PhP

PHP Arrays

An array is a special Variable, which can hold more than one value at a time.
In PHP, array() function is used to create an array.

```
<? php
$cars = array ("volvo", "BMW", "Toyota");
echo $cars[0]; // (volvo)
?>
```

Three types of array →

1) Indexed Array. → arrays with a numeric index.

```
$cars = array ("volvo", "BMW", "Toyota");
```

0th index (assigned automatically)

or assigned manually → $cars[0] = "volvo";

Loop through an Indexed Array

for loop

```
<?php
$cars = array("BMW", "volvo");
$arrlength = count($cars);

for ($x = 0; $x < $arrlength; $x++)
  {
  echo $cars[$x];
  echo "<br>";
  }
?>
```

for each loop works on arrays and objects only. and is used to loop through each key/value pair in an array.

```
foreach ($array as $value)
  {
  stmt;
  }
```

eg→

```
foreach ($cars as $Value).
  {
  echo "$value <br>";   // BMW volvo.
  }
```

Or

```
foreach ($array as $key => $value)
  {
  }
```

eg→

```
{ echo "{$key} => {$value}";
}   // 0 => BMW.
```

(2) Associative Arrays

are arrays that use named keys that you assign to them.

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

access ⇒ `echo $age['Peter'] ; // 35`

Loop through

```
foreach ($age as $x => $x-value)
{
    echo "Key =" . $x . ", Value = " . $x-value;
}
// Key = Peter , value = 35
```

— each()

each function is meant to be used in loops over collections such as arrays. each time through the array, it returns the current element's key and value & then moves to next Element.

To handle a multiple item return value from an array use list fn. { assign the two return values from each to separate variables }

```
while ( list ($key, $value) = each ($cars))
    { echo "Key: $key; Value: $value \n"; }
```

⇒ PHP → Hypertext Preprocessor.
widely used, open-source scripting language.
PHP scripts are executed on servers.

Why PHP →
- runs on various platforms (Window, Linux, Unix,) etc.
- compatible with all servers (Apache, IIS etc)

Syntax →

```
<? php
// — stmt ends with a colon ;
?>
```

Case Sensitivity
- In PHP, all keywords, classes, functions, and user-defined fns are NOT case sensitive.
- all variables names are case-sensitive.

Variables
- starts with $ sign, followed by name.

- PHP is a loosely typed language ie. we did not have to tell PHP which data type the variable is. It automatically converts the variable to the correct data type, depending on its value.

- all variable names are Case Sensitive.

→ VARIABLES

a variable starts with $ sign, followed by the name.

```
$txt = "Hello world";   // string type
$var = 5;               // Integer type
```

- PHP is a loosely typed language
  meaning that PHP converts automatically the variable to the correct data type, depending on its value.

→ Variable SCOPE
↳ part of script where the variable can be referenced / used.

(1) Local → A variable declared within a function has a local SCOPE & can ~~be~~ only be accessed within that function.

```
<? php
    function myTest()
    {   $x = 5;
        echo $x;   // 5
    }
    myTest()
    echo $x;   // error cannot be accessed outside
?>
```

(2) GLOBAL

Variable declared outside a function has a Global scope and can only be accessed outside a function.

```
<?php
    $x = 5;

    function myTest()
        {
            echo $x; // error
        }
    echo $x; // 5
?>
```

- GLOBAL KEYWORD

used to access a global variable from within a function.

```
<?php
    $x = 5;
    function myTest()
        {
            global $x;
            echo $x; // 5
        }
```

(use this keyword)

PHP stores all global variables in an array called $GLOBALS[index] (holds the name of the variable)

also
- accessed from within the function.

```
$GLOBALS['y'] = $GLOBALS['x'] +
                $GLOBALS['y'];
```

STATIC

↳ when we want a local variable not to be deleted.

```
<?php
    function myTest()
    {
        static $x = 0;
        echo $x;
        $x++;
    }
    myTest();    // 0
    myTest();    // 1
    myTest();    // 2
?>
```

each time the function is called, that variable will still have the information it contained from the last time the function was called.

⇒ **Echo / Print**

- both / used to output data to the screen.

- has no return value.
- can take multiple arguments
- fast.

