CMR
INSTITUTE OF
TECHNOLOGY

USN

CMR

IMPROVEMENT TEST solution – November 2016

Sub:	**Programming The WEB**							Code:	10CS73
Date:	16/ 11/ 16	Duration:	90 Mins	Max Marks:	50	Sem:	VII A,B	Branch:	CSE

Note: **Answer to the point. Sketch figures wherever necessary.**

Answer any 5 full questions.

		Marks
1	**Explain the Positioning and Moving of Element in Dynamic XHTML with examples?** Ans: **Cascading Style Sheet – Positioning** (CSS-P) It is completely supported by IE8 and FX3. • It provides the means not only to position any element anywhere in the display of a document, but also to move an element to a new position in the display dynamically, using JavaScript to change the positioning style properties of the element. • These style properties, which are appropriately named left and top, dictate the distance from the left and top of some reference point to where the element is to appear. • Another style property, position, interacts with left and top to provide a higher level of control of placement and movement of elements. • The position property has three possible values: absolute, relative, and static. **ABSOLUTE POSITIONING ** • The absolute value is specified for position when the element is to be placed at a specific place in the document display without regard to the positions of other elements. • One use of absolute positioning is to superimpose special text over a paragraph of ordinary text to create an effect similar to a watermark on paper. • A larger italicized font, in a light-gray color and with space between the letters, could be used for the special text, allowing both the ordinary text and the special text to be legible. //absPos.html <html> <title>Absolute Positioning</title> <style type = "text/css"> .regtext { font-family: Cambria; font-size:20pt; width: 900px; } .abstext { position:absolute; top:25px; left:100px; font-family: jokerman; font-size:30pt; width: 500px; color:#0000cd; letter-spacing: 1em; } </style> </head> <body>	**10M**

```
<p class="regtext"> Kannadada Kotyadhipathi is a Kannada primetime quiz show hosted by the power star of Kannada cinema Mr. Puneet  question. </p>
<p class="abstext"> POWER STAR PUNEETH RAJKUMAR </p>
 </body>
</html>
```

RELATIVE POSITIONING

- An element that has the position property set to relative, but does not specify top and left property values, is placed in the document as if the position attribute were not set at all.
- However, such an element can be moved later.
- If the top and left properties are given values, they displace the element by the specified amount from the position where it would have been placed.
- In both the case of an absolutely positioned element inside another element and the case of a relatively positioned element, negative values of top and left displace the element upward and to the left, respectively.
- Relative positioning can be used for a variety of special effects in placing elements.

```
//relPos.html
<html>
<head><title>Relative Positioning</title></head> <body>
<p> I am the <span style = "position: relative; top:10px; font-family: Cambria; font-size:25px; font-style: italic; color: purple;"> CULTURAL SECRETARY </span> of CMRIT </p> </body>
</html>
```

STATIC POSITIONING

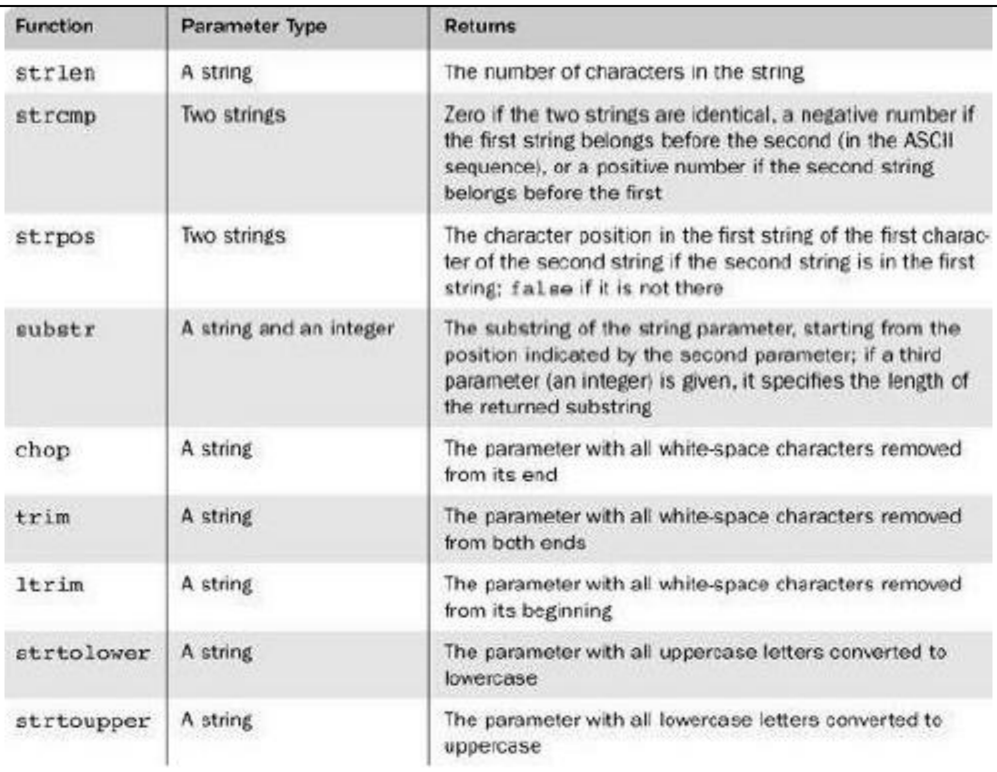
The default value for the position property is static. A statically positioned element is placed in the document as if it had the position value of relative but no values for top or left were given. The difference is that a statically positioned element cannot have its top or left properties initially set or changed later. Therefore, a statically placed element cannot be displaced from its normal position and cannot be moved from that position later.

