WEB TECHNOLOGY & ITS APPLICATIONS.

Sem : VI SubCode : 18CS63.

MODEL QUESTION PAPER.

MODULE - 1.

1.a) What are 3 aims of HTML5? [4M]

b) Explain the need of cascade in CSS? Explain the 3 principles of cascade with suitable CSS script segments. [8M]

c) Explain 2 types of URL Referencing techniques with suitable scripts in HTML5. [8M]

[OR.]

2.a) List & Explain different selectors available in CSS [8M]

b) Discuss the HTML5 semantic structure elements. [8M]

c) List the different text properties with a description [4M]

MODULE - 2.

3.a) Explain different Form widgets created with the <input> tag. [8M]

b) Write HTML code for following table.

DAY	SEMINAR		
	SCHEDULE		TOPIC
	BEGIN	END	
MONDAY	8:00am	5:00pm	INTRODUCTION TO XML Validity & DTD & NS.
TUESDAY	11:00am	2:00pm	· XPATH
	11:00am	2:00pm	XSL Transformations.
	2:00pm	5:00pm	
WEDNESDAY	8:00am	5:00pm	XSL Formatting Objects

OR

4. a) Explain liquid layout design for websites with an example. List the [illegible]d layout benefits & limitations. [8M]

b) Explain different ways of positioning elements in css layout techniques. [8M]

c) What are the importance of responsive design? Explain Briefly. [4M]

MODULE-3

5. a) Write JS code that displays text "CORONA·VIRUS" with increasing font-size in the interval of 100ms in blue color, when font size reaches 50pt in teal color & should stop. [8M]

b) Explain the advantages & disadvantages of client side scripting. [6M]

c) With suitable diagram, explain APACHE modules in PHP. [6M]

OR

6. a) With suitable code segment, explain 2 approaches for event handling in JS. [8M]

b) Write PHP program to greet the user based on time. [8M]

c) Explain 2 methods in JS to access DOM nodes with examples. [4M]

MODULE-4

7. a) List & Explain different superglobal arrays. [8M]

b) Explain the different error handling methods, with suitable code segments. [8M]

c) How do you read or write file on server from PHP? Give Examples. [4M]

OR

8. a) Write PHP Program to create a class Employee with the following specifications: [8M]

Data members : Name , ID, payment.

Member functions : Read(getters) & write(Setters).

use the above specifications to read & print the information of 10 students.

b) Explain the support for inheritance in PHP with UML class diagram. [6M]

c) Explain 3 approaches to restrict file size in file upload with suitable code segments. [6M]

MODULE - 5

9. a) Explain different types of caching need to improve performance of web applications. [8M]

b) With suitable PHP script, Explain loading & procesing an XML document in JavaScript. [8M]

c) Explain creating & reading cookies with suitable PHP Scripts. [4M]

OR

10. a) Define AJAX. Explain AJAX request by writing UML Diagram. [8M]

b) Explain JS pseudo-classes with examples. [8M]

c) Explain converting a JSON string to JSON object in Javascript with suitable code Segments. [4M]

* * * * *

PROF. FARZANA . N. NADAF.

SEM : VI

SUB : WEB TECHNOLOGY & ITS APPLICATION

SUBCODE : 18CS63.

1.a) There are 3 main aims to HTML5 :-

i) Specify unambignously how browsers should deal with invalid markup.

2) Provide an open, nonproprietary programming framework for creating rich web applications.

3) Be backwards compatible with existing web.

b) Cascade refers in CSS to conflicting rule; precedence of child elements.

* There are 3 principles of Cascading are :-

i) Inheritance :- Many CSS properties affect not only themselves but their descendants also.

ii) Specificity :- Which style rule takes precedance when more than one style rule could be applied to the same element.

iii) Location :- The principle of location is that when rules have the same specificity, then the latest are given more weight.

Eg :-

```
① <head>
     <link rel="stylesheet" href="stylesA.css" /> ②
     <link rel="stylesheet" href="styleWW.css" />
④ <style>
     #example { color: orange;      ⑤ overrides
                color: magenta;
              }                     ⑥ overrides
   </style>
</head>  <body>
                <p id="#emple" style="color: red;">
```

c> URL Referencing technique are :-

i> Relative Referencing :- We need to successfully reference file within website. It requires a Relative referencing.

ii> ABSOLUTE REFERENCING :- When referencing a page or resource on an external site, a full ABSOLUTE REFERENCING is required. i.e

a> protocol (http://) b> domain name

c> any path d> filename of desired resource.

Eg:- <a href="example.html">
<a href="css/images/back.gif">
<a href="../images/about.html">
<a href="../css/images/back.gif">

2.a) The different selectors available in CSS are :-

* Element selectors * Class Selectors
* Id Selectors * Attribute "
* Pseudo Element & " class selectors * Contextual Selectors.

