WWW.KTUNOTES.IN

MODULE - 6

INTRODUCTION TO PHP

→ PHP stands for PHP: Hypertext Preprocessor

→ It is the most popular server side scripting language used for creating dynamic webpages and is mainly used for DB access and form handling.

→ PHP is an opensource technology that provides developers with access to the software's source code & free redistribution rights.

→ Versions

- PHP was created in 1994 by Rasmus Lerdorf to track users at his website. In 1995 he released it as a package called the "Personal Home Page Tools".
- PHP2 (1997) featured built-in DB support and form handling.
- PHP 3 (1997-98) featured a rewritten parser, which substantially increased performance and led to an explosion in PHP use.
- PHP4 (2000) features the new Zend Engine (open source scripting engine that interprets the PHP)

Scanned by CamScanner

and is considerably faster & more powerful than its predecessor.

- PHP 5 (2004) features the new Zend Engine II and it has improved support for object-oriented programming, the PHP Data objects extension and numerous performance enhancements
- PHP 6 (not released) was a failure. It was supposed to add native unicode support to PHP.
- PHP 7 (2015) uses Zend Engine 3 (performance improvements and 64 bit integer support on Windows), uniform variable syntax, AST-based compilation process etc.

Stable version is 7.1 (from 2016 Dec to 2019 Dec)

Future " " 7.2 (from 2017 Nov to 2020 Nov)

→ .php extension

→ PHP is platform independent (support UNIX, LINUX, Windows)

→ PHP also supports a large number of DBs including MySQL.

→ PHP code is embedded directly into XHTML docs. This allows doc. author to write XHTML in a clear & concise manner

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

→ PHP Processor

- When the browser requests a document that includes a PHP script, the Web server calls its PHP processor.
- The PHP processor has 2 modes of operation.

  Copy mode :- When the PHP processor finds markup code in the input file, it simply copies it to the output file.

  Interpret mode :- When the processor encounters PHP script in the input file, it interprets it and send the output of the script (HTML form) to the output file.

- Then the o/p file is sent to the browser, the client never sees the PHP script.

→ Other features & use

- It has driver support for 15 different DB systems.
- Purely interpreted language. But recent implementations performs some precompilation to speed up the Interpretation
- Syntax & semantics is closely related to Javascript So it is easy to learn
- uses dynamic typing (as in Javascript) i.e, variables are not type declared
- It supports distributed object architectures - COM and CORBA

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

- PHP supports mail protocols - POP3, IMAP
- PHP supports both procedural & object-oriented programming.
- PHP has an extensive library of functions, making it flexible & powerful tool for server-side software development.

→ To run a PHP script, install PHP in your system by downloading the recent version. Also keep the PHP files in the Web server's root directory.

GENERAL, SYNTACTIC, CHARACTERISTICS

→ PHP scripts are either embedded in HTM pages or kept in seperate files that are referenced by HTML documents.

→ <?php and ?> → PHP code is written in between these delimiters.

→ If PHP script is stored in a different file, it can be brought into a document with the include construct.

filename

include ("table2.inc"); → The file may contain markup + PHP code [inc indicate the file is to be included by other PHP scripts]

<?php include ("menu.php"); ?>

→ ~~Every variable in PHP begins with a dollar sign.~~

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

→ PHP Variables

- begin with a dollar ($) sign

  $variablename = value ;

- variable name can be a letter or an underscore followed by any no. of letters, digits or underscore

  eg :- $abc , $_length , $first_value.

- PHP variable names are case sensitive

  $total & $TOTAL are different.

→ PHP reserved words

- not case sensitive eg :- while , WHILE & WHILE are same

- The reserved words of PHP are

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

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

→ PHP Comments

Single-line comments – # or //

Multi-line comments – /* and */

→ PHP ~~tem~~ statements are terminated with ;

→ { } are used to form compound statements for control structures

→ Sample PHP program

```
<! DOCTYPE html>
<?php
        $name = "LunaTic";
?>
<html>
    <head>
        <title> A simple PHP doc </title>
    </head>
    <body style = "font-size : 2em">
        <p> Welcome to PHP,
            <?php print ("$name"); ?>!
        </p>
    </body>
</html>
```

o/p :- Welcome to PHP, LunaTic!