- Has a return value of 1
- Can take only 1 argument

- echo "<h2> - - - </h2>";
- echo "Hello", "world";
- echo $txt1;

⇒ **PHP Datatypes**

1. String → sequence of 'chrs' or " "
2. Integer → non decimal no.
3. float → decimal or exponential.
4. Boolean → True or false
5. Array → stores multiple values in single variable.
6. Object → data type which stores data & info. on how to process data.

```
Q→
    class Car {
        function Car() {
            $this->Model = "VW";
        }
    }
$herbie = new Car();
echo $herbie->model;
```

7. Null Value → Null
8. Resource

⇒ String functions

1. strlen (" ") → length of a string.
2. str_word_count (" ") → word count.
3. strrev (" "); // Reverse
4. strpos (" " , " " , ") → (If found char pos else false)
   - " " → to be searched
5. str_replace (" " , " " , " ") ;
   - " " → to be replace
   - " " → with
   - " " → string
6. htmlspecialchar() → converts characters to HTML entities.
7. strcmp (" " , " ") ;
   - " " → string1
   - " " → string2
   - (case-sensitive)
   - 0 → equal
   - <0 → string1 < string2
   - >0 → string1 > string2
8. trim (" " , " ") ;
   - " " → string
   - " " → what to replace

⇒ Constants

- is an identifier for a simple value.
- global across the entire script.

define (name , value, case-sensitivity)
default → false.

⇒ **Operators** → Arithmetic, Assignment, Relational, (Relational) Logical, Bitwise, ternary (condition), MOD
⇒ Operator Precedence & associativity.

- arithmetic → +, −, *, /, %, ** ($x **$y = x^y)
- Assignment → =, +=, −=, *=, /=, %=
- Comparison → (1) == (Equal)
  (2) === (Identical) Value/type
  (3) != (Not equal) or <>
  (4) !== (Not Identical) (value & type)
  (5) >, <, >=, <=
- Increment / Decrement (++$x, $x++, --$x, $x--)
- Logical (1) and (both true) / &&
  (2) Or / ||
  (3) XOR
  (4) Not !
- Conditional
  ?: If condition is True ? Then value X : Otherwise value Y.

⇒ Unary
Binary
Condition (ternary)
Assignment

Precedence of PHP
determins the grouping of terms in an expression. This affects how an expression is evaluated.

⇒ functions not case sensitive

block of stmts that can be used repeatedly in a prg program.

Default Argument Value

function setHeight ($minheight = 50)
{
}

setHeight (350)
setHeight ( ); // uses minheight = 50;

⇒ conditional

<?php
$a = 10;
$b = 20;
$result = ($a > $b) ? $a : $b;
IF cond^n is true assign a otherwise (b)
?>

P.H.P

(1) PHP introduction (inventions, tool, s/w Reqms.)
(2) Scope of PHP
(3) Syntax [constants, variables]
(4) data types, Expressions, scope of variables
(5) Operators
(6) Operator Precedence & associativity
(7) If/Else, Switch, for, while, Goto, Break, continue, exit
(8) functions.
(9) Call by value / call by reference } scope
(10) String manipulation
(11) Regular Expressions
(12) Arrays.
(13) forms

PHP injections
→ passing vulnerabilities into your webapp. It is a malicious code injection technique.

PHP sort functions for Arrays

(1) sort()
→ sort array in ascending order
(2) rsort()
→ descending order
(3) asort() | arsort() - descending
associative - ascending - (value)
(4) ksort() | krsort() → descending
associative - ascending - (key)

Array functions

(1) array_chunk ($array, size of each chunk, preserve key (optional))

↓ true or false ↓ default

Reindexes the chunk numerically

```
$cars = array ( "Volvo", "BMW", "Toyota", "Honda", "Mercedes" );
print_r (array_chunk ($cars, 2));
```

Output: Array ([0] ⇒ Array([0] ⇒ Volvo [1] ⇒ BMW)
[1] ⇒ .

(2) array_fill (index, number, value);

index ↓ first index of returned array

number ↓ no. of elements to insert

value ↓ value to fill array

```
$cars = array_fill ( 3, 2, "Blue" );
```

output → Array ([3] ⇒ Blue [4] ⇒ Blue);

(3) array_filter (array, callbackfunction);

↓ (returns an array with filtered values)

↓ passes each value of the ~~function~~ input array to the callback fun.