* Element Selectors → p { margin : 0;
padding : 0; }.

* Class Selectors → .first { font-style : italic;
color : red; }

* ID Selectors → #Comm { margin : 0;
Padding : 0; }.

* ATTRIBUTE Selectors → [title] { border-bottom : 2px dotted;
text-decoration : underline;
}

* Pseudo Selectors → a:link { color : blue; }
a:active { background-color : teal;

* Contextual Selectors :- #main div p:first-child
{ color :green ; }

b) HTML5 Semantic Structure Elements are :-

* HEADER & FOOTER :- It contains the headings for a section, along with content such a material. Footer contains less important material such as navigation, copyright notices.

* ~~HEADING~~ GROUPS :- <hgroup> element can be used in contexts other than header.
<HEADER > <HGROUP> <H1> - - - </H1> </HGROUP>.

* NAVIGATION :- <nav> element represents a section of page that contains links to other pages.

* ARTICLES & SECTIONS :- Articles element represents a section of content that forms an independent part of a document. Section element represents section of a document.

* FIGURE & FIGURE CAPTION :- <FIGURE> element repⁿ some flow content, optionally with a caption.
<FIGCAPTION> used to annotate illustration, diagrams, photos, code listings.

* ASIDE :- It is similar to <figure> element i.e used for making up content i.e separate from main content on page.

```
Eg:- <BODY>
        <HEADER>
            <HGROUP>....
            </HGROUP>
            <nav>---</nav>
        </HEADER>.
                    <SECTION>---</SECTION>
                    <article>.---</articles
                    <figure>
                      <img .----/>
                    <figcaption>---</figures
                    <aside> ---</asides
                    <footer>.---</footer>
```

c> The different text-properties are 6-

a> text-decoration. c> letter-spacing
b> text-align d> text-decoration.
e> line-height f> word-spacing.

3> a> Form Widgets created are <input> are 6-

text : Creates single line text entry box.
<input type="text" name="a" />.

textarea : creates multiple line text entry box.
<textarea rows="3" --->.

password : single-line text with bullets/some characters.
<input type="password" />.

search : single-line text entry suitable for search string. <input type="search" ---- />.

email : which takes an email address.
<input type="email" .--- />.

tel : which takes an entering a telephone number.
<input type="tel" --- />

url : creates a single line entering URL.
<input type="url" ---- />.

b>
```
<HTML>
<BODY> <TABLE BORDER="1" CAPTION="Diff LANG">
<TR> <TH ROWSPAN="3">. DAY </TH>
    <TH COLSPAN="3">. SEMINAR </TH> </TR>
<TR> <TD COLSPAN="2"> SCHEDULE </TD>
  <TD ROWSPAN="2"> TOPICS </TD> </TR>.
<TR> <TD>. BEGIN </TD> <TD> END </TD> </TR>.
<TR> <TD> MONDAY </TD> <TD> 8:00 AM </TD>
  <TD> 5:00 pm </TD> <TD ROWSPAN="2">. INTRODUCTION
     TO XML </TD> </TR>.
<TR> <TD> VALIDITY : DTD & NG </TD> </TR>
```

```
<TR> <TD ROWSPAN='3"> TUESDAY </TD >
     <TD> 11:00 AM </TD><TD> 2:00 pm </TD>
     <TD ROWSPAN = "2" > XPATH </TD > </TR>.
 <TR> <TD> 11:00 am </TD><TD> 2:00 pm </TD></TR>
<TR> <TD> 2:00 pm</TD> <TD> 5:00 pm </TD>
     <TD> XSL transformations </TD> </TR>.
 <TR> <TD> WEDNESDAY </TD> <TD> 8:00 am </TD>
     <TD> 5:00 pm </TD> <TD> XSL Formating
          Objects </TD> </TR>. </TABLE >
 </BODY> </HTML>.
```

4.a) The LIQUID LAYOUT :-

* It's deal with the problem of multiple screen sizes. It is also called FLUID LAYOUT.

* It adapts to different browser sizes so there is neither wasted white space nor any need for Horizontal scrolling.

* DisAdvantages :- i) Difficult to create coz some elements, such as images, have fixed pixel sizes. ii) As screen grows or shrinks dramatically, in that line length may become too long or too short.

iii) Creating a usable liquid layout is generally more difficult than creating a fixed layout.

ADVANTAGES :- BENEFITS :-

* Liquid LAYOUT is that it adapts to different browser sizes, so there is neither wasted white space nor any need for horizontal scrolling.

