Q.2 a. How can a movie be inserted into an HTML document? Give example.

Answer:

```
Embed src="http://www.computerhope.com/issues/ibm-linux.mov"
        Pluginspage="http://www.apple.com/quicktime/" width="320"
        height="250" CONTROLLER="true" LOOP="false"
        AUTOPLAY="false" name="IBM Video"></embed>
```

In the above embed example the first portion of the code we point to the **src** (source) of the movie file, which is the movie file itself. Next the **pluginspage** is the location of the plug-in if the visitor does not have it installed. The **width** and **height** are the actual dimensions of the video file, may wish to increase the height of the video file if you are unable to see the controller. **Controller** can be set to true or false and is a setting that defines if the bottom controls for the video should be displayed or not. **Loop** Is a setting for when you wish for the file to automatically start over when finished. Finally, **autoplay** tells the browser to start or not to start playing the video file after the page has finished loading.

b. Write a HTML code to create radio buttons and a submit button.

Answer:

```
<!DOCTYPE html>
<html>
<body>

<form name="input" action="html_form_action.asp" method="get">
<input type="radio" name="sex" value="male" /> Male<br />
<input type="radio" name="sex" value="female" /> Female<br />
<input type="submit" value="Submit" />
</form>

<p>If you click the "Submit" button, the form-data will be sent to a page called
"html_form_action.asp".</p>

</body>
</html>
```

The output is:

- Male
- Female

Submit

If you click the "Submit" button, the form-data will be sent to a page called "html_form_action.asp".

c. What is XHTML Modularization? Why is it required?

Answer:
The decomposition of XHTML and by reference HTML into a group of modules that are abstract to provide modularity is known as XHTML modularization. These modules are utilized in the XML document type definition language. The rules needed to define the abstract model uses XML DTD.
Modularization of XHTML refers to specify well defined set of XHTML elements which can be compiled and extended by the XHTML document developers, other XML standards specifications.
Modularization of XTML acts as a means for the designers of product for specifying the elements that supports a specific devise by using standard building blocks and standard methods for building blocks usage. The content community gets “points of conformance” by using these modules.

Q.3 a. Write a CSS code to create sections and to number the sections and sub-sections with "Section 1", "1.1", "1.2", etc.

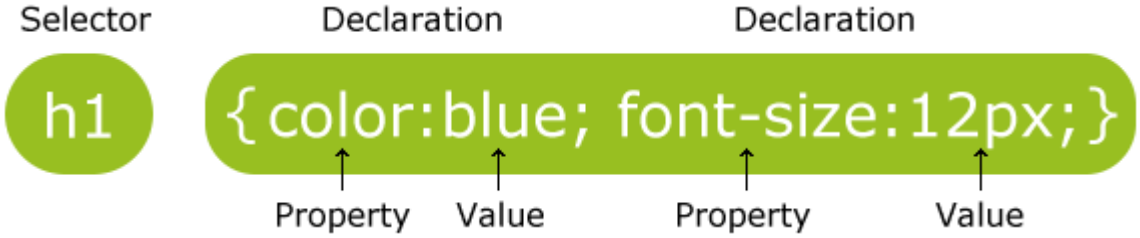
Answer:

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
body {counter-reset:section;}
h1 {counter-reset:subsection;}
h1:before
{
counter-increment:section;
content:"Section " counter(section) ". ";
}
h2:before
{
counter-increment:subsection;
content:counter(section) "." counter(subsection) " ";
}
</style>
</head>
<body>
<p><b>Note:</b> IE8 supports these properties only if a !DOCTYPE is specified.</p>
<h1>HTML tutorials</h1>
<h2>HTML Tutorial</h2>
<h2>XHTML Tutorial</h2>
<h2>CSS Tutorial</h2>
<h1>Scripting tutorials</h1>
<h2>JavaScript</h2>
<h2>VBScript</h2>
```

```
<h1>XML tutorials</h1>
<h2>XML</h2>
<h2>XSL</h2>
</body>
</html>
```

b. Give an example to explain CSS syntax and comments.

Answer:
A CSS rule has two main parts: a selector, and one or more declarations:



The selector is normally the HTML element you want to style.
Each declaration consists of a property and a value.
The property is the style attribute you want to change. Each property has a value.

CSS Example

A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly brackets:

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

To make the CSS more readable, you can put one declaration on each line, like this:

Example

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

CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers.
A CSS comment begins with "/*", and ends with "*/", like this:

```
/*This is a comment*/
p
{
text-align:center;
/*This is another comment*/
color:black;
font-family:arial;
}
```

c. Explain the background properties in CSS. Give examples of each property

Answer:
CSS background properties are used to define the background effects of an element.

CSS properties used for background effects:

* background-color
* background-image
* background-repeat
* background-attachment
* background-position

Background Color

The background-color property specifies the background color of an element.

The background color of a page is defined in the body selector:
Example
body {background-color:#b0c4de;}

Background Image
The background-image property specifies an image to use as the background of an element.
By default, the image is repeated so it covers the entire element.
The background image for a page can be set like this:

Example
body {background-image:url('paper.gif');}

Below is an example of a bad combination of text and background image. The text is almost not readable:

Example
body {background-image:url('bgdesert.jpg');}
Background Image - Repeat Horizontally or Vertically
By default, the background-image property repeats an image both horizontally and vertically.
Some images should be repeated only horizontally or vertically, or they will look strange, like this:
Example
body
{
background-image:url('gradient2.png');
}

f the image is repeated only horizontally (repeat-x), the background will look better:

Example

```
body
{
background-image:url('gradient2.png');
background-repeat:repeat-x;
}
```

Background Image - Set position and no-repeat

When using a background image, use an image that does not disturb the text.
Showing the image only once is specified by the background-repeat property:
Example

```
body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;
}
```

Background - Shorthand property

As you can see from the examples above, there are many properties to consider when dealing with backgrounds.
To shorten the code, it is also possible to specify all the properties in one single property. This is called a shorthand property.
The shorthand property for background is simply "background":

Example

```
body {background:#ffffff url('img_tree.png') no-repeat right top;}
```

All CSS Background Properties

Property	Description
background	Sets all the background properties in one declaration
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page
background-color	Sets the background color of an element
background-image	Sets the background image for an element
background-position	Sets the starting position of a background image
background-repeat	Sets how a background image will be repeated

Q.4 a. Explain exception handling in JavaScript. Write a code in JavaScript to create a confirm box to display a custom message telling users that they can click OK to continue viewing the page or click Cancel to go to the homepage.

Answer:

```
<!DOCTYPE html>
<html>
<head>
```

```
<script type="text/javascript">
var txt="";
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Click OK to continue viewing this page,\n";
  txt+="or Cancel to return to the home page.\n\n";
  if(!confirm(txt))
    {
    document.location.href="http://www.w3schools.com/";
    }
  }
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()" />
</body>

</html>
```

b. Write a code to execute a JavaScript when a user changes the content of an input field.

Answer:

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function upperCase()
{
var x=document.getElementById("fname");
x.value=x.value.toUpperCase();
}
</script>
</head>
<body>

Enter your name: <input type="text" id="fname" onchange="upperCase()" />
```

```
<p>A function is triggered when the input field is changed. The function transforms the input text to upper case.</p>
<p>Write some text in the input field, and the click outside the input field to run the function.</p>

</body>
</html>
```

o/p:

Enter your name:
A function is triggered when the input field is changed. The function transforms the input text to upper case.
Write some text in the input field, and the click outside the input field to run the function

Q.5 a. Write a code to add and remove HTML elements dynamically with JavaScript

Answer:

```
function addElement() {
  var ni = document.getElementById('myDiv');
  var numi = document.getElementById('theValue');
  var num = (document.getElementById('theValue').value -1)+ 2;
  numi.value = num;
  var newdiv = document.createElement('div');
  var divIdName = 'my'+num+'Div';
  newdiv.setAttribute('id',divIdName);
  newdiv.innerHTML = 'Element Number '+num+' has been added! <a href=\'#\' onclick=\'removeElement('+divIdName+')\'>Remove the div "'+divIdName+'"</a>';
  ni.appendChild(newdiv);
}

function removeElement(divNum) {
  var d = document.getElementById('myDiv');
  var olddiv = document.getElementById(divNum);
  d.removeChild(olddiv);
}
```

b. Write the JavaScript code to display messages in the status bar when the mouse is moved over a link.

