Web Technologies

Module I

PHP

Introduction to PHP: Origins and Uses of PHP, Overview of PHP - General Syntactic Characteristics - Primitives, Operations, and Expressions – Control Statements, Arrays, Functions, Pattern Matching, Form Handling, Cookies, Session Tracking

1. Origin and Use of PHP

PHP was developed by Rasmus Lerdorf, in 1994 PHP stands for "Hypertext Preprocessor ".Originally, PHP was an acronym for Personal Home Page. It is open source general-purpose scripting language As a server-side scripting language, PHP is naturally used for form handling and database access.

2. Overview of PHP

PHP is a web development tool and can be embedded into HTML .When a browser requests a document that includes PHP script, the Web server that provides the document calls its PHP processor. The server determines that a document includes PHP script by the file-name extension. If it is .php, .php3, or .phtml, the document has embedded PHP.

The PHP processor has two modes of operation: copy mode and interpret mode. It takes a PHP document file as input and produces an HTML or XHTML document file. When the PHP processor finds markup code (which may include embedded client-side script) in the input file, it simply copies it to the output file. When the processor encounters PHP script in the input file, it interprets it and sends any output of the script to the output file. This implies that the output from a PHP script must be HTML or XHTML, either of which could include embedded client-side script. The new file (the output file) is sent to the requesting browser. The client never sees the PHP script. If the user clicks View Source while the browser is displaying the document, only the markup (and embedded client-side script) will be shown, because that is all that ever arrives at the client. Separated in files with the <?php ?> tag . php commands can make up an entire file, or can be contained in html. Program lines end in ";" or you get an error.PHP scripts are executed on the server.PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.) .

3.General Syntactic Characteristics

PHP scripts either are embedded in HTML or XHTML documents or are in files that are referenced by such documents. PHP code is embedded in documents by enclosing it between the <?php and ?> tags. If a PHP script is stored in a different file, it can be brought into a document with the include construct, which takes the filename as its parameter— for example, `include("hello.php");` The included file can contain markup or client-side script, as well as PHP code, but any PHP script it includes must be the content of a <?php tag.

The reserved words of PHP

and	else	global	require	virtual
break	elseif	if	return	xor
case	extends	include	static	while
class	false	list	switch	
continue	for	new	this	
default	foreach	not	true	
do	function	or	var	

PHP allows comments to be specified in three different ways. Single-line comments can be specified either with # or with //, as in JavaScript. Multiple-line comments are delimited with /* and */, as in many other programming languages. PHP statements are terminated with semicolons. Braces are used to form compound statements for control structures.

4.Primitives, Operations, and Expressions

PHP has four scalar types—Boolean, integer, double, and string; two compound types—array and object; and two special types—resource and NULL.

Variable

All variable names in PHP begin with a dollar sign ($). The part of the name after the dollar sign is like the names of variables in many common programming languages: a letter or an underscore followed by any number (including zero) of letters, digits, or underscores. PHP variable names are case sensitive.PHP is compatible with almost all servers used today (Apache, IIS, etc.) . PHP is easy to learn and runs efficiently on the server side

```
<P>
<?php $myvar = "Hello World!";
echo $myvar;
?>
</P>
```

Assigned by value - *$foo = "Bob"; $bar = $foo;*

Assigned by reference, this links vars

$bar = &$foo;

Integer Type

PHP has a single integer type, named integer. (corresponds to the long type of C) . In most cases, this is 32 bits, or a bit less (not fewer) than ten decimal digits.

Double Type

PHP's double type corresponds to the double type of C and its successors. Double literals can include a decimal point, an exponent, or both. The exponent has an E or an e.T here need not be any digits before or after the decimal point, so both `.345` and `345.` are legal double literals.

String Type

Characters in PHP are single bytes.There is no character type. A single character data value is represented as a string of length 1. String literals are defined with either single (') or double quotes (") delimiters. In single-quoted string literals, escape sequences, such as \n, are not recognized and the values of embedded variables are not substituted. This substitution is called interpolation.

Print 'The sum is :$sum' is exactly as it is typed.ie The. sum is:$sum

In double-quoted string literals, escape sequences are recognized, and embedded variables are replaced by their current values.

```
$sum=10;

Print "The sum : $sum";

o/p :The sum : 10
```

Boolean Type

The only two possible values for the Boolean type are TRUE and FALSE, both of which are case insensitive. If an integer expression is used in Boolean context, it evaluates to FALSE if it is zero; otherwise, it is TRUE. If a string expression is used in Boolean context, it evaluates to FALSE if it is either the empty string or the string "0"; otherwise, it is TRUE.

PHP Objects

An object is a data type that not only allows storing data but also information on, how to process that data. An object is a specific instance of a class which serve as templates for objects. Objects are created based on this template via the new keyword.

Every object has properties and methods corresponding to those of its parent class.

NULL

The special NULL value is used to represent empty variables in PHP. A variable of type NULL is a variable without any data. NULL is the only possible value of type null.

RESOURCE

A resource is a special variable, holding a reference to an external resource. Resource variables typically hold special handlers to opened files and database connections.

Arithmetic Operators and Expressions

PHP has the usual (for C-based programming languages) collection of arithmetic operators (+, -, *, /, %, ++, and --).If either operand is a double, the operation is double and a double result is produced .

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Function	Parameter Type	Returns
`floor`	Double	Largest integer less than or equal to the parameter
`ceil`	Double	Smallest integer greater than or equal to the parameter
`round`	Double	Nearest integer
`srand`	Integer	Initializes a random number generator with the parameter
`rand`	Two numbers	A pseudorandom number greater than the first parameter and smaller than the second
`abs`	Number	Absolute value of the parameter
`min`	One or more numbers	Smallest
`max`	One or more numbers	Largest

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

String Operations

The only string operator is the concatenation operator (.) ,used to put two string values together. To concatenate two string variables together, use the concatenation operator.

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

```
Hello World! What a nice day!
```

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 is given (an integer), it specifies the length of the returned substring
chop	A string	The parameter with all whitespace characters removed from its end
trim	A string	The parameter with all whitespace characters removed from both ends
ltrim	A string	The parameter with all whitespace 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

Note for strpos: Because false is interpreted as zero in numeric context, this can be a problem. To avoid it, compare the returned value to zero using the === operator (see Section 11.6.1) to determine whether the match was at the beginning of the first string parameter (or there was no match).