* The ~~CSS~~ % values in CSS are a % of current browser width, so layout in which all widths are expressed as % should adapt to any browser size.

b) The different ways of positioning elements in CSS layout ~~benefits~~ techniques. are :-

* Relative Positioning :-

It is an element displaced out of its normal flow position & moved relative to where it would have been placed.

* ABSOLUTE POSITIONING :-

It is removed completely from the normal flow. The space is not left for the moved element, as it is no longer in the normal flow.

eg :-
```
figcaption { background-color : #EDEDDD;
             padding : 5px; width : 150px;
             position : absolute;
           }

figure { top : 150px; left : 200px;
         position : relative;        }
```

*Z-INDEX :- Each positioned element has a stacking order defined by the z-index property. Items closest to viewer have a larger z-index value,

Eg:-
```
figure {
    position: absolute;
    top: 150px;
    left: 400px;
    z-index: 1;
}
```

```
figcaption {
    position: absolute;
    top: 90px;
    left: 140px;
    z-index: -1;
}
```

* Fixed Position :-

It is a type of absolute positioning, except the positioning values are in relation to viewport.

* It is commonly used to ensure that navigation elements or advertisements are always visible.

eg:- `figure { position: fixed; top: 0px; left: 0px; }`

4.c) The importances of responsive design are :-

* Liquid layouts ①
* Setting viewports via <meta> tag ②

③* Scaling images to viewport size ⇒ `img{max-width: 100%; }`

④* Customizing CSS for different viewports using media queries.

② `<meta name="viewport" content="width=device-width"/>`

④ eg:-
```
@media only screen and (max-width: 480px)
    { - - - - - }
```

① It is the one which most elements have their width specified in %.

`img { max-width: 100%; }`

```
<BODY> <p id="demo" > </p>.
5. a) <script type="text/javascript">
  var r1 = setInterval(inTimer ,1000);
  var fs = 5;    var ids = document.getElementById("demo");
  function inTimer()
  {
    ids.innerHTML = "CORONA VIRUS";
    ids.setAttribute('style', "font-size:"+fs+"px;
                                              color:blue");
    fs+=5;
    if(fs >=50) { clearInterval(r1);
                  r2 = setInterval (deTimer ,1000);
    }             }
  function deTimer()
  {
    fs -=5;
    ids.setAttribute('style',"font-size:"+fs+"px;
                                   color: Teal");
    if(fs ===5) {
               clearInterval (r2);
             }
  }
  </script> </BODY> </HTML>.
```

b). Client-Side Scripting :-

Advantages :-

* Processing can be offloaded from the server to client machines, ∴ reducing the load on server.

* Browser can respond more rapidly to user events than a request to remote server, which improves user experience.

* JS can interact with downloaded HTML in a way that server can't, creating a user experience more like desktop s/w than simple HTML.

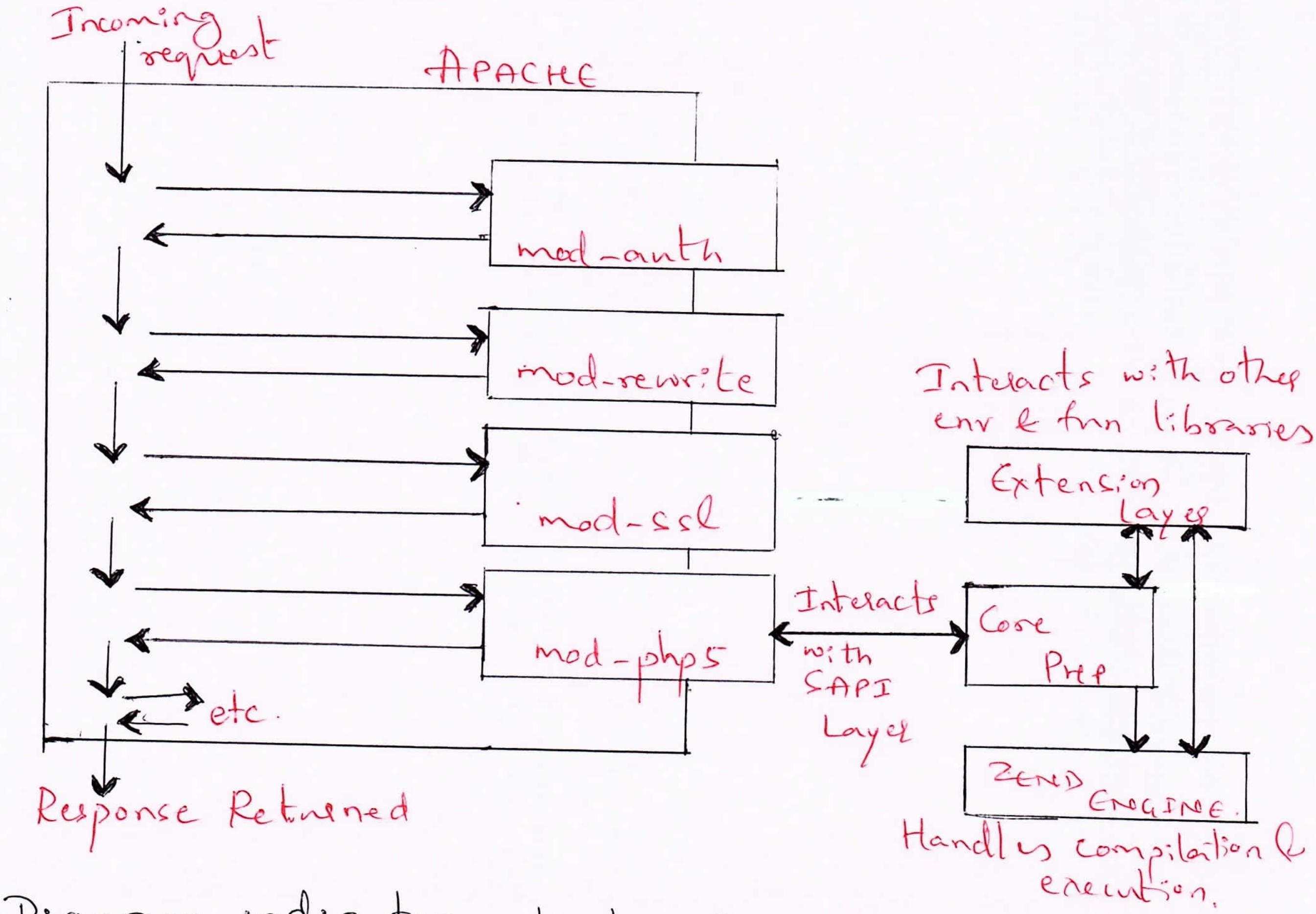
*Disadvantages of Client-side Scripting:-