Answer:

```
<ul>

<li><a href="term_1.html"
```

```
  onClick="alert('A caldera is a circular shaped landform
  depression caused by the eruption of a large, near surface
  body of magma.'); return false"
  onMouseOver="window.status='what is a caldera?';
  return true">caldera</a>
<li><a href="term_2.html"
  onClick="alert('Vesicularity is a measure how much of a
  rock volume consists of air chambers.'); return false"
  onMouseOver="window.status='what is vesicularity?';
  return true">vesicularity</a>
<li><a href="term_3.html"
  onClick="alert('Pahoehoe is a type of basaltic lava
  flow texture that comes from the Hawaiian word for
  smooth and ropy.'); return false"
  onMouseOver="window.status='what is pahoehoe?';
  return true">pahoehoe</a>
<li><a href="term_4.html"
  onClick="alert('Rheology is the study of how materials
  deform.'); return false"
  onMouseOver="window.status='what is rheology?';
   return true">rheology</a>
<li><a href="term_5.html"
  onClick='alert("A lahar is a mudslide generated from
  the flanks of a volcano. Some say it comes from the
  phrase \"Look Out!\" in the Indonesian language.");
  return false'
  onMouseOver="window.status='what is a lahar?';
  return true">lahar</a>
</ul>
```

NOTE: If you would like to copy and paste this code, use this sample of HTML that will load in a new browser window.

While we are adding this feature, it would help add MouseOver messages to the "hot area" of the clickable image map we created for this page in lesson 23. So modify the HTML between the <map>...</map> tags to read:

```
<map name="volcmap">
<area shape="rect"
href="http://volcano.und.edu/vwdocs/frequent_questions/grp7/europe/question308.html"
  coords="48,46,204,153"
  onMouseOver="window.status='information about surtseyan type volcanos';
  return true">
<area shape="rect"
  href="explode.html"
  coords="0,66,26,227"
  onMouseOver="window.status='description of explosiveness scale';
  return true">
<area shape="rect"
  href="height.html"
  coords="95,283,378,309"
  onMouseOver="window.status='description of height scale';
  return true">
<area shape="rect"
  href="http://www.geo.mtu.edu/volcanoes/pinatubo/"
  coords="321,154,468,261"
  onMouseOver="window.status='information about plinian type volcanos';
  return true">
<area shape="rect"
  href="http://stromboli.net/"
  coords="172,155,318,274"
  onMouseOver="window.status='information about strombolian type volcanos';
  return true">
```

```
<area shape="rect"
  href="http://hvo.wr.usgs.gov/volcanowatch/"
  coords="36,155,168,276"
  onMouseOver="window.status='information about hawaiian type volcanos';
  return true">
<area shape="rect"
  href="http://www.geo.mtu.edu/volcanoes/santamaria/"
  coords="90,3,343,123"
  onMouseOver="window.status='information about phreato-plinian type volcanos';
  return true">
</map>
```

Q.6 a. Briefly explain and give syntax for any five PERL functions that operate an arrays of data.

Answer:
If you want to make random numbers in a given range, inclusive, such as when you randomly pick an array index, simulate rolling a die in a game of chance, or generate a random password, use Perl's rand function:

```
$random = int( rand( $Y-$X+1 ) ) + $X;
```

This code generates and prints a random integer between 25 and 75, inclusive:

```
$random = int( rand(51)) + 25;
print "$random\n";
```

The rand function returns a fractional number, from (and including) 0 up to (but not including) its argument. We give it an argument of 51 to get a number that can be 0 or more, but never 51 or more. We take the integer portion of this to get a number from 0 to 50, inclusive (50.99999.... will be turned into 50 by int). We then add 25 to it to get a number from 25 to 75, inclusive.

A common application of this is the random selection of an element from an array:

```
$elt = $array[ rand @array ];
```

That's just like saying:

```
$elt = $array[ int( rand(0+@array) ) ];
```

Because rand is prototyped to take just one argument, it implicitly imposes scalar context on that argument, which, on a named array, is the number of elements in that array. The function then returns a floating-point number smaller than its argument and greater than or equal to zero. A floating-point number used as an array subscript implicitly undergoes integer truncation (rounding toward zero), producing in the end an evenly distributed, randomly selected array element to assign to $elt.

Generating a random password from a sequence of characters is similarly easy:

```
@chars = ( "A" .. "Z", "a" .. "z", 0 .. 9, qw(! @ $ % ^ & *) );
$password = join("", @chars[ map { rand @chars } ( 1 .. 8 ) ]);
```

We use map to generate eight random indices into @chars, extract the corresponding characters with a slice, and join them together to form the random password. This isn't a *good* random number, though, as its security relies on the choice of seed, which (in older versions of Perl) is based on the time the program started.