Scalar Type Conversions

PHP includes both **implicit** and **explicit** type conversions. Implicit type conversions are called **coercion**s. The context can cause a coercion of the type of the value of the expression. Whenever a numeric value appears in string context, the numeric value is coerced to a string. whenever a string value appears in numeric context, the string value is coerced to a numeric value. If the string contains a period, an e, or an E, it is converted to double; otherwise, it is converted to an integer. When a double is converted to an integer, the fractional part is dropped; rounding is not done.

Explicit type conversions can be specified in three ways.

Using the syntax of C, an expression can be cast to a different type. The cast is a type name in parentheses preceding the expression. For example, if the value of $sum is 4.777, the following produces 4: **(int)$sum.**

Another way to specify explicit type conversion is to use one of the functions intval, doubleval, or strval. For example, if $sum is still 4.77 7, the following call returns 4:

```
intval($sum)
```

The third way to specify an explicit type conversion is the settype function, which takes two parameters: a variable and a string that specifies a type name. For example, if $sum is still 4 . 777, the following statement converts the value of $sum to 4 and its type to integer:

```
settype($sum, "integer");
```

5. Output

The print function is used to create simple unformatted output. It can be called with or without parentheses around its parameter.

Legal statements example

```
print "Apples are red <br /> Kumquats aren't <br />";

print( 47 ); produce 47

print "The result is: $result <br />";
```

The general form of a call to printf is as follows:

```
printf (literal_string, param 1, param 2, ...)
```

The literal string can include labeling information about the parameters whose values are to be displayed.

It also contains format codes for those values. The form of the format codes is a percent sign (%) followed by a field width and a type specifier. The most common type specifiers are **s** for strings, **d** for integers, and **f** for floats and doubles.

Consider the following examples:

% 10 s—a character string field of 10 characters

%6d—an integer field of 6 digits

%5. 2f—a float or double field of 8 spaces, with two digits to the right of the decimal point, the decimal point, and 5 digits to the left.

```
<?xml version = "1.0" encoding = "utf-8"?>

<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1 //EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns = "http://www.w3.org/1999/xhtml"> <head>

<title> today.php </title> </head>

<body>

<p>

<?php
```

```
print "<b>Welcome to my home page <br /> <br />";
print "Today is:</b>date("Y/m/d"); print "<br />";
?>
</p>
</body>
</html>
```

OUTPUT

Welcome to my Home Page

Today is 2014/09/24

NB: Try, date(l,F,jS)=>Saturday June 1st

6.Control Statements

Relational Operators

PHP uses the eight relational operators of JavaScript. The usual six (>, <, >=, <=, ! =, and ==) have the usual meanings. It also has ===, which produces TRUE only if both operands are the same type and have the same value, and ! ==, the opposite of ===.

Boolean Operators

There are six Boolean operators: and, or, xor, !, &&, and | |. The and and && operators perform the same operation, as do or and | |.

Selection Statements

if

PHP's if statement is like that of C.The control expression can be an expression of any type,but its value coerced to Boolean.The controlled statement segment can be either a single statement or a compound statement.An is can include any number of elseif clauses

Example of an if construct:

```
if ($day == "Saturday" || $day == "Sunday")
{$today = "weekend"};
else
{$today = "weekday";}
```

switch

```
switch (n) {
 case label1:
  code to be executed if n=label1;
  break;
```

```
  case label2:
    code to be executed if n=label2;
   break;
  case label3:
   code to be executed if n=label3;
   break;
 ...
 default:
   code to be executed if n is different from all labels;
}
```

Loop Statements

Loop statements are while, for, do-while and foreach statement

while

Compute the factorial of $n

```
  $fact=1;

  $count=1

  While($count<$n) {

  $count++;

 $fact *=$count;

 }
```

do-while

computes the sum of the positive integers up to 100using do while

```
 $count = 1;

 $sum = 0;

 do

  {

 $sum += $count;

 $count++;

 } while ($count <= 100);
```

for

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init; condition; increment)
  {
  code to be executed;
  }
```

```
for($count=1,$fact=1;$count<$n;$count++)

 $fact *=$count;
```

Break and continue

The break statement can be used to terminate the execution of a for, foreach, while, or do-while construct. The continue statement is used in loop constructs to skip the remainder of the current iteration but continue execution at the beginning of the next.

7 .PHP Arrays

- In PHP, there are three kind of arrays:
- **Numeric array** - An array with a numeric index
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

Array creation

Accessing Array elements

Functions dealing with Arrays

Sequential Access to Array

Sorting Arrays

7.1 Numeric Arrays creation

- A numeric array stores each array element with a numeric index.
- There are two methods to create a numeric array.
- The assignment operation creates scalar variables. The same operation works for arrays —assigning a value to an element of an array that does not yet exist creates the array.
- For example, assuming no array named $list currently exists, the following statement creates one:$list[0] = 17;
- If the script has a scalar variable named $list prior to this assignment, $list is now an array.
- If the array currently has no elements with numeric keys, the value 0 is used. For example, in the following code, the second element's subscript will be 2:

- $list[l] = "Today is my birthday!"; $list[] = 42;
- The second way to create an array is with the **array** construct.
- In the following example the index is automatically assigned (the index starts at 0):

```
$list==array(17,24,45,91);
```

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

The elements are stored like this

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

Array construct can be used to create empty array.in the following statement $list becomes an array variable with no element.

Accessing Array Elements

Individual array elements can be accessed by subscrpting. The value in the subscript (enclosed in brackets) is the key of the value being referenced. For example ,the value of the element whose key is “mary” in the $ages array can be set to 29 with the following statement:

```
$ages['Mary']=29;
```

In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

- **The code above will output:**

```
Saab and Volvo are Swedish cars.
```

PHP Associative Arrays

- With an associative array, each ID key is associated with a value.
- When storing data about specific named values, a numerical array is not always the best way to do it.

> With associative arrays we can use the values as keys and assign values to them

- In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

- This example is the same as the one above, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

The code above will output:

```
Peter is 32 years old.
```

PHP Multidimensional Arrays

- In a multidimensional array, each element in the main array can also be an array.And each element in the sub-array can be an array, and so on.

The array above would look like this if written to the output:

```
Array
(
[Griffin] => Array
  (
  [0] => Peter
  [1] => Lois
  [2] => Megan
  )
[Quagmire] => Array
  (
  [0] => Glenn
  )
[Brown] => Array
  (
  [0] => Cleveland
  [1] => Loretta
  [2] => Junior
  )
)
```

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
  (
  "Griffin"=>array
  (
  "Peter",
  "Lois",
  "Megan"
  ),
  "Quagmire"=>array
  (
  "Glenn"
  ),
  "Brown"=>array
  (
  "Cleveland",
  "Loretta",
  "Junior"
  )
  );
```

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

The code above will output:

```
Is Megan a part of the Griffin family?
```

Functions dealing with Arrays

A whole array can be deleted with **unset**, as with a scalar variable. Individual elements of an array also can be removed with unset, as in the following:

```
$list = array(2, 4, 6, 8);

unset($list[2]);
```

Now $list has three remaining elements with keys 0,1, and 3 and elements 2, 4, and 8.