⇒ There is no guarantee that the client has JS enabled, it means required functionality must be housed on server.

⇒ The idiosyncrasies betⁿ various browsers & OS make it difficult to test for client configuration

⇒ Javascript heavy web-applications can be complicated to debug & maintain.

c) APACHE Modules in PHP:-



* Diagram indicates that; PHP usually installed as an APACHE module.

* PHP_module mod-php5 is sometimes refered to as SAPI layer. (Server Application Prog. Interface).

* SAPI handles the interaction between PHP & web server environment.

*ZEND Engine is virtual machine that processes & executes PHP files. It also handles memory mangement garbage collection & dispatching fun$^{\underline{n}}$ calls to modules outside of PHP.

6. a) The 2 approaches for event handling in JS are:-

*Inline -Event Handler :-

⇒ JS events allow programmer to react to user interactions.

⇒ inline JS calls are intuitive.

eg:-
```
<div id="exp1" onclick="alert('hello')" >
        click for pop-up </div>
```

⇒ When user clicks the <div>, event is triggered & alert is executed.

* LISTENER APPROACH.:-

There are 2 functions to add an event & delete an event.

⇒ Main advantage of this approach is that the code can be written anywhere, including an external file.

⇒ Limitation is only one handler can respond to any given element event.

eg:-
```
var greet = document.getElementById('exp1');
  greet.addEventListener('click', alert('Good Mor');
  greet.addEventListener('mouseout', alert('GoodBye');
  - -
  - -
  greet.removeEventListener(  );
b> <?php
   $t. = date('H');
   $t2 = date('e');
   if($t <="12") { echo "Good Morning"); }.
  else if ($t >="12" && $t.<"17")
   { echo " Good Afternoon" ; }
  else echo "Good Night"; }        ?>.
```

c> The two methods in JS to access DOM nodes are :-

* getElementId : using id of a element, we can access the element data. Since ID must be a unique in an HTML document, It returns a single node rather than a set of results.

```
eg:- var v1 = document.getElementById("demo");
         v1. style. display = "block";
         v1. style. border = "solid 1px";
```

* getElementByTagname :- It returns a NodeList of elements whose tagname matches the passed name parameter.

```
eg:- var mylink = document.getElementsByTagName("pe");
 for (i=0; i<mylink.length; i++)
  { if(mylink[i]. className == "std_class") {
         mylink[i]. onClick = function(){
                  this.stylebackgroundcolor = "#f00"; }
    }
  }.
```

7. a) The supeeglobal arrays are :-

$GLOBALS, $_COOKIES, $_ENV, $_FILES, $_GET, $_POST, $_REQUEST, $_SESSION, $_SERVER.

$GLOBALS :- array for storing data that needs supeeglobal scope.

$_COOKIES :- array of cookie data passed to page via HTTP request. It gives more information about user logged-in, ~~who~~ how many times hit the pages.