(Keys are preserved)

④ array_flip (array) → keys → values
values → keys

array map (fn, array)

⑤ array_map (function, array1, array2, array3...)
(Σ operands)

(makes changes in original array) (to all values)

```
$ai = array("Dog", "Cat");
$a2 = array("Horse", "Kitten");

$a = array_map(null, $a1, $a2);
```

output ↓

Array ([0] ⇒ Array ([0] ⇒ Dog [1] ⇒ ~~Cat~~ Horse)
[1] ⇒ Array ([0] ⇒ Cat [1] ⇒ Kitten))

keeps key same (value changes)

⑥ Array_merge (array1, array2 ...)

If two array have same keys, last one overrides

⑦ array_rand (array, number) (no. of random values to be generated)

```
$a = array(  );
$key_random = array_rand($a);
```

→ echo $a [$key_random[0]]

(8) array_replace (array1, array2, ...) — optional

```
$a1 = array("red","green");
$a2 = array("blue","yellow");
array_replace($a1, $a2);
```

Output → Array ([0] ⇒ blue [1] ⇒ Yellow)

(9) array_slice (array, start, length, preserve) — (optional optional)

(new Indexes) ↙

start: 0 (first element) ; -2 (starts at second last element of the array)

preserve: Default false reset keys

```
$a = array("1","2","3","4");
array_slice($a, 2);
```

Output →

Array ([0] ⇒ 3 , [1] ⇒ 4);

(10) array_splice (array, start, length, array) — optional

↓ (removes selected elements from an array & replaces with new elements)

(new indexes) ↙

(changes are done in this array)

```
$a1 = array("a" ⇒ "1", "b" ⇒ "2", "c" ⇒ "3")
$a2 = array("a" ⇒ "one", "b" ⇒ "two")
array_splice($a1, 0, 2, $a2)
```

output →

Array ([0] ⇒ one [1] ⇒ two [2] ⇒ 3)

IF length = 0 ⇒ do not remove any element but add a2 at start position & then continue

array_walk (array function)

(11) array_walk (array , function , parameter ...)
optional ↑

```
function a ( $value , $key )
{ echo " $key $value ";
}
$a = array ( "a" => "red" , "b" => "green" , "c" => "blue" );
array_walk ( $a , " function a " ) ;
```

output →

→ fn. runs each array element in a user defined fn. The array's keys & values are parameters in fn.

(12) ~~Php~~
count ($array)
↳ no. of Elements in array.

(13) $ array_diff ($a1 , $a2) ⇒ (a1 - a2)

(14) array_intersect ($a1 , $a2)

(15) array_fill_keys (~~$o~~ array , ~~key~~ "Value") ;

```
$a = arrays ( "a" , "b" , "c" ) ;
$a1 = array_fill_keys ( ~~$key~~ $a , "Blue" ) ;
```

⇒ Array {
[a] ⇒ blue
[b] ⇒ blue
[c] ⇒ blue
}

(16) array_key_exists ("key", $array) → true or false

(17) array_keys ($array)

(18) mt_rand()
round()
ceil()
floor()
json_encode()
json_decode();

} Important

strtolower
strtoupper ($str)

⇒ json_decode / encode

$arr = array ("a" ⇒ "1", "b" ⇒ "2", "c" ⇒ "3")
$x = json_encode ($arr);

output →

{"a":"1","b":"2","c":"3"}

$a = json_decode ($x);
associative array

⇒ mt_rand (0, 100);
start (end)

```
$heroes[] = 'Captain Ann';
array_pop($heroes); // }
```

72

⇒ Cookies

- A cookie is often used to identify a user.
- Stored on user's computer in text file format
- Cannot hold multiple variables
- We can set expiry for a cookie

⇒

Create cookies

- setcookie (name, value, expire, path, domain, secure, httponly); (required)

→ must appear before <html> tag.

```
<?php
    $cookie_name = "user";
    $cookie_value = "John Doe";
    setcookie ($cookie_name, $cookie_value, time() +
                (86400 * 30), "/"); // 86400 = 1day
?>
```

("/" → is available in entire website)

```
<Html>
<body>
<?php
  if (!isset ($_COOKIE[$cookie_name]))
    {
      echo "cookie named '". $cookie_name. "' is not set!";
    }else {
      echo "Cookie" . $cookie_name. " is set ";
    }
?>
```

⇒ to delete → setcookie ("user", "", time() - 3600);

→ enabled → if (count ($_COOKIE) > 0) (enabled)

to one hr to go

new page → instead, they are retrieved by
session_start();

⇒ SESSIONS

- a session is a way to store information (in variables) to be used across multiple web pages.
- Info is stored on server
- can hold multiple variables.
- remains active as long as browser is open.
- cannot access data stored in sessions.