FIXED POSITIONING

Set the Position Property to particular document, it will be fixed even you scroll down, that position is fixed.

```
//FixPos.html
<html>
<head><title>Fixed Positioning</title></head> <body>
<p> I am the <span style = "position: fixed; top:10px; font-family: Cambria; font-size:25px; font-style: italic; color: purple;"> CULTURAL SECRETARY </span> of CMRIT </p> </body>
</html>
```

2	**a) Explain any five string functions in PHP.** Ans:	**5M**

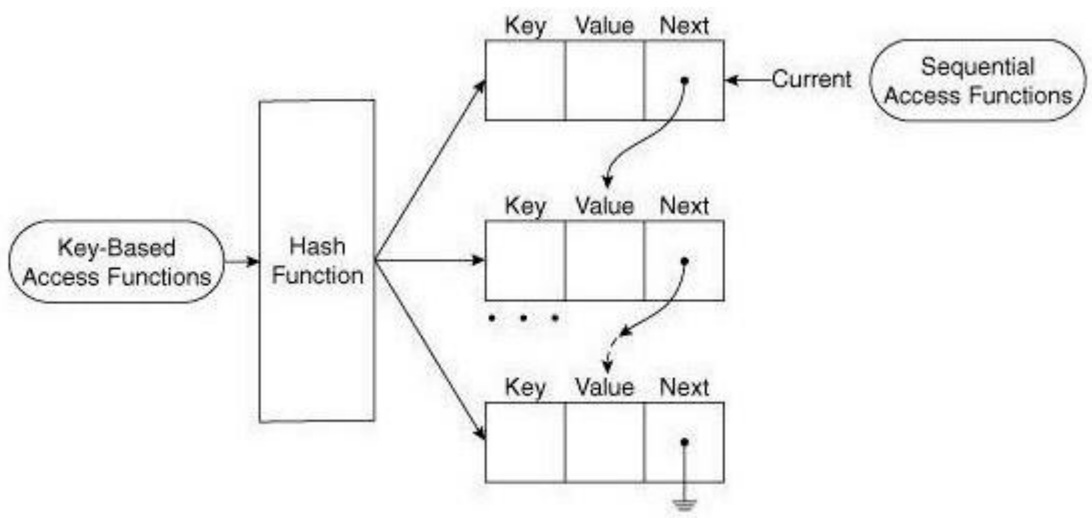


Function	Parameter Type	Returns
`strlen`	A string	The number of characters in the string
`strcmp`	Two strings	Zero if the two strings are identical, a negative number if the first string belongs before the second (in the ASCII sequence), or a positive number if the second string belongs before the first
`strpos`	Two strings	The character position in the first string of the first character of the second string if the second string is in the first string; `false` if it is not there
`substr`	A string and an integer	The substring of the string parameter, starting from the position indicated by the second parameter; if a third parameter (an integer) is given, it specifies the length of the returned substring
`chop`	A string	The parameter with all white-space characters removed from its end
`trim`	A string	The parameter with all white-space characters removed from both ends
`ltrim`	A string	The parameter with all white-space characters removed from its beginning
`strtolower`	A string	The parameter with all uppercase letters converted to lowercase
`strtoupper`	A string	The parameter with all lowercase letters converted to uppercase

b) With an neat diagram, explain logical internal structure of array in PHP

5M

Ans:



Internally, the elements of an array are stored in a linked list of cells, where each cell includes both the key and the value of the element. The cells themselves are stored in memory through key-hashing function so that they are randomly distributed in a reserved block of storage. Accesses to elements through string keys are implemented through the hashing function.