$_ENV :- Server environment data.

$_FILES :- file items uploaded to server

$_GET :- Array of Query string data passed to server via the URL. The interaction with user input can be accessed by the 2 methods GET & POST of HTML form elements.

$_POST :- Array of Query string data passed to server via HTTP header.

$_SERVER :- It contains a variety of information. It some of the info contained within HTTP request headers sent by the client.

eg:- echo $_SERVER["SERVER_NAME"]. "<br/>";
echo $_SERVER['SERVER_SOFTWARE"]. "<br/>";

b). The different error handling methods are :-

* PROCEDURAL ERROR HANDLING :-

⇒ Programmer needs to explicitly test for error conditions after performing a task might generate an error.

* It may result in a great deal of code duplication.

Eg:-
```
$conn = mysqli-connect(DBHOST, DBUSER, DBPASS, DNAME);
$error = mysqli_connect_error();
if($error != null) { . ___ . }.
```

*OBJECT_ORIENTED EXCEPTION HANDLING :-

→ When a runtime error occurs, PHP throws an exception. This exception can be caught & handled either by the function, class or page that generated the exception or by code that called function or class.

→ PHP uses the try---catch progⁿ construct to progⁿtically deal with exception at runtime.

Eg:-
```
function throwException($msg=null, $code=null)
  { throw new Exception($msg, $code); }
 try {   $conn = mysqli_connect( ---- ) or
      }   throwException("error");
 Catch(Exception e) {    echo 'Caught exception'.on line'.
                 }           $e-getLine();
 finally { --- }
```

c) There are 2 technique for read/write file in PHP.

* Stream Access :- to read a small portion of the file at a time. It is the most memory efficient approach when reading large files.

* To read a content from file, we use fgets(), fopen(), fread() & fgetsc().

eg:- while($line =fgets($f) {
 $ln++; }
 pf("%2d", $ln); }.

All-In-Memory :-

⇒ We can read the entire file into memory.

⇒ It is easier to use, at cost of relinquishing fine-grained control.

eg:- $f = file_get_contents(FILENAME);
 file_put_contents(FILENAME, $w);

8.a)
```
<?php
class EMPLOYEE {
                private $name;
                private $ID;
                public $pay;

  function __construct($fn, $id, $pay){
        $this->setName($fn);
        $this->setID($id);
        $this->setPay($pay);     }.
  public function getName() {return $this->name; }
 public function getID(){ return $this->ID; }
 public function getPay(){ return $this->pay; }
 public function setName($n) {$this->name = $n} }
 public function setID($d) { $this->ID = $d; }
 public function setPay($p) { $this->pay = $p; }
}
  EMPLOYEE[] a=new EMPLOYEE[10];

    $a[0]=new EMPLOYEE("FOO", 2124, 70,000);
    $a[1]=new EMPLOYEE("MOHIT", 9584, 80,000);
    - - -
 echo "EMPLOYEE INFO n";
 For ($i=1; $i<=10; $i++ { echo $a[$i]->outputAsTable();
}. }
```

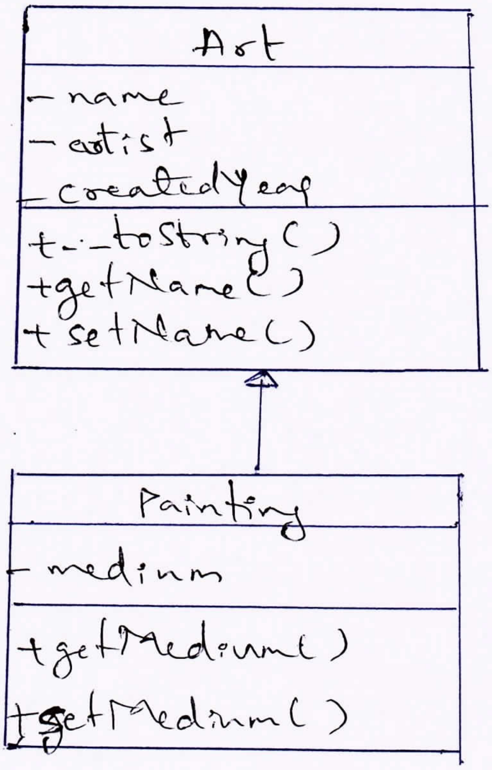
b> INHERITANCE :-

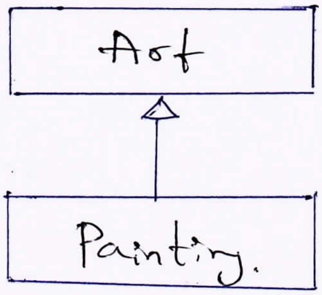
* It is the key concept of Object oriented design & programming. It allows us to create new PHP classes that reuse, extend & modify the behavior of a defined class in another PHP class.