The collection of keys and the collection of values of an array can be extracted with built-in functions. The **array_keys** function takes an array as its parameter and returns an array of the keys of the given array. The returned array uses 0, 1, and so forth as its keys. The **array_values** function does for values what array_keys does for keys.

For example:

```
   $highs = array("Mon" => 74, "Tue" => 70, "Wed" => 67,"Thu" => 62,
"Fri" => 65);

  $days = array_keys($highs);

   $temps = array_values($highs);
```

Now the value of $days is ("Mon", "Tue", "Wed", "Thu", "Fri"), and

The value of $temps is (74 , 70, 67 , 62 , 65).

The existence of an element of a specific key can be determined with the **array_key_exists** function, which returns a Boolean value. For example, consider the following:

```
  $highs = array("Mon" => 74, "Tue" => 70, "Wed" => 67,"Thu" => 62,
"Fri" => 65);

  if (array_key_exists("Tue", $highs))

  { $tues_high = $highs["Tue"];

  print "The high on Tuesday was $tues_high <br />";

  }
```

The **is_array** function is similar to the is_int function: It takes a variable as its parameter and returns TRUE if the variable is an array, FALSE otherwise. The **in_ array** function takes two parameters an expression and an array and returns TRUE if the value of the expression is a value in the array; otherwise, it returns FALSE.

The number of elements in an array can be determined with the **sizeof** function.

For example, consider the following code:

```
  $list = array("Bob", "Fred", "Alan", "Bozo");
  $len = sizeof($list);
```

After executing this code, $len will be 4.

Note that PHP does not interpolate array elements embedded in double quoted strings.

It is able to convert between strings and arrays. These conversions can be done with the implode and explode functions.

explode & implode

The **explode** function explodes a string into substrings and returns them in an array.The delimiters of the substrings are defined by the first parameter to explode, which is a string; the second parameter is the string to be converted. For example, consider the following:

```
$str = "April in Paris, Texas is nice";

$words = explode(" ", $str);
```

$words contains ("April", "in", "Paris,", "Texas", "is", "nice").

The **implode** function does the inverse of explode. Given a separator character (or string) and an array, it catenates the elements of the array together, using the given separator string between the elements, and returns the result as a string.

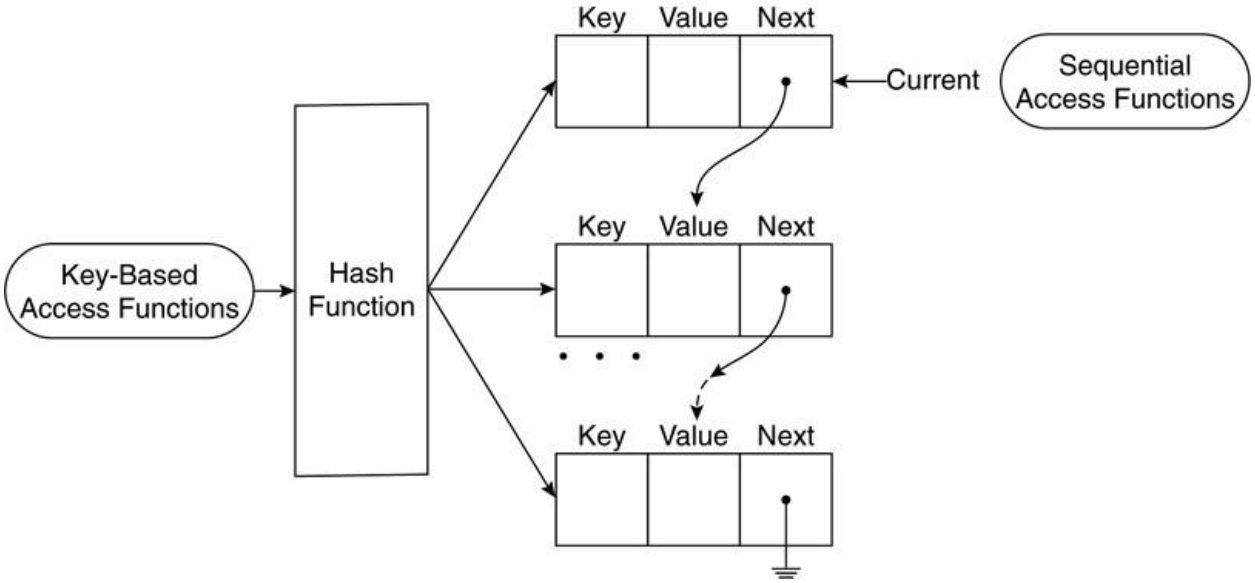
```
$words = array("Are", "you", "coming", "tonight");

 $str = implode(" ", $words);
```

$str has "Are you coming tonight".

Internal structure of Arrays

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

Logical internal structure of arrays



Sequential Access to Array

PHP includes different ways to access array elements in sequential order. Every array has an internal pointer, or marker, that references one element of the array called the "current" pointer. current pointer is initialized to reference the first element of the array at the time the array is created.

```
$cities = array("Hoboken", "Chicago", "Moab", "Atlantis");

$city = current($cities);

print("The first city is $city <br />");
```

This code produces the following:

The first city is Hoboken

current() & next()

The "current" pointer can be moved with the `next` function, which both moves the pointer to the next array element and returns the value of that element. If the "current" pointer is already pointing at the last element of the array, next returns FALSE.

For example, if the "current" pointer is referencing the first element of the $cities array, the following code produces a list of all of the elements of that array:

```
$city = current($cities);

print("$city <br />");

while ($city = next($cities))

print("$city <br />");
```

NB: creates problem when array includes an element with false value. `each` function avoids this problem.

each function

The `each` function, which returns a two-element array consisting of the key and the value of the "current" element .It returns FALSE only if the "current" pointer has gone past the last element of the array. difference between each and next .