However, the elements all have links that connect them in the order in which they were created, allowing them to be accessed in that order if the keys are strings and in the order of their keys if the keys are numbers.

3 **a)Explain the sort, rsort, ksort array functions in PHP with example**

6M

Ans: PHP - Sort Functions For Arrays

The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.

sort() - sort arrays in ascending order

rsort() - sort arrays in descending order

ksort() - sort associative arrays in ascending order, according to the key

Sort Array in Ascending Order - sort()

The following example sorts the elements of the $cars array in ascending alphabetical order:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

Output:

```
 BMW
Toyota
Volvo
```

Sort Array in Descending Order - rsort()

The following example sorts the elements of the $cars array in descending alphabetical order:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);
?>
```

Output

```
Volvo
Toyota
BMW
```

Sort Array (Ascending Order), According to Key - ksort()

The following example sorts an associative array in ascending order, according to the key:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
?>
```

Output:

```
Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35
```

b) Explain element visibility

4M

Ans: Document elements can be specified to be visible or hidden with the value of their visibility
property.

- The two possible values for visibility are, quite naturally, visible and hidden.
- The appearance or disappearance of an element can be controlled by the user through a widget.

```
//showHide.html
<html>
<head> <title>Visibility Control</title>
<script type="text/javascript" src="showHide.js">
</script> </head>
<body>
```

```
<form action="">
<div id="rnsit" style="position:relative; visibility: visible;">
<img src = "CLASS PHOTO.jpg" alt = "RNSIT CSE 2009-2013" /> </div>
<p><br><input type="button" value="Toggle Rnsit" onclick = "flipImag()"
/></p>
</form>
</body>
</html>
//showHide.js
function flipImag() {
dom=document.getElementById("rnsit").style;
if(dom.visibility == "visible")
dom.visibility = "hidden";
else
dom.visibility = "visible";
}
```

4 **Briefly discuss the event handling from body elements and button element in JavaScript**. 10M

Ans:

HANDLING EVENTS FROM BODY ELEMENTS

The events most often created by body elements are load and unload. As our first example of event handling, we consider the simple case of producing an alert message when the body of the document has been loaded. In this case, we use the onload attribute of <body> to specify the event handler:

```
<?xml version = "1.0"  encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- load.html
     A document for load.js
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> load.html </title>
    <script type = "text/javascript"  src = "load.js" >
    </script>
  </head>
  <body onload="load_greeting();">
    <p />
  </body>
</html>
// load.js
//   An example to illustrate the load event

// The onload event handler
function load_greeting () {
  alert("You are visiting the home page of \n" +
        "Pete's Pickled Peppers \n" + "WELCOME!!!");
}
```

The unload event is probably more useful than the load event. It is used to do some cleanup before a document is unloaded, as when the browser user goes on to some new document. For example, if the document opened a second browser window, that window could be closed by an unload event handler

HANDLING EVENTS FROM BUTTON ELEMENTS

Buttons in a Web document provide an effective way to collect simple input from the browser user. Example: //radio_click.html

```
<html> <head> <title> radio_click.html</title>
<script type = "text/javascript" src = "radio_click.js"> </script> </head>
<body> <h4> Choose your favourite Director in Kannada Film Industry</h4>
<form id = "myForm" action = " ">
<p> <label>
<input type = "radio" name = "dButton" value = "1" onclick = "dChoice(1)"/> Yogaraj
Bhat</label><br/>
<label><input type = "radio" name = "dButton" value = "2" onclick = "dChoice(2)"/>
Suri</label><br/>
<label><input type = "radio" name = "dButton" value = "3" onclick = "dChoice(3)"/> Guru
Prasad</label><br/>
<label><input type = "radio" name = "dButton" value = "4" onclick = "dChoice(4)"/>
Prakash</label> </p>
 </form> </body> </html>
//radio_click.js
function dChoice(ch) {
 switch(ch) {
 case 1: alert("Mungaaru Male");
break;
case 2: alert("Duniya");
break;
case 3: alert("Eddelu Manjunatha");
break;
case 4: alert("Milana");
break;
default: alert("Ooops..Invalid choice :O");
break;
}
}
```

