable 'var'.

```
<?php
// begin the session
session_start();

// set the value of the session variable 'var'
$_SESSION['var']='bar';

// echo a little message to say it is done
echo 'Setting value of foo';
?>
```

b. Give a short note on PHP-XML Parser Functions. (4+4)

Answer:

The XML functions lets you parse, but not validate, XML documents.

XML is a data format for standardized structured document exchange. More information on XML can be found in our XML Tutorial. This extension uses the Expat XML parser. Expat is an event-based parser, it views an XML document as a series of events. When an event occurs, it calls a specified function to handle it.

Expat is a non-validating parser, and ignores any DTDs linked to a document. However, if the document is not well formed it will end with an error message.

Because it is an event-based, non validating parser, Expat is fast and well suited for web applications. The XML parser functions lets you create XML parsers and define handlers for XML events.

XML Parser Functions

- utf8_decode — Converts a string with ISO-8859-1 characters encoded with UTF-8 to single-byte ISO-8859-1
- utf8_encode — Encodes an ISO-8859-1 string to UTF-8
- xml_error_string — Get XML parser error string
- xml_get_current_byte_index — Get current byte index for an XML parser
- xml_get_current_column_number — Get current column number for an XML parser
- xml_get_current_line_number — Get current line number for an XML parser
- xml_get_error_code — Get XML parser error code
- xml_parse_into_struct — Parse XML data into an array structure
- xml_parse — Start parsing an XML document
- xml_parser_create_ns — Create an XML parser with namespace support
- xml_parser_create — Create an XML parser
- xml_parser_free — Free an XML parser
- xml_parser_get_option — Get options from an XML parser
- xml_parser_set_option — Set options in an XML parser
- xml_set_character_data_handler — Set up character data handler
- xml_set_default_handler — Set up default handler
- xml_set_element_handler — Set up start and end element handlers
- xml_set_end_namespace_decl_handler — Set up end namespace declaration handler
- xml_set_external_entity_ref_handler — Set up external entity reference handler
- xml_set_notation_decl_handler — Set up notation declaration handler
- xml_set_object — Use XML Parser within an object
- xml_set_processing_instruction_handler — Set up processing instruction (PI) handler
- xml_set_start_namespace_decl_handler — Set up start namespace declaration handler
- xml_set_unparsed_entity_decl_handler — Set up unparsed entity declaration handler

Q.9 a. Exemplify DOM to present an XML document as a tree-structure. (2+6)
Answer: I (14.4) Page No. 533 – 534

The XML DOM is:

- A standard object model for XML
- A standard programming interface for XML
- Platform- and language-independent
- A W3C standard

The XML DOM defines the **objects and properties** of all XML elements, and the **methods** (interface) to access them.

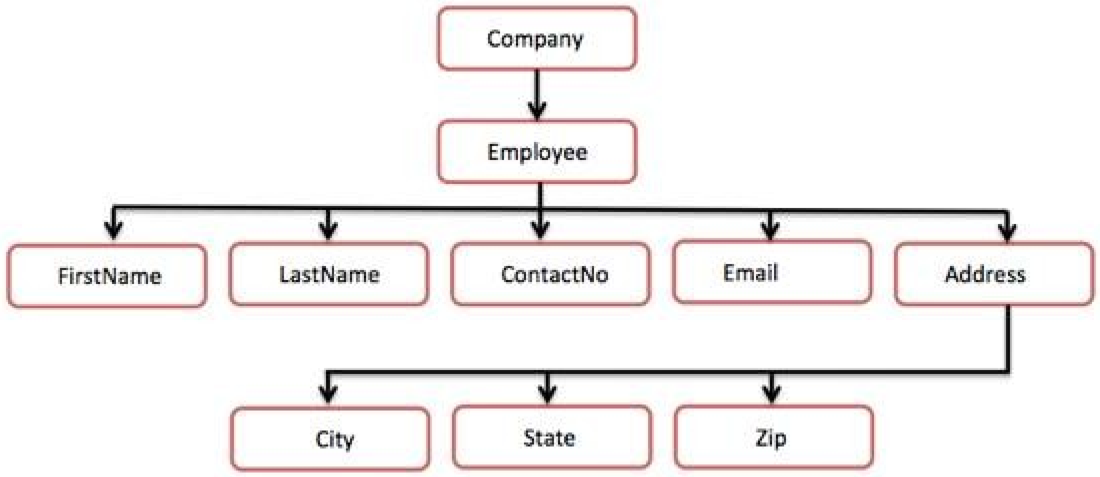
The XML DOM is a standard for how to get, change, add, or delete XML elements.

XML DOM Tree 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>
```

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



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>.

b. What is a DTD? Why is it used? **(2+2)**

Answer: I (14.2) Page No. 523

A Document Type Definition (DTD) is a set of rules that defines the elements, element attribute and attribute values, and the relationship between elements in a document. When your XML document is processed, it is compared to its associated DTD to be sure it is structured correctly and all tags are used in the proper manner. This comparison process is called validation and is is performed by a tool called a parser.

Remember, you don't need to have a DTD to create an XML document; you only need a DTD for a valid XML document.

Here's a few reasons you'd want to use a DTD:

• Your document is part of a larger document set and you want to ensure that the whole set follows the same rules.

• Your document must contain a specific set of data and you want to ensure that all required data has been included.

c. What is an Entity in XML? How many types of entities are used in xml? **(2+2)**

Answer: I (14.2.2) Page No. 528 – 529

Entities reference data that act as an abbreviation or can be found at an external location. Entities help to reduce the entry of repetitive information and also allow for easier editing (by reducing the number of occurrences of data to edit).

Entities in xml document is used for

- Denoting special markup, such as the > and < tags.
- Managing binary files and other data not native to XML.
- Reducing the code in DTD by bundling declarations into entities.
- Offering richer multilingual support.
- Repeating frequently used names in a way that guarantees consistency in spelling and use.
- Providing for easier updates. By using entities in your markup for items you know will be changed later-such as weather reports or software version changes-you greatly improve dynamic document automation.
- Merging multiple file links and interaction.

Types of entities

In general we have three types of entities: **internal entities, external entities and parameter entities.**

Internal Entities

- These are entities that refer to entities whose definitions can be found entirely within a document's DTD.

External Entities

- These are entity references that refer to entities whose definitions can be found outside of a document.

Parameter Entities

- These are available within internal or external subsets of DTD.
- In the subsequent pages we have discussed each of these entities in detail.
- Beside the type of Entities mentioned above, we finds it necessary to discuss the Document Entity of an xml document to be discussed specially.
- The document entity is the most important entity in an xml document and is actually one of only two kinds of entities that are allowed to exist without having a name assigned to them (the other one is the external subset of the DTD). This entity is the first thing the xml processor encounters when parsing a document. It is also referred to as the document root, and it provides programmatic access to the rest of the document. The reason the document entity is important is that, at

the end of the day, it's the only thing the xml specifications requires an xml parser to read.
- Document entities are defined as xml documents are parsed before it has been used.

TEXT BOOK

I. Web Programming – Building Internet Applications, Chris Bates, Third Edition, Wiley Student Edition, 2006