- `each` returns the element being referenced by the "current" pointer and then moves that pointer.

- `next` function first moves the "current" pointer and then returns the value being referenced by the "current" pointer.

consider the following code:

```
$salaries = array("Mike" => 4250,"Jery" => 5250,"Fred" => 3792);

 while ($employee = each($salaries))

{$name = $employee["key"];

$salary = $employee["value"];

print("The salary of $name is $salary <br />");
```

```
}
```

prev(), reset(), end(), key()

- The "current" pointer can be moved backward (that is, to the element before the "current" element) with the **`prev`** function.
- Like the next function, the prev function returns the value of the element referenced by the "current" pointer after the pointer has been moved.
- The "current" pointer can be set to the first element with the **`reset`** function, which also returns the value of the first element. It can be set to the last element of the array with the **`end`** function, which also returns the value of the last element.
- The `key` function, when given the name of an array, returns the key of the "current" element of the array.

array_push() and array_pop()

The **`array_push`** and **`array_pop`** functions provide a simple way to implement a stack in an array.

The `array_push` function takes as its first parameter an array. After this first parameter, there can be any number of additional parameters. The values of all subsequent parameters are placed at the end of the array. The array_push function returns the new number of elements in the array. The `array_pop` function takes a single parameter, the name of an array. It removes the last element from the array and returns it. The value NULL is returned if the array is empty.

foreach statement

- The foreach statement is designed to build loops that process all of the elements of an array.
- This statement has two forms:

```
foreach (array as scalar_variable) loop body

foreach (array as key => value) loop body
```

- In the first form, one of the array's values is set to the scalar variable for each iteration of the loop body.

For example:

```
foreach (Slist as $temp)

print("$temp <br />");
```

- This code will produce the values of all of the elements of $list .

The second form of foreach provides both the key and the value of each element of the array.

Syntax:

foreach (array as key => value) loop body

For example:

```
$lows = array("Mon" => 23, "Tue" => 18, "Wed" => 27);

 foreach ($lows as $day => $temp)

print("The low temperature on $day was $temp <br />");
```

Sorting Arrays

Sort()

The sort function, which takes an array as a parameter, sorts the values in the array, replacing the keys with the numeric keys, 0, 1, 2, ...The array can have both string and numeric values. The string values migrate to the beginning of the array in alphabetical order. The numeric values follow in ascending order.

asort()

The asort function is used to sort arrays that correspond to Perl hashes. It sorts the elements of a given array by their values but keeps the original key/ value associations.As with sort, string values all appear before the numeric values, in alphabetical order. The numeric values follow in ascending order.

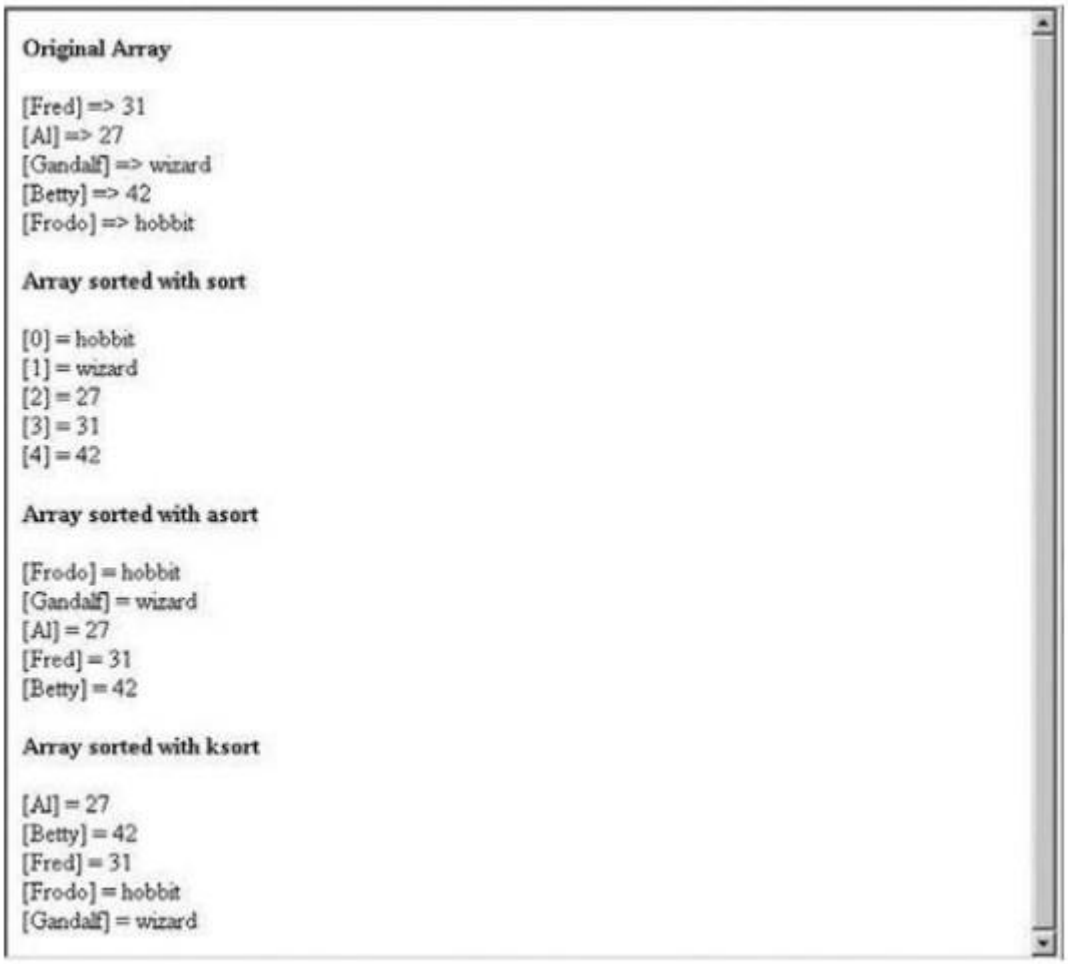
ksort()

The ksort function sorts its given array by keys, rather than values. The key/value associations are maintained by the process.