Q7 a. Give a code in PERL to explain the GET and POST methods. Explain how information is passed to a CGI program by POST method.

Answer:

Passing Information using POST method:

A generally more reliable method of passing information to a CGI program is the POST method. This packages the information in exactly the same way as GET methods, but instead of sending it as a text string after a ? in the URL it sends it as a separate message. This message comes into the CGI script in the form of the standard input.

Below is hello_post.cgi script to handle input given by web browser. This script will handle GET as well as POST method.

```
#!/usr/bin/perl

   local ($buffer, @pairs, $pair, $name, $value, %FORM);
   # Read in text
   $ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
   if ($ENV{'REQUEST_METHOD'} eq "POST")
   {
      read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
   }else {
         $buffer = $ENV{'QUERY_STRING'};
```

```
    }
    # Split information into name/value pairs
    @pairs = split(/&/, $buffer);
    foreach $pair (@pairs)
    {
          ($name, $value) = split(/=/, $pair);
          $value =~ tr/+/ /;
          $value =~ s/%(..)/pack("C", hex($1))/eg;
          $FORM{$name} = $value;
    }
    $first_name = $FORM{first_name};
    $last_name  = $FORM{last_name};

print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
print "<title>Hello - Second CGI Program</title>";
print "</head>";
print "<body>";
print "<h2>Hello $first_name $last_name - Second CGI Program</h2>";
print "</body>";
print "</html>";

1;
```

b. Give a procedure in CGI PERL to perform the following methods:
(i) Set up Cookies (ii) Retrieve Cookies

Answer:

i) Setting up Cookies

This is very easy to send cookies to browser. These cookies will be sent along with HTTP Header. Assuming you want to set UserID and Password as cookies. So it will be done as follows

```
#!/usr/bin/perl

print "Set-Cookie:UserID=XYZ;\n";
print "Set-Cookie:Password=XYZ123;\n";
print "Set-Cookie:Expires=Tuesday, 31-Dec-2007 23:12:40 GMT";\n";
print "Set-Cookie:Domain=www.tutorialspoint.com;\n";
print "Set-Cookie:Path=/perl;\n";
print "Content-type:text/html\r\n\r\n";
.....Rest of the HTML Content....
```

From this example you must have understood how to set cookies. We use **Set-Cookie** HTTP header to set cookies.

Here it is optional to set cookies attributes like Expires, Domain, and Path. It is notable that cookies are set before sending magic line **"Content-type:text/html\r\n\r\n**.

ii) Retrieving Cookies

This is very easy to retrieve all the set cookies. Cookies are stored in CGI environment variable HTTP_COOKIE and they will have following form.

key1=value1;key2=value2;key3=value3....

Here is an example of how to retrieving cookies.

```
#!/usr/bin/perl
  $rcvd_cookies = $ENV{'HTTP_COOKIE'};
  @cookies = split /;/, $rcvd_cookies;
  foreach $cookie ( @cookies ){
     ($key, $val) = split(/=/, $cookie); # splits on the first =.
     $key =~ s/^\s+//;
     $val =~ s/^\s+//;
     $key =~ s/\s+$//;
     $val =~ s/\s+$//;
     if( $key eq "UserID" ){
        $user_id = $val;
     }elsif($key eq "Password"){
        $password = $val;
     }
  }
print "User ID  = $user_id\n";
print "Password = $password\n";
```

This will produce following result
User ID = XYZ
Password = XYZ123

Q9. a. Explain the following terms in XML:

(i) Prolog
(ii) Entity
(iii) Parser

Answer:

XML Entity

An entity is a symbolic representation of information. What does that mean? Well, let's imagine for a moment that we want to create an introduction that is included in every single memo that we write. It would be monotonous to have to type out a three sentence introduction for every letter, but not to worry! XML entities can help us out.

With symbolic representation of information, a lot of text, such as, "Hello my name is Tizag.com and I am an artificial intelligence that teaches the general public how to program in web languages for free," can be represented by an entity symbol.

Entity Syntax
You may have used entities in the past. The format of an entity in XML is an ampersand (&), followed by the name of the symbol, and concluded with a semicolon.