The next example, radio_click2.html, whose purpose is the same as that of radio_click.html, registers the event handler by assigning the name of the handler to the event properties of the radio button objects. The following example uses three files—one for the XHTML, one for the script for the event handlers, and one for the script to register the handlers:

```
//radio_click2.html
<html>
<head>
<title> radio_click2.html</title>
<script type = "text/javascript" src = "radio_click2.js"> </script> </head>
<body> <h4> Choose your favourite Director in Kannada Film Industry</h4>
 <form id = "myForm" action = " ">
<p>
<label><input type = "radio" name = "dButton" value = "1" id = "1"/> Yogaraj
Bhat</label><br/>
 <label><input type = "radio" name = "dButton" value = "2" id = "2"/> Suri</label><br/>
<label><input type = "radio" name = "dButton" value = "3" id = "3"/> Guru
Prasad</label><br/>
<label><input type = "radio" name = "dButton" value = "4" id = "4"/> Prakash</label> </p>
</form>
 <script type = "text/javascript" src = "radio_click2r.js"> </script> </body> </html>
//radio_click2.js
function dChoice(ch) {
var dom = document.getElementById("myForm");
for(var index = 0; index < dom.dButton.length; index++) {
if(dom.dButton[index].checked) {
ch = dom.dButton[index].value;
break;
}
}
switch(ch) {
case 1: alert("Mungaaru Male");
break;
case 2: alert("Duniya");
break;
case 3: alert("Eddelu Manjunatha");
break;
case 4: alert("Milana");
break;
default: alert("Ooops..Invalid choice :O");
break;
}
}
//radio_click2r.js
var dom = document.getElementById("myForm");
dom.getElementById("1").onclick = dChoice; dom.getElementById("2").onclick = dChoice;
dom.getElementById("3").onclick = dChoice; dom.getElementById("4").onclick = dChoice;
```

There are two advantages to registering handlers as properties over registering them in XHTML attributes.

First, it is good to keep XHTML and Java-Script separated in the document. This allows a kind of modularization of XHTML documents, resulting in a cleaner design that will be easier to maintain.

Second, having the handler function registered as the value of a property allows for the possibility of changing the function during use.

5 **Explain below keywords briefly with an example** **10M**

i) SESSION

Ans:

Session tracking

- Some applications need to keep track of a session
- Sessions are represented internally in PHP with a session id
- A session consists of key/value pairs
- A session can be initialized or retrieved by using the session_start function
- This function retrieves $_SESSION, an array containing the key/value pairs for each cookie in the current request

```
<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>
```

Example://Number of times Visited the SESSION

```
<?php
session_start();
```

```
if(isset($_SESSION['count']))
{
echo "The page view count:" .$_SESSION['count'],"<br/>";
$_SESSION['count']++;
}
else
{
$_SESSION['count']=1;
echo "Session not yet start";
}
?>
```

COOKIE

• HTTP is a *stateless* protocol, that is, the server treats each request as completely separate from any other

• This, however, makes some applications difficult

• A shopping cart is an object that must be maintained across numerous requests and responses

• The mechanism of cookies can be used to help maintain state by storing some information on the browser system

• A cookie is a key/value pair that is keyed to the domain of the server

• This key/value pair is sent along with any request made by the browser of the same server

• A cookie has a lifetime which specifies a time at which the cookie is deleted from the browser

• Cookies are only returned to the server that created them

• Cookies can be used to determine usage patterns that might not otherwise be ascertained by a server

• Browsers generally allow users to limit how cookies are used

• Browsers usually allow users to remove all cookies currently stored by the browser

• Systems that depend on cookies will fail if the browser refuses to store them

Example://Set COOKIE for Two Months

```
<?php
if(isset($_COOKIE['LastVisit']))
{
 $visit = $_COOKIE['LastVisit'];
echo "Your last visit was - ".$visit;
}
else
echo "you've got some stale cookies!";
$inTwoMonths = 60 * 60 * 24 * 60 + time();
setcookie('LastVisit',date("g:i - m/d/y"), $inTwoMonths);
?>
```

6 **a) Explain Keyboard input and screen output in ruby.** **5M**

Ans: The operand for puts is a string literal. A newline character is implicitly appended to the string operand. If the value of a variable is to be part of a line of output, the

```
        #{...}
```

notation can be used to insert it into a double-quoted string literal, as in the following interactions:

```
>> name = “Fudgy”

=> “Fudgy”
>> puts “My name is #{name}”
My name is Fudgy
=> nil
```

The value returned by puts is nil, and that is the value returned after the string has been displayed.