The rsort,arsort and krsort functions behave like the sort,assort and ksort functions respectivelyexcept that they sort into the reverse order of their counterparts.

```
<!DOCTYPE html>

<!—sorting.php>

<html lang=”en”>

<head>

<title>sorting</title>

<meta charset=”utf-8” />

</head>

<body>
  <?php
    $original = array("Fred" => 31, "Al" => 27,
                      "Gandalf" => "wizard",
                      "Betty" => 42, "Frodo" => "hobbit");
  ?>
  <h4> Original Array </h4>
  <?php
    foreach ($original as $key => $value)
      print("[$key] => $value <br />");
```

```
        $new = $original;
        sort($new);
      ?>
      <h4> Array sorted with sort </h4>
      <?php
        foreach ($new as $key => $value)
          print("[$key] = $value <br />");

        $new = $original;
        asort($new);
      ?>
      <h4> Array sorted with asort </h4>
      <?php
        foreach ($new as $key => $value)
          print("[$key] = $value <br />");

        $new = $original;
        ksort($new);
      ?>
      <h4> Array sorted with ksort </h4>
      <?php
        foreach ($new as $key => $value)
          print("[$key] = $value <br />");
      ?>
    </body>
</html>
```



8. Functions

General characteristics of functions

PHP supports user-defined functions.The general form of a PHP function definition is as follows:

```
function name( [parameters] ) { }
```

Function names are not case sensitive. So, you cannot have a function named sum and another named Sum. The return statement is used in a function to specify the value to be returned to the caller.If no return statement no value returned.

Parameters

The parameters in the call to a function **actual parameters**. The parameters that are listed in the function definition **formal parameters**. The number of actual parameters in a call to a function does not need to match the number of formal parameters defined in that function. If there are too few actual parameters in a call, the corresponding formal parameters will be unbound variables. If there are too many actual parameters, the excess parameters will be ignored.

pass-by-value

Default parameter-passing mechanism of PHP is pass by value. The values of actual parameters are copied into the memory locations associated with the corresponding formal parameters in the called function. The values of the formal parameters are never copied back to the caller, so passing by value a one-way communication to the function.

```
   function max_abs($first, $second)
   {$first = abs($first);
  $second = abs($second);
  if ($first >= $second)
 return $first;
  else
  return $second;}
```

Pass-by-reference

- Pass-by-reference parameters can be done in PHP in two ways.
- One way is to add an ampersand (&) to the beginning of the name of the formal parameter that you want to be passed by reference.
- The other way is to add an ampersand to the actual parameter in the function call.

```
function set_max(&$max, $first, $second)
{ if ($first >= $second)
$max = $first;
else
$max = $ second;
}
```

In this example, the first actual parameter in the caller is set to the larger of the second and third parameters.

The Scope & Lifetime of Variables

The default scope of a variable defined in a function is local. If a variable defined in a function has the same name as a variable used outside the function, there is no interference between the two. A local variable is visible only in the function in which it is used.

```
function summer($list) {
  $sum = 0;
  foreach ($list as $value)
    $sum += $value;
  return $sum;
}
$sum = 10;

$nums = array(2, 4, 6, 8);
$ans = summer($nums);
print "The sum of the values in \$nums is: $ans <br />";
print "The value of \$sum is still: $sum <br />";
```

produces the following output:

```
The sum of the values in $nums is: 20
The value of $sum is still: 10
```

In some cases it is convenient for the code in a function to be able to access a variable that is defined outside the function. For this,PHP has the global declaration. When a variable is listed in a global declaration in a function, that variable is expected to be defined outside the function. So, such a variable has the same meaning inside the function as outside.

```
$big_sum = 0;
...
/* Function summer
   Parameter: An array of integers
   Returns: The sum of the elements of the parameter
            array
   Side effect: Add the computed sum to the global,
                $big_sum
*/
function summer ($list) {
  global $big_sum;   //** Get access to $big_sum
  $sum = 0;
  foreach ($list as $value)
    $sum += $value;
  $big_sum += $sum;
  return $sum;
} //** end of summer
...
$ans1 = summer($list1);
$ans2 = summer($list2);
...
print "The sum of all array elements is: $big_sum <br />";
```

8.4 The lifetime of variables

The default lifetime of local variables in a PHP function is from the time the variable is first used (that is, when storage for it is allocated) until the function's execution terminates. The lifetime of a **static** variable in a function begins when the variable is first used in the first execution of the function. Its lifetime ends when the script execution ends. **static** declaration can include an initial value,which is only assigned the first time the declaration is reached.

```
function do_it ($param) {
  static $count = 0;
  count++;
  print "do_it has now been called $count times <br />";
  ...
}
```

displays the number of times it has been called, even if it is called from several different places. The fact that its local variable $count is static allows this to be done.

9. Pattern Matching

PHP includes two different kinds of string pattern matching using regular expressions: one that is based on POSIX regular expressions and one that is based on Perl regular expressions.

The POSIX regular expressions are compiled into PHP, but the Perl-Compatible Regular Expression (PCRE) library must be compiled before Perl regular expressions can be used.

The preg_match function takes two parameters, the first of which is the Perl-style regular expression as a string. The second parameter is the string to be searched.

```
if (preg_match("/"PHP/", $str))

print "\$str begins with PHP <br />";

 else

print "\$str does not begin with PHP <br />";
```

The preg_split function operates on strings but returns an array and uses patterns. preg_split takes two parameters, the first of which is a Perl-style pattern as a string. The second parameter is the string to be split.

For example, consider the following sample code:

```
$fruit_string = "apple : orange : banana";

 $fruits = preg_split("/ : /", $fruit_string);
```

The array $ fruits now has ("apple", "orange", "banana").

```
<!DOCTYPE html>

<!-- word_table.php
       Uses a function to split a given string of text into
       its constituent words. It also determines the frequency of
       occurrence of each word. The words are separated by
       white space or punctuation, possibly followed by white space.
       The punctuation can be a period, a comma, a semicolon, a
       colon, an exclamation point, or a question mark.
       -->
```

```
<html lang="en">

<head>

<title> word_table.php </title>

<meta charset="utf-8" />

        <body>
        <?php

        // Function splitter
        //   Parameter: a string of text containing words and punctuation
        //   Returns: an array in which the unique words of the string are
        //            the keys and their frequencies are the values.
        function splitter($str) {

        // Create the empty word frequency array
          $freq = array();

        // Split the parameter string into words
          $words = preg_split("/[ \.,;:!\?]\s*/", $str);

        // Loop to count the words (either increment or initialize to 1)
          foreach ($words as $word) {
            $keys = array_keys($freq);
            if(in_array($word, $keys))
              $freq[$word]++;
            else
              $freq[$word] = 1;
          }
          return $freq;
        } #** End of splitter

        // Main test driver
          $str = "apples are good for you, or don't you like apples?
                  or maybe you like oranges better than apples";

        // Call splitter
          $tbl = splitter($str);

        // Display the words and their frequencies
          print "<br /> Word Frequency <br /><br />";
          $sorted_keys = array_keys($tbl);
          sort($sorted_keys);
          foreach ($sorted_keys as $word)
            print "$word $tbl[$word] <br />";
        ?>
        </body>
        </html>
</head>
```