* PHP allows only one class to inherit at a time.

* PHP class is defined as a subclass by using the extends keyword.

class Painting extends Art { .... }





```
eg:- class Art
     {
       = _
     }
  class Painting extends Art {--public function foo() {
                                  -- --
  $p = new Painting();             $w = parent::getName();
                                   $x = parent::getOriginal();
                                  }
```

c) The 3 approaches to restrict file size in file upload are :-

* via HTML (1) in input form
* via JS in input (2) form.
* PHP (3) coding.

Eg:- 1)

```
foreach($_FILES as $fk => $fArr)
{
      if ($fArr["error"] != UPLOAD_ERR_OK)
        {  echo "Error". $fk ."has eror" . $fArr["error"];
        }
}
else {   echo $fk ."uploaded successfully"; }
```

* Limiting upload file size via HTML.

```
<HTML><FORM enctype="multipart/form-data' method='post'>
 <input type="hidden" name="MAX_FILE_SIZE"
                                   value="1000000" />
<input type='file' name='f1'/>
<input type='submit' />      </form>.
```

* Limiting upload file size via JS

```
<script>
 var file = document.getElementById('file1');
 var m_size= document.getElementById("max_file_size").value;
 if (file.files && file.files.length == 1) {
     if (file.files[0].size > m_size) {
       alert ("file must b less than" + (max_size/1024) +"KB");
       e.preventDefault ();     }   }    </script>.
```

* Limiting upload file size via PHP.

```
$max_file_size = 10000000;
foreach($_files as $fk => $fArr) {
    if($fArr["size"] > $max_file_size) {
       echo "Error" . $fk . " is too big";  }
   PL("%s %s %.2f KB" $fArr["size"]/(1024);
```

9.a) The different types of caching used to improve performance of web applications are :-

* Page O/P Caching :- which saves the rendered output of a page or user control & reuses the output instead of reprocessing the page when a user requests the page again.

*It is useful for pages whoes content does not change frequently but which require significant processing to create.

* 2 models ~~for~~. are :- < full page caching
partial page caching.

*The full page caching ,entire contents of a page are cached.

* In partia page caching, only specific parts of page are cached while other parts all dynamically generated in normal manner.

*APPLICATION DATA CACHING :-

⇒ Limitation of page output caching is performance gains only if entire page is same for numerous requests

⇒ It is a page, which will place commonly used collections of data require time-intensive queries from DB or web server into cache memory.

⇒ A widely available free PECL extension called memcache is used to provide this caching.

⇒ It should be stressed memcache should not be used to store large collections.

b). Loading & Procening an XML document via JS.

eg:-
```
<script>
 if(window.XMLHttpRequest){
    xmlHttp=new XMLHttpRequest(); }
else{ xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
xmlhttp.open("GET","art.xml", false);
xmlhttp.send();   xmlDoc=xmlhttp.responseXML.
paintaings=xmlDoc.getElementByTagName("painting");
if(paintangs){  for(i=0;i<paintings.length; i++)
              {  alert("id= "+paintings[i].getAttribute("id"));
    title=paintings[i].getElementsByTagName("title");
  if(title){  alert("title = "+title[0].textContent);
                  }
 } }</script>.
```

* XML procening in PHP is divided into 2 basic styles :-

⇒ In-Memory. Approach :- which involves reading the entire XML file into memory into some type of data Structure with functions for accening & manipulating the data.

⇒ Even or Pull Approach :- which reading a file in a few elements or lines at a time, therefore avoiding the memory load of large XML files.

⇒ JS loads the entire document into memory where it is transformed into a hierarchical tree data structure.

→ JS supports a variety of node traversal fun^n as well as properties for accessing information within an XML node.

c). PHP provides mechanisms for writing & reading Cookies. PHP ~~c~~ookies are created using the setcookie() function & retrieved using the $_COOKIES superglobal associative array.

```
eg:- <?php
          $etm = time() + 60*60*24;
          $nm = "Username"; $v = "Mohit";
          setCookie($nm, $v, $etm);   ?>
```