→

```
Start -   <?php
          session_start();
          ?>
```
// top of the page

```
<body>
<?php
    $SESSION["favcolor"] = "green";
    $_SESSION["favanimal"] = "cat";
?>
```

(favcolor: variable, green: value)

Destroy / unset-

```
session_unset();   // remove all session variables.
session_destroy()  // destroy session.
```

⇒ Diff. b/w $var & $$var

Variable Interpolation

```
$a = "abc";
$$a = "5";
echo $abc;   // 5
```

⇒ String

① chop ($string, optional "word") ;
removes characters from right end of a string.

rtrim()

If chop ($string)
// removes → "\0", "\t", "\n" & white space

② echo (str)
$color = "red" ;
difference of single & double quotes.
echo "Roses are $color" ; // Roses are red
echo 'Roses are $color' ; // Roses are $color

str_split (string, length) ← default

③ explode (separator , string , limit)
↓ optional (no. of elements to be returned)
⟹ 0, +ve, -ve (from last) SKIP
(where to break string)
(the string to split)
[breaks a string into an array]

eg → $str = "Hello world. It is a day" ;
explode (" ", $str)) ;
output →
Array ([0] ⇒ Hello [1] ⇒ world. [2] It [3] is [4] a [5] day)

④ htmlspecialchars ($str) ;

⑤ implode (separator , array)
optional
[returns a string from the elements of array]

$arr = array ('Hello', 'world!') ;
echo implode (" ", $arr) ;

⑥ ltrim ($str , "Hello");
↳ removes from left

⑦ str_repeat ("str" , repeat) ;
numerical value

⑧ str_replace (find , replace , string , count)
(optional)

a) { echo str_replace ("world" , "Peter" , "Hello world!") ;
output → Hello peter!)

b) { echo $a = array ("Blue" , "Red" , "Yellow") ;
print_r (str_replace ("Red" , "Pink" , $a , $i)) ;
output →
array ([0] => Blue [1] => Pink [2] => Yellow) .

⑨ str_word_count (string) // returns no. of words

⑩ strchr (string , search , before_search) ;
↓ to search ↓ to search for (optional)
→ returns rest of the string

⑪ strcmp (str1 , str2) → 0 - equal
<0 - str1 < str2
>0 - str1 > str2

⑫ strpos (str , find , start)
(returns start pos of str) optional

⑬ strlen (str)
↳ returns length of a str.

(14) strrev (string)

(15) trim (string, charlist)
optional
If omitted, removes whitespaces

(removes charlist from both sides)

⇒ Global Variables — Superglobals

{ Predefined variables in PHP that are always accessible regardless of scope.

(1) $GLOBALS → used to access global variables
→ php stores all global variables in an array called $GLOBALS[index].
↳ name of variable.

(2) $_SERVER → holds info about headers, paths & script locations

- $_SERVER['PHP_SELF']
returns filename of the currently executing script
- $_SERVER['SERVER_NAME']
returns name of the host server
- ['HTTP_HOST']
returns host header from current request.
- ['HTTP_REFERER'] ['HTTP_USER_AGENT'] ['SCRIPT_NAME']
(URL of current page)
- ['QUERY_STRING'] → query string if the page is accessed via a query string

(3) $_REQUEST
used to collect data after submitting an HTML form.

(4) $_POST
method = "post".

(5) $_GET
method = "get"
also collect data send in URL.

⇒ $_COOKIE , $_SESSION , $_FILES , $_ENV.

⇒ What is query string & Explain with an example

Ans →
The information can be sent across the web pages. This information is called query string.

- This query string can be passed from one page to another by appending it to the address of the page.
- You can pass more than one query string by inserting the '&' sign b/w query strings.

contains
query string ID & its value.