The print method is used if you do not want the implied newline that puts adds to the end of your literal string.

The way to convert a floating-point value to a formatted string is with a variation of the C language function sprintf. This function, which also is named sprintf, takes a string parameter
that contains a format code followed by the name of a variable to be converted. The string version is returned by the function. The format codes most commonly used are f and d.

The form of a format code is a percent sign (%), followed by a field width, and followed by the code letter (f or d). The field width for the f code appears in two parts, separated by a decimal point. For example, %f7.2 means a total field width of seven spaces, with two digits to the right of the decimal point—a perfect format for money. The d code field width is just a number of spaces—for example, %5d. So, to convert a floating-point value referenced by the variable total to a string with two digits to the right of the decimal point, the following statement could be used:

```
str = sprintf(“%5.2f”, total)
```

Keyboard Input

Because Ruby is used primarily for Rails in this book, there is little need for keyboard Input. However, keyboard input is certainly useful for other applications, so it is briefly introduced here.

The gets method gets a line of input from the keyboard.

The retrieved line includes the newline character. If the newline is not needed, it can be discarded with chomp:

```
>> name = gets
apples
=> “apples\n”
>> name = name.chomp
=> “apples”
```

This code could be shortened by applying chomp directly to the value returned by gets:

```
>> name = gets.chomp
Apples
=> “apples”
```

If a number is to be input from the keyboard, the string from gets must be converted to an integer with the to_i method, as in the following interactions:

```
>> age = gets.to_i
27
=> 27
```

If the number is a floating-point value, the conversion method is to_f:

```
>> age = gets.to_f
27.5
=> 27.5
```

b)Explain with a neat diagram, directory structure of rails1 application

Ans: Rails responds by creating more than 45 files in more than 30 directories. This is part of the framework that supports a Rails application. Directly under the specific application directory—in this case, greet—11 subdirectories are created, the most interesting of which atthis point is app. The app directory has four subdirectories: models, views, and controllers—which correspond directly to the MVC architecture of a Rails application—and helpers. The helpers subdirectory contains Rails-provided methods that aid in constructing applications. Most of the user code to support an application will reside in models, views, or controllers, Orin subdirectories of those directories.

One of the directories created by the rails command is script, which has several important Ruby scripts that perform services. One such script, generate, is used to create part of an application controller. This script creates a file containing a class in the controller's directory, and also a subdirectory of the views directory where views documents will be stored. For our application, we pass two parameters to generate, the first of which is controller, which indicates that we want the controller class built. The second parameter is the name we chose for the controller. An important part of how Rails works is its focused use of names. Our first example of this feature is the name of the controller, which will also be part of the file name of the controller class and part of the name of the controller class itself. In addition, it will be the name of the subdirectory of the views directory and a part of the URL of the application.

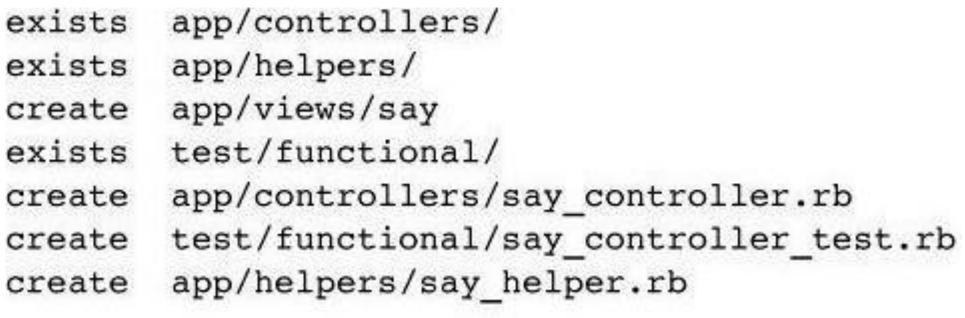
For our example, the following command is given in the greet directory to create the controller:

5M

```
>ruby script/generate controller say
```

With this command, we have chosen the name say for the controller for our application.

The response produced by the execution of the command is as follows:



```
exists  app/controllers/
exists  app/helpers/
create  app/views/say
exists  test/functional/
create  app/controllers/say_controller.rb
create  test/functional/say_controller_test.rb
create  app/helpers/say_helper.rb
```

The exists lines in this response indicate files and directories that Rails found to already exist.