Word Frequency

```
apples 3
are 1
better 1
don't 1
for 1
good 1
like 2
maybe 1
or 2
oranges 1
than 1
you 3
```

10. Form handling

- when PHP is used for form handling, the PHP script is embedded in an XHTML document, like other uses of PHP.
- The recommended approach is to use the implicit arrays for form values, $_POST and $_GET.
- These arrays have keys that match the form element names and values that were input by the client.
- For example, if a form has a text box named phone and the form method is POST, the value of that element is available in the PHP script as follows:

$_POST["phone"]

- The built-in $_GET function is used to collect values from a form sent with method="get".

> Information sent from a form with the GET method is visible to everyone and has limits on the amount of information to send (max. 100 characters).

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

—

```
http://www.w3schools.com/welcome.php?fname=Peter&age=37
```

Notice how the URL carries the information after the file name.

```
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
```

PHP Forms - $_POST Function

```
<form action="action.php" method="post">
 <p>Your name: <input type="text" name="name" /></p>
 <p>Your age: <input type="text" name="age" /></p>
 <p><input type="submit" /></p>
</form>
```

And here is what the code of action.php might look like:

```
Hi <?php echo htmlspecialchars($_POST['name']); ?>.
You are <?php echo (int)$_POST['age']; ?> years old.
```

- Apart from htmlspecialchars() and (int), it should be obvious what this does. htmlspecialchars() makes sure any characters that are special in html are properly encoded so people can't inject HTML tags or Javascript into your page.
- For the age field, since we know it is a number, we can just convert it to an integer which will automatically get rid of any stray characters. The $_POST['name'] and $_POST['age'] variables are automatically set for you by PHP.
- When to use method="post"?
- > Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
- > However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Simple php program

Wt12.php

```
<!DOCTYPE html>
<html lang=”en”>
<head>
<title> A Simple math calculator </title>
<meta charset=”utf-8” />
</head>
    <body>Insert two numbers in the form and hit submit button <br>
   <form action="calculation.php" method="post">
   Firstnumber: <input name="num1" type="text" /><br>
   Secondnumber: <input name="num2" type="text" />
   <input type="submit" />
   </form>
   </body>
   </html>
```

……..//calculation.php

```
   <html><head><title>Simple </title></head><body>
   <?php
```

```
$num1 = $_POST['num1'];
$num2 = $_POST['num2'];
$a = $num1 + $num2;
$b = $num1 - $num2;
$e=sqrt($num1);
$f=pow($num1,$num2);
echo "The sum of the two numbers is ". $a;
echo "The difference of the two numbers is ". $b;
echo "The sqrt of the  numbers is ". $e;
echo "The power of the  numbers is ". $f;
?></body></html>
```

Example of PHP program which handles forms with output:popcorn.html

```
<!DOCTYPE html>
<html lang=”en”>
<head>
<title> Popcorn Sales form </title>
<meta charset=”utf-8” />
<style type=”text/css”>
 td,th,table {border:thin solid black;}
</style>
</head>
<body>
<form action = ”http://localhost/popcorn.php”  method = “POST”>
<label>Buyers Name: <input type=”text” name= ”name” ></label>
<label>Address: <input type = “text” name = “address”></label>
<table>
<tr>
<th>Product Name</th>
<th>Price</th>
<th>quantity</th>
</tr>
```

```
<tr>
<td>Unpopped Popcorn</td>
<td>$3.00</td>
<td><input type=”text” name=”unpop” size=”2”></td>
</tr>
<tr>
<td>Caramel Popcorn</td>
<td>$3.50</td>
<td><input type=”text” name= “caramel” size=”2”></td>
</tr></table><p>input type = “submit” value=”submit Form” /> </p>
</form></body></html>
```

OUTPUT:

Buyer's Name: Anu

Street Address: Ernakulam

Product Name	Price	Quantity
Unpopped Popcorn	$3.00	2
Caramel Popcorn	$3.50	2

Submit Order

The PHP script that handles the data from the form is described in popcorn3.html

```
<!DOCTYPE html>
<html lang=”en”>
<head>
<title> Process the popcorn.html form </title>
<meta charset=”utf-8” />
<style type =”text/css”>
td,th,table {border:thin solid black;}</style>
</head>
<body>
<?php
```

```
//get form data values
  $total_price=0;
  $total_item=0;
  $name=$_POST[“name”];
 $street = $_POST[“street”];
  $unpop=$_POST[“unpop”];
  $caramel=$_POST[“caramel”];
  $total_price= 3.0*$unpop+3.5*$caramel;
  $total_items=$unpop+$caramel;
  //Return the results to the browser in a table
  ?>
  <h3> Customer:</h3>
  <?php
  Print “$name<br/>\n”, $street<br/>\n”;
  Print “Thank You for your order<br/>\n”;
  Print <b> “Your total bill is:</b> $  $total_price<br/>\n”;
  ?>
  </body></html>
```

OUTPUT

Customer

Anu

Ernakulam

Thank you for your order

Your total bill is: $13

EXAMPLE:

REGISTRATION PAGE

```
<form method="post" action="">
// if action of set as null then the page will redirect to itself
 <table>
    <tr><td>Name: </td><td>
```

```
  <input type="text" name="name"></td></tr>
  <tr><td>Address:</td><td>

  <textarea name="address"></textarea></td></tr>
    <tr><td>
  Gender:</td><td>
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  </td></tr>
    <tr><td></td><td><input type="submit" name="submit"
value="Submit"></td></tr>
  </table>
 </form>
```

// Storing registration details into database

```
  <?php
  if(isset($_POST['submit']))

  { // Check if we click on SUBMIT BUTTON
  $servername = "localhost";
 $username = "root";
 $password = "";
  $dbname = "myDB";
  // Create connection
  $conn = new mysqli($servername, $username, $password, $dbname);
  // Check connection
  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
  }// insert Query

  $sql = "INSERT INTO register(name,address,gender) VALUES
('$_POST[name]', '$_POST[address]', '$_POST[gender]')";
  if ($conn->query($sql) === TRUE) {
    echo "Registration Completed successfully!";
  } else {
    echo "Error: " . $sql . "<br />" . $conn->error;
  }$conn->close();}?>
```

11. FILES

All file operations in PHP are implemented as functions.