The query string passed across the web pages is stored in $_REQUEST variable, $_GET or $_POST.

```
<html>
<head>
<title> ——
</head
<body>
<form action = "welcome.php" method = "get">
<input type = "text" name = "usuname" >
<input type = "text" name = "email" >
<input type = "submit" name = "sub" value = "submit" >
</form>
```

welcome.php ↓

```
echo " Welcome $_GET['usuname']
$_GET['email']
```

Q→ Give the syntax to pass & capture the variables between PHP webpages during navigation

① url

```
<a href = "index.php?id = Tanya&usn = tom" > ___ </a>
```

② GET / POST method form.
access → $_REQUEST['id'];

③ sessions
start_session();
$_SESSION['id'] = "Tanya";

Q →

```
$ String1 = "into onto unto.";

$ Sub := "to";
  IF ( strpos (str1, Find) ;
  str_replace ( $ $sub, "Tanya" , $String1 )
```

→ strstr ()

```
Strstr ( $string , $search , $ before)
```

$string → in which we search

$search → to search for

Optional → false default; true returns part of $string before the first occurrence of $search

returns rest of the string from matching point

```
strstr ( $use , "@")

$url = "nachiketan@example.com";
```

→ output → @example.com

```
ltrim ("@example.com" , "@" );
```

→ output → example.com

Regular Expressions

↳ are powerful pattern matching algorithm that can be performed in a single expression.
↳ helps in validating email, IP address etc

functions

(1) preg_match ( )
used to perform match on a string. It returns true if a match is found & false otherwise

```
<?php
    $my_url = "www.guru99.com";
    IF (preg_match ("/guru/", $my_url))
    {
    {
                "/pattern/", String.
?>
```

(2) preg_split ( )
used to perform a pattern match on a string and then split the results into a numeric array.

```
<?php
$my_text = "I love Regular Expressions";
$my_array = preg_split ("/ /", $mytext );
?>
```

= Output → Array ([0] ⇒ I [1] ⇒ love [2] ⇒ Regular [3] ⇒ Expressions).

③ Preg_replace()

↳ used to perform a pattern match on a string & then replace the match with the specified text.

```
$text = "We at Guru99 to make quality";
$text = preg_replace("/Guru/",
        '<Span style="background: yellow">Guru</span>',
        $text);
```

⇒ \d → any digit (0-9)
\s → any white space
\w → any word char (a-z) (A-Z) (0-9) (_)
. → any char
[abc] ⇒ a, b or c
[^a-z] ⇒ not a-z

/D → not digit
/W ⇒ not word

(abc){3} → abc abc abc

{3} → match only three of those
? → zero or one char
* → zero or more chars
+ → one or more
^abc → beginning
abc$ → at the end
(abc) → group

⇒ Email 'first_name.last_name@domain.com'

```
$regexp = "/^[\w.]+@[\w.]+[a-z]{2,4}$/
```

c";

Switch for case insensitive match

Database

① configuration file

```
<?php
$servername = "localhost";
$password = "";
$username = "root";
$db = "myDB";
$conn = mysqli_connect ($servername, $username, $password, $mydb);
?>
```

② (included only once)
createdb

```
<?php
  $sv =
  pas =
  usn =

$conn = mysqli_connect ($sv, $pas, $usn);
IF ($conn)
  { echo "connection ";
    $sql = "create database myDB";
    $res = mysqli_query ($conn, $sql);
     IF ($res)
       { echo " Database";
    else }{
```

(3) create Table (only once)

```
<?php
    $ss - include ('configureatn.php');
IF ($conn)
  {
echo " success ";
$sql = "create table users (id int not null primary
    key , name varchar (250) not null ,
    email - -
     pass- - -          ) ;
$res = mysqli_query ($conn , $sql ) ;
IF ($res)
  { echo "success" ; }

?>
```

(4) login.php

```
<?php
    includ 'config.php' ;
echo " <form action = 'login.php' method='post' >
    <input —
    </form> " ;
IF (isset ($_POST ["submit"])) {
      $email = $_POST ['email'] ;
      $pass - -
$sql = " select name from users where email =
      '$email' AND password = '$pass' ; " ;
$res = mysqli_query ($conn , $sql ) ;
$rescheck = mysqli_num_rows ($res) ;
```

```
if ($rescheck > 0) {
  $row = mysqli_fetch_assoc ($res);
    $name = $row["name"];
    echo "Hello, $name";
    }
else
  { echo "Invalid email or password..."; }
?> }
```

(5) Insert

```
$sql = "Insert into employee ( )
        values ( ) ; ";
$sql. = '_____
$sql. = _____
if (mysqli_multi_query ($conn, $sql))
    {
    echo "Multiple rows inserted";
    }
else {
    ___
    ___
    }
```

⇒ Magic constants in PHP

- php provides a set of special defined constants that change depending on where they are used
  __LINE__ , __FILE__ , __DIR__ .