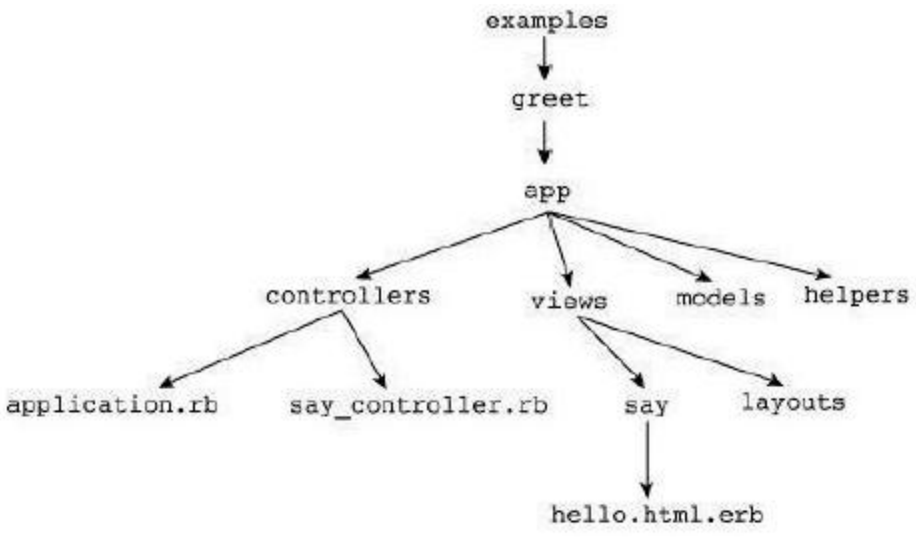
The create lines show the newly created directories and files. There are now two files in the controllers directory—application.rb and say_controller.rb which contain the Application Controller and SayController classes, respectively. The SayController class is a subclass of the ApplicationController class. As the parent class, Application-Controller provides the default behavior for SayController, the controller class of the application class.

There may be other controllers and their corresponding controller classes in an application. Such classes are subclasses of ApplicationController, which was built by the initial rails command.

The following is a listing of say_controller.rb:

```
class SayController < ApplicationController
end
```

The template file for our application resides in the say subdirectory of the views subdirectory of the app subdirectory of the greet directory.



7 a**) Write a PHP functions that reads contents from a file and write into a file** 5M

Ans: **Reading from a File**

The most common way to input a text file in PHP is to read the contents of the file into a scalar variable as a string. Then the impressive collection of PHP string manipulation functions can be used to process the file as a string. The fread function reads part or all of a file and returns a string of what was read. This function takes two parameters: a file variable and the number of bytes to be read. The reading operation stops when either the end-of-file marker is read or the specified number of bytes has been read. Large collections of data are often stored in database systems, so usually only smaller data sets are stored in files.

Therefore, files are often read in their entirety with a single call to fread. If the whole file is to be read at once, the file's length is given as the second parameter to fread. The best way to get the correct file length is with the filesize function, so a call to filesize is often used as the second parameter to fread. The filesize function takes a single parameter, the name of the file (not the file variable). For example, to read the entire contents of the file testdata.dat as a string into the variable $file_string, the following statement could be used:

```
$file_string = fread($file_var,
                     filesize("testdata.dat"));
```

One alternative to fread is file, which takes a file name as its parameter and returns an array of all of the lines of the file. (A line is a string of non-newline characters, followed by a newline.) One advantage of file is that the file open and close operations are not necessary. For example, the following statement places the lines of testdata.dat into an array named @file_lines:

```
$file_lines = file("testdata.dat");
```

PHP has another file input function that does not require calling fopen:

The function file_get_contents takes the file's name as its parameter.

This function reads the entire contents of the file, as exemplified in the following call:

```
$file_string = file_get_contents(“testdata.dat”);
```

A single line of a file can be read with fgets, which takes two parameters: the file variable and a limit on the length of the line to be read. As an example, the statement

```
$line = fgets($file_var, 100);
```

reads characters from the file whose file variable is $file_var until it finds a newline character, encounters the end-of-file marker, or has read 99 characters. Note that the maximum number of characters fgets reads is one fewer than the limit given as its second parameter.

A single character can be read from a file with fgetc, whose only parameter is the file variable. In reading a file by lines or by characters, the read operation must be controlled by the detection of the end of the file. This can be done with the feof function, which takes a file variable as its only parameter. It returns a Boolean value: TRUE if the last-read character of the file was the end-of-file character, FALSE.