- Generic Entity - &name;

HTML is another markup language that supports entities. Below are some example entities and the information they represent.

- © = ©
- < = <
- & = &
- " = "

Creating an XML Entity
An entity must be created in the Document Type Definition (DTD). Once you know where to place the entity, the rest is easy. Here is the syntax for creating your own XML entities.

- <!ENTITY entityName "The text you want to appear when the entity is used">

Below, we have created an entity for the default introduction we want to include on all of our documents.
XML Code:
<!ENTITY intro "Hello
my name is Tizag.com and I am an artificial intelligence that teaches the general public how to program in web languages for free">
Using Your Entity
After the entity has been created in the DTD, it can then be referenced. An example email XML document that uses such an entity would look like:
XML Code:
<!ENTITY intro "Hello
my name is Tizag.com and I am an artificial intelligence that teaches the general public how to program in web languages for free">
<email>
<to>A. Nony Mouse</to>
<body>&intro;</body>
</email>
Entities are great for many situations. Such as when you...

- Use something a lot. If you have a default introduction, signature, or something else that is commonly used, you should use an entity.
- Change something often. If you have a relatively static document that has one or two pieces of frequently-changing information that are used throughout the document, replace them with entities. You only need to change the value of the

entity to change hundreds or maybe even thousands of references that are in your XML document.
- Are using complex ASCII characters that don't occur on your keyboard: © and ® are easy when you use entities.

II. All modern browsers have a built-in XML parser.

An XML parser converts an XML document into an XML DOM object - which can then be manipulated with JavaScript.

Parse an XML Document

The following code fragment parses an XML document into an XML DOM object:

```
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.open("GET","books.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;
```

Parse an XML String

The following code fragment parses an XML string into an XML DOM object:

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

```
if (window.DOMParser)
  {
  parser=new DOMParser();
  xmlDoc=parser.parseFromString(txt,"text/xml");
  }
else // Internet Explorer
  {
  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  xmlDoc.async=false;
  xmlDoc.loadXML(txt);
  }
```

III. XML Prolog

The prolog is an optional component of the XML document. If included, the prolog must be appear **before** the root element. A prolog consists of two parts: the XML declaration and the Document Type Declaration (DTD). Depending on your needs, you can choose to include both, either, or neither of these items in your XML document. The DTD is most oftenly used, so we will discuss its use and purpose firs

here are two type declarations that may be used to reference an external DTD: PUBLIC and SYSTEM. When creating an XML document under the rules of a publicly distributed DTD, use PUBLIC. Otherwise, use the SYSTEM type declaration.

The example below shows a prolog that would be used for an HTML document that is using an XML prolog. The DTD is publicly available thanks to the W3C.

XML Code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Let's take a look at each piece of this external DTD reference.

- **!DOCTYPE** - Tell the XML processor that this piece of code defines the Document Type Definition
- **html** - Specifies the root element of the XML document. Here our example is an HTML file, which has <html> as the root element.
- **PUBLIC** - Specifies that this a publicly available DTD.
- **"-//W3C//D..."** - The definition of the public document. Describing this is beyond the scope of this tutorial.
- **"http://www.w3.org/..."** - The physical location of the DTD. Notice how this DTD resides on w3's servers.

b. Give an example which shows XML document has a reference to an XML Schema.

Answer:

ML documents can have a reference to a DTD or to an XML Schema.

A Simple XML Document

Look at this simple XML document called "note.xml":

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

A DTD File

The following example is a DTD file called "note.dtd" that defines the elements of the XML document above ("note.xml"):

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

The first line defines the note element to have four child elements: "to, from, heading, body".

Line 2-5 defines the to, from, heading, body elements to be of type "#PCDATA".

An XML Schema

The following example is an XML Schema file called "note.xsd" that defines the elements of the XML document above ("note.xml"):

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="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 note element is a complex type because it contains other elements. The other elements (to, from, heading, body) are simple types because they do not contain other elements. You will learn more about simple and complex types in the following chapters.

A Reference to a DTD

This XML document has a reference to a DTD:

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

<!DOCTYPE note SYSTEM
"http://www.w3schools.com/dtd/note.dtd">
```

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

A Reference to an XML Schema

This XML document has a reference to an XML Schema:

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

<note
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Text Book

Web Programming –Building Internet Applications, Chris Bates, 3rd Edition, Wiley Student Edition, 2006.