Opening and Closing Files

Reading from a file

Writing to a File

Locking Files 11.1Opening

and Closing Files

The first step in some file operations is to open it, a process that prepares the file for use and associates a program variable with the file for future reference. This program variable is called the file variable.

- For example,

```
$file_var = fopen("testdata.txt", "r") or

die ("Error – testdata.txt cannot be opened");
```

fopen & fclose

The fopen function takes two parameters: the filename, including the path to it if it is in a different directory, and a use indicator, which specifies the operation or operations that must be performed on the file. Both parameters are given as strings.

The fopen function returns the reference to the file for the file variable. Every open file has an internal pointer(file pointer) that is used to indicate where the next file operation should take place within the file. The file_exists function takes a single parameter, the file's name. It returns TRUE if the file exists, FALSE otherwise. A file is closed with the fclose function, which takes a file variable as its only parameter.

Table File use indicators

Use Indicator	Description
"r"	Read only. The file pointer is initialized to the beginning of the file.
"r+"	Read and write an existing file. The file pointer is initialized to the beginning of the file; if a read operation precedes a write operation, the new data is written just after where the read operation left the file pointer.
"w"	Write only. Initializes the file pointer to the beginning of the file; creates the file if it does not exist.
"w+"	Read and write. Initializes the file pointer to the beginning of the file; creates the file if it does not exist. Always initializes the file pointer to the beginning of the file before the first write, destroying any existing data.
"a"	Write only. If the file exists, initializes the file pointer to the end of the file; if the file does not exist, creates it and initializes the file pointer to its beginning.
"a+"	Read and write a file, creating the file if necessary; new data is written to the end of the existing data.

Reading from a file

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. 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:

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

One alternative to fread is file, which takes a filename as its parameter and returns an array of all of the lines of the file

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

PHP has another file input function that does not require calling fopen, file_get_contents, which takes the file's name as its parameter. This function reads the entire contents of the file. For example, consider 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. Consider the following statement:

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

This statement reads characters from testdata.dat until it finds a newline character, encounters the end-of-file marker, or has read 99 characters.

Writing to a File

The fwrite function takes two parameters: a file variable and the string to be written to the file. It is possible to include a third parameter, which would be used to specify the number of bytes to be written.

The fwrite function returns the number of bytes written.

```
$bytes_written = fwrite($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 writes the value of its second parameter, a string, to the file specified in its first parameter.

```
file_put_contents("savedata.dat", $str);
```

```
//test.txt

AJAX = Asynchronous
JavaScript and XML
CSS = Cascading Style
Sheets
HTML = Hyper Text Markup
Language
PHP = PHP Hypertext
Preprocessor
SQL = Structured Query
Language
SVG = Scalable Vector
Graphics
XML = EXtensible Markup
Language
```

```
//test.php

<?php
$myfile =
fopen("test.txt", "r")
or die("Unable to open
file!");

echo fread($myfile,filesize("t
est.txt"));
fclose($myfile);
?>
```

Check End-Of-File - feof()

```
<?php
$myfile = fopen("test.txt", "r") or die("Unable to open file!");

// Output one line until end-of-file
while(!feof($myfile))

 {
  echo fgets($myfile) . "<br />";
}
fclose($myfile);
?>
```

fwrite() function

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "Doe John \n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

11.4 Locking Files

- file lock prevents any other access to the file while the lock is set.
- File locking is done in PHP with the flock function, which should sound familiar to UNIX programmers.
- The flock() function locks or releases a file.

Syntax: flock (file, lock , block)

file	Required. Specifies an open file to lock or release
lock	Required. Specifies what kind of lock to use.
block	Optional. Set to 1 to block other processes while locking

- **LOCK_SH - Shared lock (reader). Allow other processes to access the file**
- **LOCK_EX - Exclusive lock (writer). Prevent other processes from accessing the file**
- **LOCK_UN - Release a shared or exclusive lock**
- **LOCK_NB - Avoids blocking other processes while locking**

```
<?php

$file = fopen("test.txt","w+");

// exclusive lock

if (flock($file,LOCK_EX))

  {  fwrite($file,"Write something");

  // release lock

  flock($file,LOCK_UN);

  }else { echo "Error locking file!"; }

fclose($file); ?>
```

12.COOKIES

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.Cookies provide a general approach to storing information about sessions on the browser system itself. The server is given this information when the browser makes subsequent requests for Web resources from the server. Cookies allow the server to present a customized interface to the client. They also allow the server to connect requests from a particular client to previous requests, thereby connecting sequences of requests into a session.

A cookie is a small object of information that consists of a name and a textual value. A cookie is created by some software system on the server.A message from a browser to a server is a request; a message from a server to a browser is a response. The header part of an HTTP communication can include cookies. So, every request sent from a browser to a server, and every response from a server to a browser, can include one or more cookies.At the time it is created, a cookie is assigned a lifetime. When the time a cookie has existed reaches its associated lifetime, the cookie is deleted from the browser's host machine. A particular cookie is information that is exchanged exclusively between one specific browser and one specific server.

setcookie function

A cookie is set in PHP with the setcookie function. This function takes one or more parameters.

setcookie(*name, value, expire, path, domain, secure, httponly*)

The first parameter, which is mandatory, is the cookie's name given as a string. The second, if present, is the new value for the cookie, also a string. If the value is absent, setcookie undefines the cookie. The third parameter, when present, is the expiration time in seconds for the cookie, given as an integer.

The default value for the expiration time is zero, which specifies that the cookie is destroyed at the end of the current session

```
setcookie("voted", "true", time() + 86400);
```

This call creates a cookie named "voted" whose value is "true" and whose lifetime is one day (86,400 is the number of seconds in a day).