⇒ include() → gives warning if file is not present
require() → does same but upon failure generates fatal error & halts execution.
include_once() → check if file already has been included & so it will not include the file again

★ $car = array("one" ⇒ "BMW", "two" ⇒ "toyota");

⇒ var_dump ($car) → displays no. of elements, datatype & values

Output :

array (2) {
['One'] ⇒ BMW

⇒ $arr = array('', false, 42, array("42"));

① var_dump ($arr);

array(4) {
[0] ⇒ string(0) ""
[1] ⇒ boolean (false)
[2] = int (42)
[3] = array (1) { [0] ⇒ string(2) "42" }
}

② print_r($arr)

⇒

Array (
[0] ⇒
[1] ⇒ false
[2] ⇒ 42
[3] ⇒ Array ([0] ⇒ 42)
)

③ Var_export ($arr)

⇒

array (
0 ⇒ '',
2 ⇒ false,
3 ⇒ 42,
3 ⇒ array (0 ⇒ 42,),
)

$val = 42;
$valu = '41';
If (is_numeric ($val) // True
If (is_numeric ($valu) // True

Question paper

Q → 6

```
<? php

$server = "localhost";
$user = "root";
$pass = "";
$db = "admin";
$conn = mysqli_connect ($server, $user, $pass,
                        $db);
IF (! $conn)
  {
    die ("Connection failed : ". mysqli_connect_
                                 error ());
  }
else {
    $sql = "select id, name, salary from
            employee";
$result = mysqli_query ($conn, $sql);
  $res = mysqli_num_rows ($result);

IF ($res > 0)
  {
  echo "<table> <tr> <th> ID </th>
                <th> Name </th>
                <th> Salary </th>
            </tr> ";

while ( $row = mysqli_fetch_assoc ($result))
  {
    echo "<tr> <td>". $row ["id"]. " </td>
          <td>". $row ["name"]. " </td> <td>.
          $row ["salary"]. " </td> </td>";
  }
```

```
    echo "</table>";
}
else {
    echo "0 results";
    mysqli_close($conn);
}
?>
```

```
<?php
$x = 3;
function fn($var)
{
    global $a;
    static $a = 0;
    $b = 2;
    $b++;
    $a++;
    $x++;
    echo $b;
    echo $a;
    echo $x;
    echo "<br>";
}
$ar = array(1,2,3,4);
array_walk($ar, "fn");
?>
```

Output:

$x = 3, $a = 0, $b = 2
1 2 3
echo → 314

$ 325

Page No.
Date : / /

☆ GET method()

- Displayed in URL.
- Remain in Browser history and can never be cached.
- Can be bookmarked.
- Should not be used with sensitive data, used only to retrieve data.
- Have length restrictions of upto 1024 characters only.
- Can't be used to send binary data like images or word documents to the server.

☆ POST method()

- Not displayed in URL.
- Do not remain in browser history.
- Can't be bookmarked.
- Should be used with sensitive data.
- Do not have any length restrictions.
- Can be used to send text as well as binary data.

☆ SuperGlobals

It is a predefined variable that is always accessible, through any function, class or file.

$GLOBALS: Contains a reference to every variable within the global scope of the script. The keys are the names of the global variable.

$_SERVER: An array containing information such as headers, paths and script locations. The entries in this array are created by web server.

Page No.
Date : / /

$_REQUEST: An associative array consisting of contents of $_GET, $_POST and $_COOKIE.

$_GET: An associative array of variables passed to the current script via HTTP GET method.

$_POST: An associative array of variables passed to the current script via HTTP POST method.

$_FILES: An associative array of items uploaded to current script via HTTP POST method.

$_COOKIE: An associative array of variable passed to current script via HTTP cookie.

$_SESSION: An associative array containing session variable to the current script.

$_PHP_SELF: A string containing PHP script file name in which it is called.

$php_errormsg: Variable containing the text of last error message generated by PHP.

$_SERVER['PHP_SELF']: Returns the filename of currently executing script.

$_SERVER['SERVER_ADDR']: Returns the IP address of the host server.

Page No.
Date : / /

$_SERVER['SERVER_NAME'] : Returns name of host server.

$_SERVER['REQUEST_METHOD']: Returns the request method used to access the page.

$_SERVER['QUERY_STRING']: Returns the query string if the page is accessed via a query string.

$_SERVER['HTTP_HOST']: Returns the host header from the current request.