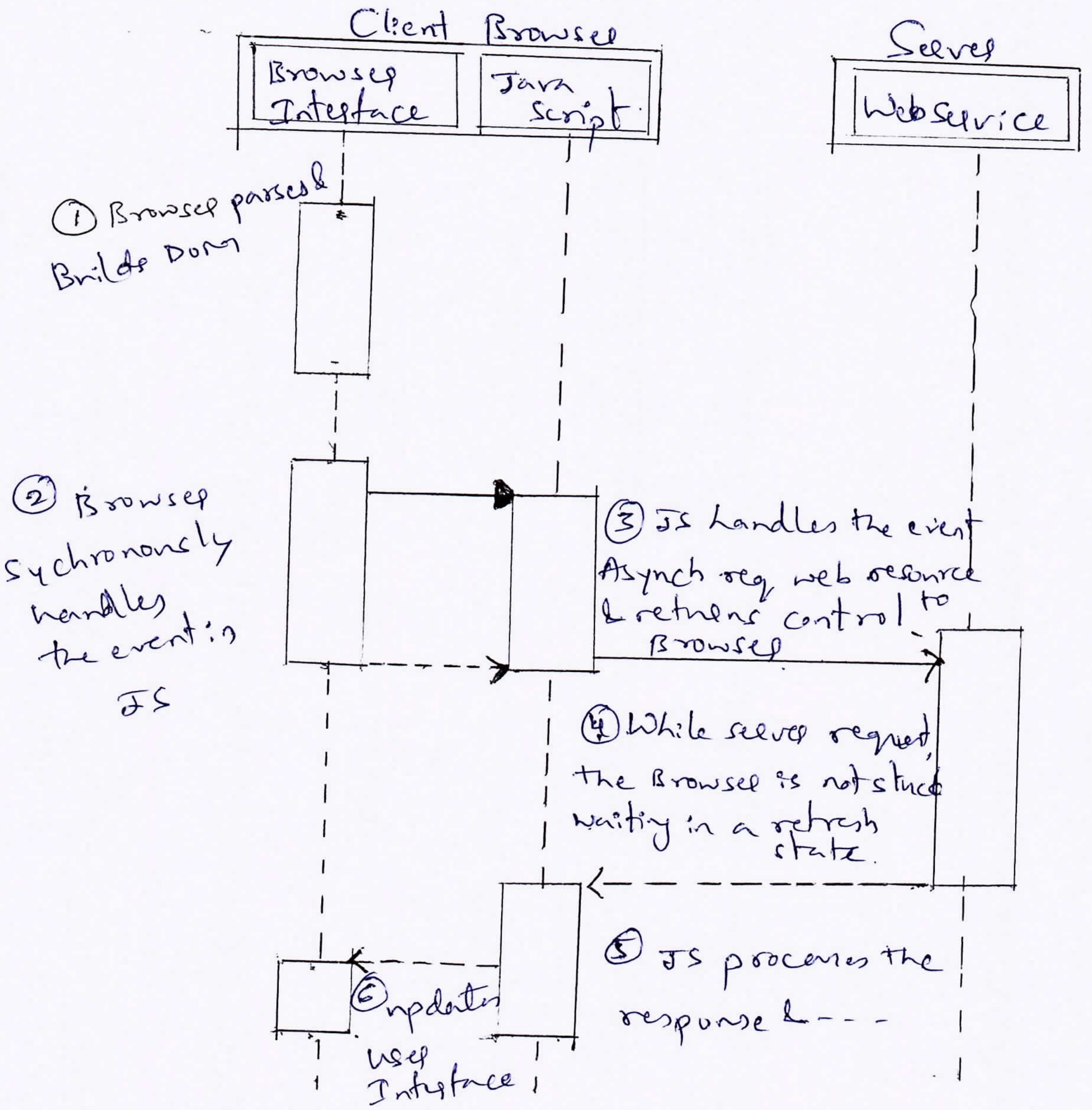
* Before reading a cookie, we must check to ensure that the cookie exists. In PHP, If cookie has expired, then client Browser would not send anything & $_COOKIE array would be blank.

```
<?php   if(!isset($_COOKIE['username']))
          {   //no valid Cookie    }
       else {
                echo " Username Retrieved from Cookie:";
          }  echo $_COOKIE['username'];
?>
```

10. a) Asychronous JS with XML (AJAX) is term used describe a paradigm that allows a web browser to send messages back to server without interrupting the flow.

→ The UML sequence diagram of AJAX request

* Responses to asynchronously requests are caught in JS as events. These events subsequently trigger in UI or make additional requests.


Client Browser
Browser Interface
Java Script
Server
Webservice
① Browser parses & Builds DOM
② Browser sychronously handles the event in JS
③ JS handles the event Asynch req web resource & returns control to Browser
④ While server request, the Browser is not stuck waiting in a refresh state.
⑤ JS processes the response & - - -
⑥ updates user Interface

* The difference from the typical synchronous requests which require the entire page to refresh in response to a request.

b). JS doesn't support most of the object oriented features. Instead it define pseudo-classes through a variety of interesting & nonintuitive syntax.

⇒ JS include increased code reuse, better memory management & easier maintainance.

⇒ Example to define Die pseudo-class with an internally defined method.