```
<?php
$cookie_name = "user";
$cookie_value = "admin123";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");

 // 86400 = 1 day
?>
<html>
<body>

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

The example creates a cookie named "user" with the value "admin123". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer). Retrieve the value of the cookie "user" (using the global variable $_COOKIE).

use the isset() function to find out if the cookie is set:

Delete a Cookie

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

Example

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body></html>
```

In PHP, cookie values are treated much like form values. All cookies that arrive with a request are placed in the implicit $_COOKIES array, which has the cookie names as keys and the cookie values as values. Most browsers have a limit on the number of cookies that will be accepted from a particular server site. In many cases, information about a session is needed only during the session. Rather than using one or more cookies, a single session array can be used to store information about the previous requests of a client during a session. In particular, session arrays often store a unique session ID for a session.

PHP Session

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.Session variables solve this problem by storing user information to be used across multiple pages (e.g. username,pwd, etc). By default, session variables last until the user closes the browser.So; Session variables hold information about one single user, and are available to all pages in one application.

Session Tracking

One significant way that session arrays differ from cookies is that they can be stored on the server, whereas cookies are stored on the client.In PHP, a session ID is an internal value that identifies a session. Session IDs need not be known or handled in any way by PHP scripts.

PHP is made aware that a script is interested in session tracking by calling the session_start function, which takes no parameters. The first call to session_start in a session causes a session ID to be created and recorded. On subsequent calls to session_start in the same session, the function retrieves the $_SESSION array, which stores any session variables and their values that were registered in previously executed scripts in this session.

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html><body>
<?php
// Set session variables
$_SESSION[“uname"] = “admin";
$_SESSION[“pwd"] = “admin123";
echo "Session variables are set.";
?>
</body></html>
```

Modify a PHP Session Variable

To change a session variable, just overwrite it:

Example

```
 <?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// to change a session variable, just overwrite it
$_SESSION[“uname"] = “aministrator";
```

```
print_r($_SESSION);
?>
</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?></body>
</html>
```

Session key/value pairs are created or changed by assignments to the $_SESSION array. They can be destroyed with the unset operator.

```
session_start();

if ($_POST[“uname"])!=“ ”)

{

$_SESSION[“username"] =$_POST[“uname"]) ;

print("You are visitor:  $_SESSION[“username"] <br />");

}
```

Session tracking is relatively simple in PHP.

The session_start function creates a session ID. Session variables are stored in the $_SESSION array.

Sample programs in PHP

```
1. //Wt12.php

 <html><title>A simple math calculator</title>

<body>Insert two numbers in the form and hit submit button <br>

<form action="" method="post">
```

```
     Firstnumber: <input name="num1" type="text" /><br>
     Secondnumber: <input name="num2" type="text" />
     <input type="submit" />
     </form>
     </body>
     </html>
   // Reg.php
   <?php
  if(isset($_POST['submit']))
  {$num1 = $_POST['num1'];
   $num2 = $_POST['num2'];
   $a = $num1 + $num2;
  $b = $num1 - $num2;
  $e=sqrt($num1);
  $f=pow($num1,$num2);
  echo "The sum of the two numbers is ". $a;
  echo "The difference of the two numbers is ". $b;
  echo "The sqrt of the  numbers is ". $e;
  echo "The power of the  numbers is ". $f;
   }?></body></html>
```

2. Program to check whether the number is Armstrong or not using function

```
     <?php
// PHP code to check whether a number is
// armstrong number or not

// function to check whether the number is
// armstrong number or not

function armstrongCheck($number){
    $sum = 0;
    $x = $number;
    while($x != 0)
    {
        $rem = $x % 10;
        $sum = $sum + $rem*$rem*$rem;
        $x = $x / 10;
    }
```

```
    // if true then armstrong number
            if ($number == $sum)
         return 1;

    // not an armstrong number
            return 0;
        }
// Driver Code
        $number = 407;
        $flag = armstrongCheck($number);
        if ($flag == 1)
            echo "Yes";
        else
            echo "No"
        ?>
```

QUESTIONS

1. What are the different ways to create an array in PHP? Explain with example?

2. What is server side scripting?

3. Write a PHP program to check whether the given number is Armstrong or not?

5. Design an HTML form for entering a number by the user. Write a PHP code to display a message indicating, whether the number is odd or even, when clicking on the submit button.

6. Write a PHP script to count the instances of words in a string?

6. What is the significance of cookies in web? How can a cookie be created and destroyed in PHP?

Write a program to find the factorial of a number

```
<html>

<head>

<title>Factorial Program using loop in PHP</title>

</head>

<body>

<form method="post">

Enter the Number:<br>

<input type="number" name="number" id="number">

<input type="submit" name="submit" value="Submit" />

</form>

<?php
```

```
$fact = 1;
//getting value from input text box 'number'
$number = $_POST['number'];
echo "Factorial of $number:<br /><br />";
//start loop
for ($i = 1; $i <= $number; $i++){
   $fact = $fact * $i;
   }
   echo $fact . "<br />";
  ?>
   </body>
   </html>
```

OR

```
<?php

$num = 4;
$factorial = 1;
for ($x=$num; $x>=1; $x--)
{
  $factorial = $factorial * $x;
}
echo "Factorial of $num is $factorial";

?>
```

OUTPUT

```
Factorial of 4 is 24
```

Write a program to swap two numbers

```
<?php

$a=10;
$b=20;

echo "Value of a: $a</br>";
```

```
echo "Value of b: $b</br>";

$temp=$a;
$a=$b;
$b=$temp;

echo "Value of a: $a</br>";
echo "Value of b: $b</br>";

?>
```

Output

```
Value of a: 10

Value of b: 20

Value of a: 20

Value of b: 10
```

Write a program to check whether the number is odd or even

```
<?php

$num=10;

if($num%2==0)
{
 echo "Even number";
}
else
{
 echo "Odd number";
}

?>
```

OUTPUT

Even number

Write a program to find the reverse of a number

```
<?php

$num = 2039;
$revnum = 0;
while ($num != 0)
{
$revnum = $revnum * 10 + $num % 10;
//below cast is essential to round remainder towards zero
$num = (int)($num / 10);
}

echo "Reverse number: $revnum";

?>
```

OUTPUT

```
Reverse number: 9302
```

Write a program to find the reverse of string

```
<?php

$string = "hitesh";
$length = strlen($string);

for ($i=($length-1) ; $i >= 0 ; $i--)
{
  echo $string[$i];
}

?>
```

OUTPUT

hsetih

Write a program to print the Fibonacci series using function

```
<?php
```

```php
function printFibonacci($n)
 {

  $first = 0;
  $second = 1;

  echo "Fibonacci Series \n";

  echo $first.' '.$second.' ';

  for($i = 2; $i < $n; $i++){

    $third = $first + $second;

    echo $third.' ';

    $first = $second;
    $second = $third;

    }
}

/* Function call to print Fibonacci series upto 6 numbers. */

printFibonacci(6);

?>
```

Output

```
Fibonacci Series 0 1 1 2 3 5
```