Writing to a File

The fwrite7 function takes two parameters: a file variable and the string to be written to the file. The fwrite function returns the number of bytes written.

The following is an example of a call to fwrite:

```
$bytes_written = fwrite($file_var, $out_data);
```

This statement writes the string value in $out_data to the file referenced with $file_var and places the number of bytes written in $bytes_written. Of course, this will work only if the file has been opened for writing.

The file_put_contents function is the counterpart of file_get_contents —it writes the value of its second parameter, a string, to the file specified in its first parameter.

For example,

consider the following call:

```
file_put_contents(“savedata.dat”, $str);
```

b)Write a short note on Ruby on Rails

5M

Ans:

RUBY:

- Ruby was designed in Japan by Yukihiro Matsumoto (a.k.a. Matz) and was released in 1996. It started as a replacement for Perl and Python, languages that Matz found inadequate for his purposes. The use of Ruby grew rapidly in Japan and spread to the rest of the world a few years later.
- The quick growth of the use of Rails, the Web application development framework that both is written in Ruby and uses Ruby, has accelerated the expansion of the language.
- Rails is probably the most common use of Ruby. Learning Ruby is made easier by its implementation method: pure interpretation. Rather than needing to learn about and write a layer of boilerplate code around some simple logic, in Ruby one can write just that simple logic and request its interpretation.
- For example,
  consider the difference between a complete —Hello, World program in a language like C++ or Java and the Ruby —Hello, World program: puts —Hello, World From Perl, Ruby gets regular expressions and implicit variables.

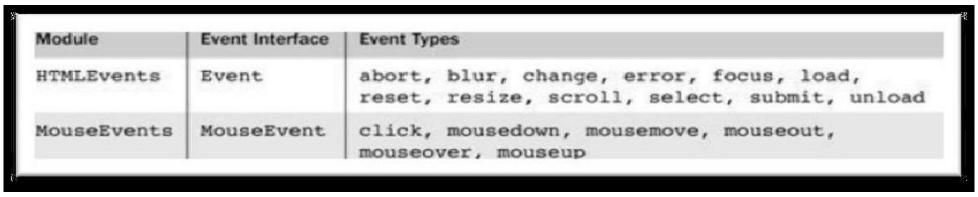
From Java-Script, it gets objects that can change during execution. However, Ruby has many more differences with those languages than it has similarities. For example, as in pure object-oriented languages, every data value in Ruby is an object, whether it is a simple integer literal or a complete file system. Ruby is available for every common computing platform. Furthermore, as is the case with PHP, the Ruby implementation is free.

RAILS:

- Rails respond by creating more than 45 files in more than 30 directories. This is part of the framework that supports a Rails application. Directly under the specific application directory—in this case, greet—11 subdirectories are created, the most interesting of which at this point is app.
- The app directory has four subdirectories: models, views, and controllers— which correspond directly to the MVC architecture of a Rails application—and helpers. The helpers subdirectory contains Rails-provided methods that aid in constructing applications. Most of the user code to support an application will reside in models, views, or controllers, or in subdirectories of those directories.
- One of the directories created by the rails command is script, which has several important Ruby scripts that perform services. One such script, generate, is used to create part of an application controller.
- This script creates a file containing a class in the controllers directory, and also a subdirectory of the views directory where views documents will be stored. For our application, we pass two parameters to generate, the first of which is controller, which indicates that we want the controller class built. The second parameter is the name we chose for the controller.
- An important part of how Rails works is its focused use of names.
- Our first example of this feature is the name of the controller, which will also be part of the file name of the controller class and part of the name of the controller class itself.
- In addition, it will be the name of the subdirectory of the views directory and a part of the URL of the application.
- For our example, the following command is given in the greet directory to create the controller: ruby script/generate controller say

8 **Explain event handler connection for DOM2 event model**.

Ans: The DOM 2 model is a modularized interface. One of the DOM 2 modules is Events, which includes several sub-modules. The ones most commonly used are HTML Events and MouseEvents. The interfaces and events defined by these modules are as follows



Module	Event Interface	Event Types
HTMLEvents	Event	abort, blur, change, error, focus, load, reset, resize, scroll, select, submit, unload
MouseEvents	MouseEvent	click, mousedown, mousemove, mouseout, mouseover, mouseup

EVENT PROPAGATION:

• A browser which understands DOM, on receiving the XHTML document from the server, creates a tree known as document tree.