$_SERVER['HTTP_REFERER']: Returns the complete URL of the current page.

$_SERVER['REMOTE_ADDR']: Returns the IP address from where the user is viewing the current page.

$_SERVER['REMOTE_HOST']: Returns the host name from where the user is viewing the current page.

$_SERVER['REMOTE_PORT']: Returns the port being used on the user's machine to communicate with web server.

$_SERVER['SCRIPT_FILENAME']: Returns the absolute pathname of currently executing script.

$_SERVER['SCRIPT_NAME']: Returns the path of the current script.

$_SERVER['SCRIPT_URI']: Returns the URI of current page.

Page No.
Date : / /

* File Handling functions

fopen : opens file or URL.
fclose : closes an open file pointer.
fread : Binary-safe file read.
fwrite : Binary-safe file write.
feof : Tests for end of file on a file pointer
file_exists : checks whether a file or directory exists.
fgetc : gets character from file pointer.
fgets : gets line from file pointer.
fflush : flushes the output to a file.
fseek : seeks on a file pointer.
ftell : Returns current position of the file read/write pointer.
fscanf : Parses output from a file according to a format
readfile : outputs a file.
rename : Renames a file or directory
rewind : Rewind the position of a file pointer.
mkdir : Makes a directory
rmdir : Removes a directory

filesize : Returns length of file.
copy : copies a file
file : Reads a file into an array.
file_get_contents : Reads a file into a string.
file_put_contents : Writes a string to a file.
fileowner : Returns user ID (owner) of a file.
fileperms : Returns the permissions of a file.
filetype : Returns file type.
flock : Locks or releases a file.
ftruncate : Truncates a file to a specified length

unlink: Deletes a file.
is_readable: checks whether a file is readable.
is_writeable: checks whether a file is writeable.
pathinfo: Returns information about a file path.
stat: Returns information about a file.
umask: changes file permissions for file.

☆ Cookies

- A cookie is a small file that the server embeds on the user's computer.
- A cookie is used to identify a user.
- Each time the some computer requests a page with a browser it will send the cookie too.
- cookies are usually set in an HTTP header.
- Cookie header contains a name-value pair, path & domain.

* Creating a cookie

A cookie is created with the setcookie() function.

setcookie(name, value, expire, path, domain, secure);

setcookie() function must appear before the <html> tag.

Simplest way to access cookies is with $_COOKIE associative array.

isset() function is used to check if a cookie is set or not.

```
<form action="setcookie.php" method="POST">
    Name: <input type="text" name="name" />
    Age: <input type="text" name="age" />
    <input type="submit" name="submit" />
</form>
```

```
<?php
    if (isset($_POST['submit']))
    {
        $nm = $_POST['name'];
        $age = $_POST['age'];
        setcookie("name", $nm);
        setcookie("age", $age, time()+3600, "/");
?>
```

```
<?php
    if((isset($_COOKIE['name'])) && (isset($_COOKIE['age'])))
    {
        echo "Name :". $_COOKIE['name']. "<br> Age :". $_COOKIE['age'];
    }
?>
```

* Deleting a cookie

To delete a cookie, use the setcookie() function with an expiration date in past.

```
<?php
    setcookie("age", $age, time()-3600, "/");
?>
```

Database

Page No.
Date : / /

2. php

```
$host = "localhost";
$username = "username";
$password = "password";

// create connection
$conn = mysqli_connect ($host, $username, $password);

// check connections
if (! $conn)
    die ("connection failed : ". mysqli_connect_error ());

// create database
$ sql = " CREATE DATABASE myDB";
if (mysqli_query ($conn, $ sql))
    echo " Database created successfully ";
else
    echo "Error creating database :". mysqli_error ($conn);

// close a connection
mysqli_close ($conn);
?>
```

To get ID of last inserted record

```
$last_id = mysqli_insert_id ($ conn);
```

mysqli_query(): used to execute single SQL statement.

mysqli_multi_query(): used to execute multiple SQL statements.

Page No.
Date : / /

* Select Data

```
$sql = "SELECT id, firstname, lastname FROM MyDB";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0)
    while ($row = mysqli_fetch_assoc($result))
        echo "id: ". $row["id"]. " Name: ". $row["firstname"];
else echo "0 results";
```

num_rows() : checks if there are more than 0 rows returned.

fetch_assoc() : puts all the returned rows into an associative array.