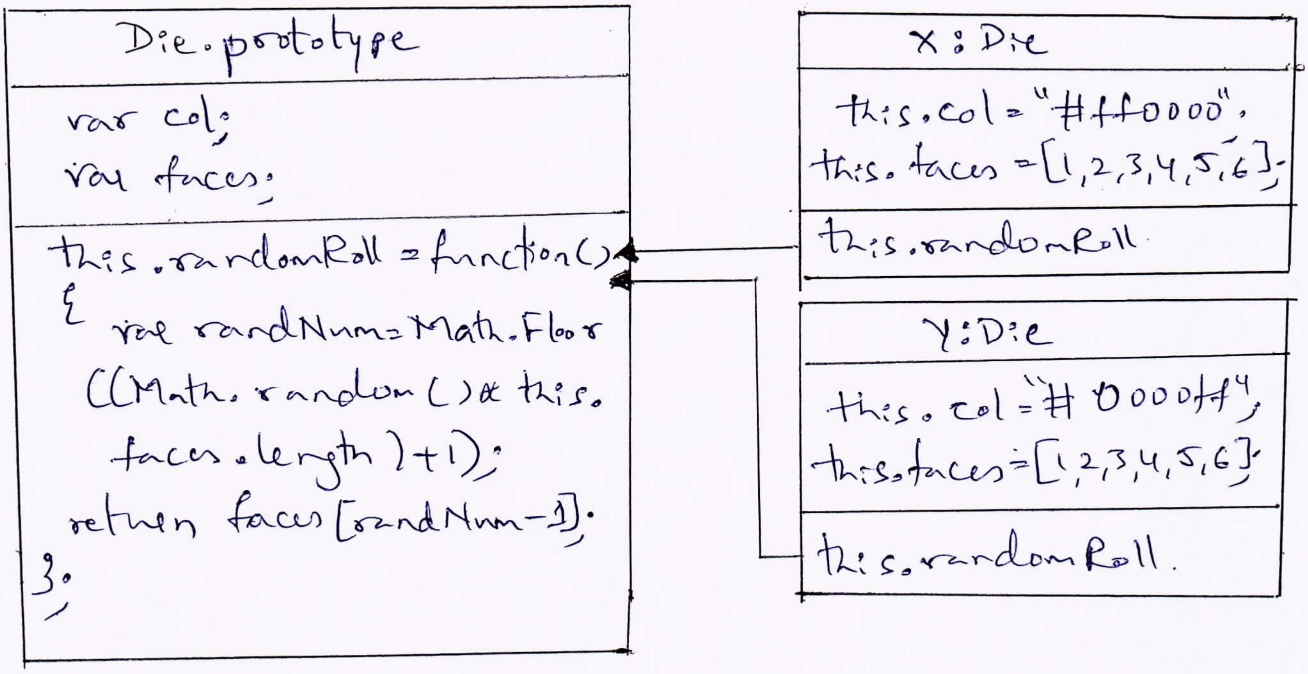
```
function Die(col) {
                    this.color = col;
                    this.faces = [1,2,3,4,5,6].
Die.prototype.
  ^ .randomRoll = function () {
    var randNum = Math.floor((Math.random ()*this.faces.
                  length +1);
    return faces [randNum-1];
  };
};
```

* Adding method inside of a class definition is by assigning an anonymous function to a variable.

* Prototype are an essential syntax mechanism in JS, & used to make JS behave more like OO Language.

* The prototype properties & methods are defined once for all instances of an object.

* Figure illustrates JS prototypes as pseudo-classes.



```
Die.prototype
var col;
var faces;
this.randomRoll = function()
{
  var randNum=Math.Floor
  ((Math.random ()* this.
   faces.length )+1);
 return faces [randNum-1].
};
```

```
x : Die
this.col="#ff0000".
this.faces =[1,2,3,4,5,6].
this.randomRoll
```

```
y:Die
this.col="#0000ff",
this.faces=[1,2,3,4,5,6].
this.randomRoll.
```

c) Converting a JSON. string into a PHP object is straightforward :

→ Ex :-

```
<?php
    $text = '{"artist":{"name":"Foo","nationality":
                                          "INDIAN"}}';
   $anObj = json_decode($text).
   echo $anObj->artist->nationality;
   $anArr = json_decode($text, true);
   echo $anArr['artist']['nationality']. ?>.
```

* JSON-decode function retuen 'either PHP Object or an associative array.

* JSON data is Coming from exteenal souece, we should always check for parse errors via the json_last_error() function.

```
<?php
      $text = '{"artist":{"name":"Foo",
                          "nationality":"INDIAN"}}';
    $anObj = json_decode($text).
    if(json_last_error() == JSON_ERROR_NONE)
    {
       echo $anObj->artist->nationality; }
 ?>.
```