• The tree constructed consists of elements of the document except the HTML

• The root of the document tree is document object itself

• The other elements will form the node of the tree

• In case of DOM2, the node which generates an event is known as target n Once the event is generated, it starts the propagation from root node

10M

- During the propagation, if there are any event handlers on any node and if it is enabled then event handler is executed
- The event further propagates and reaches the target node.
- When the event handler reaches the target node, the event handler gets executed
- After this execution, the event is again re-propagated in backward direction
- During this propagation, if there are any event handlers which are enabled, will be executed.
- The propagation of the even from the root node towards the leaf node or the target node is known as **capturing phase.**
- The execution of the event handler on the target node is known as **execution phase.**
- This phase is similar to event handling mechanism in DOM – 0
- The propagation of the event from the leaf or from the target node is known as **bubbling phase**
- All events cannot be bubbled for ex: load and unload event
- If user wants to stop the propagation of an event, then stop propagation has to be executed.

EVENT REGISTRATION:

- In case of DOM2, the events get registered using an API known as **addEventListener**
- The first arg is the eventName. Ex: click, change, blur, focus
- The second arg is the event handler function that has to be executed when there is an event
- The third arg is a Boolean argument that can either take a true or false value
- If the value is true, it means event handler is enabled in capturing phase
- If the event value if off (false), then event handler is enabled at target node
- The addEventListener method will return event object to eventhandler function. The event object can be accessed using the keyword "Event"
- The address of the node that generated event will be stored in **current target**, which is property of **event object**

AN EXAMPLE OF THE DOM 2 EVENT MODEL

The next example is a revision of the validator.html document and validator.js script from
previous example, which used the DOM 0 event model. Because this version uses the DOM 2
event model, it does not work with IE8.

```
//validator2.html
<html>
<head> <title>Illustrate form input validation with DOM 2</title>
<script type = "text/javascript" src = "validator2.js"> </script> </head>
<body> <h3>enter your details</h3> <form action="">
<p> <label><input type="text" id="custName"/>
Name(last name, first name, middle initial)</label><br/><br>
<label><input type="text" id="custPhone"/>Phone (ddd-dddddddd)</label>
<br/><br> <input type="reset" /> <input type="submit" id="submitButton"/>
</p> </form>
```

```
<script type = "text/javascript" src = "validator2r.js"/>
</body> </html>
//validator2.js
function chkName(event) {
 var myName = event.currentTarget;
 var pos = myName.value.search(/^[A-Z][a-z]+, ?[A-Z][a-z]+, ?[A-Z]\.?$/);
if(pos != 0) {
alert("The name you entered (" + myName.value + ") is not in the correct
form.\n" + "The
correct form is: " + "last-name, first-name, middle-initial \n" + "Please go and
fix your
name");
myName.focus();
myName.select();
}
}
 function chkPhone(event) {
var myPhone = event.currentTarget;
var pos = myPhone.value.search(/^\d{3}-\d{8}$/);
if(pos != 0) {
alert("The phone you entered (" + myPhone.value + ") is not in the correct
form.\n" + "The
correct form is: " + "ddd-dddddddd \n" + "Please go and fix your phone
number");
myPhone.focus();
myPhone.select();
}
}
//validator2r.js
var c = document.getElementById("custName");
var p = document.getElementById("custPhone");
c.addEventListener("change",chkName,false);
p.addEventListener("change",chkPhone,false);
```

Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1:	Design Multimedia and web pages that include the CSS and standard textual components needed on web pages	2	2	2	1	3	1	1	0	1	2	1	1
CO2:	Create interactive websites using javascript and DHTML	2	1	2	1	3	1	0	0	0	2	1	1
CO3:	Write well-formed XML document for a given schema using DTD, XSLT	1	1	1	1	3	0	0	0	0	0	0	0
CO4:	Write simple CGI programs using PERL	2	2	1	1	2	0	1	0	0	1	0	0
CO5:	Implement a simple web application using PHP language	1	2	2	1	3	1	1	0	0	1	2	2
CO6:	Implement a simple web application using Ruby On Rails.	2	2	1	1	2	1	2	0	1	1	1	1

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PO1 - *Engineering knowledge*; PO2 - *Problem analysis*; PO3 - *Design/development of solutions*; PO4 - *Conduct investigations of complex problems*; PO5 - *Modern tool usage*; PO6 - *The Engineer and society*; PO7- *Environment and sustainability*; PO8 – *Ethics*; PO9 - *Individual and team work*; PO10 - *Communication*; PO11 - *Project management and finance*; PO12 - *Life-long learning*