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

PRIMITIVES, OPERATIONS & EXPRESSIONS

1. PHP Datatypes

→ 4 scalar types

int, integer → whole numbers, In most cases this is 32 bits.

bool, Boolean → 2 possible values < TRUE / FALSE

double, float → real numbers, Double literals can include a decimal point, an exponent or both.

string → text enclosed in ' or "
In single quoted string, escape sequences & variables are not substituted for their names, but in double quoted string, variables are substituted with their values.

eg:- $sum=10; 'The sum is : $sum' → prints the same string $sum

"The sum is : $sum → print value of $sum i.e. 10

→ 2 compound types

array → group of elements of the same type

object → Group of associated data & methods.

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

→ 2 Special types

Resource → An external data source

NULL → No value

(2) Scalar Type Conversions

→ 2 types

1) Explicit type conversions → Programmer does it.

- It can be done in 3 ways

a) typecasting

format: (new datatype) $variablename ;

eg:- (int) $sum ;

b) Use of intval(), doubleval() or strval()

format: intval/doubleval/strval ($variablename);

$sum = 4.777

eg:- intVal ($sum);

It returns 4.

c) use of settype()

format: settype ($variablename, new datatype);

eg:- settype ($sum, ~~float~~ double);

settype() can result in loss of data (truncation) and also the value of the variable is changed after each datatype conversion.

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

2) Implicit type conversions (Coercions)

- Automatic type conversion
- eg :- between integer & double types
  " Boolean & other scalar types
  " numeric & string types
- eg :-

```
<?php
$V1 = "56";
$V2 = 12;
$c = $a / $b;
print "$c";
?>
```

o/p is : 4.66 (The result is converted to floating point

KTUNOTES.IN

3) Arithmetic Operators & Expressions

→ operators → +, −, *, /, %, ++, −−

→ For +, −, * if operand1 is integer & operand2 is integer ; result is integer.
If one operand is double ; result is double.

→ In case of '/', integer division results in a double value.

→ % operation → The values are coerced to integers.

Downloaded from Ktunotes.in Scanned by CamScanner

→ Some of the PHP predefined functions that operate on numeric values are

Function	Parameter type	Returns
floor	double	Largest integer less than or equal to the parameter
ceil	"	Smallest " " "
round	"	Nearest integer
srand	[integer]	Initializes a random-number generator with the parameter
rand	[two numbers]	A pseudorandom no. greater than the 1st parameter & smaller than the 2nd.
abs	number	Absolute value of the parameter (-ve to +ve)
min	one or more numbers	smallest number
max	"	largest "

→ Assignment operator (=) → we can use +=, /= etc

→ define () creates a named constant eg. define ("VALUE", 5);

(4) String Operations

→ Catenation operator (.) :- combines multiple strings.

eg :-
```
$a = "Hello";
$b = $a . "World";
print "$b";
```

O/P
Hello World

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

→ String variables can be treated like arrays to access individual characters. (index starts from zero).

eg:- $str = "Apple";
print "$str{3}"

o/p
l

→ Some of the php functions that operate on strings are.

strlen (string) → returns the no. of characters in the string.

strcmp ($s1, $s2) → returns 0 → if equal
<0 → s1 < s2
>0 → s1 > s2

strpos ($s1, $s2) → returns the starting position of 2nd string in 1st string. otherwise false is returned.

substr ($s1, integer [integer]) → extracts a substring, 2nd parameter is starting position & 3rd parameter is no. of characters

eg:- $str = "Apples are good";
$sub = substr ($str, 7, 1);

It returns 'a'

chop ($s1) → The parameters "\0", "\t", "\n", "\r" etc are removed from right end.
Also we can specify a 2nd parameter to chopped off

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

trim ($s1, [$s2]) → characters are removed from both ends.

ltrim (") → " " beginning

strtolower ($s1) → uppercase letters are converted to lowercase.

strtoupper ($s1) → lowercase " " uppercase

(5) Output

→ All outputs from the PHP processor must be in the form of HTML or XHTML.

→ print () function

- used to create simple unformatted output
- ~~parant~~ () can be omitted.

  eg:- print "$sum";
  print ($sum);

- Automatic conversion to string or coersion is done on parameter.

  eg:- print (47);
  will print string "47".

- HTML tags can be embedded in print ()

  eg:- print "<h2> $ sum </h2>";

→ echo ()

echo () serves the same function of print ()

eg:- echo "$sum";
echo ("$sum");

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

→ printf () function

- PHP borrows printf () from C.
- format :
  printf (literal_string , param1, param2, ...)

The literal string can include type specifiers like
s – string
d – integers
f – float & double.

- eg :-
```
$day = "Tuesday";
$high = 79;
printf ("The high on %7s was
        %3d", $day , $high);
```

KTUNOTES.IN

(6) Control Statements

→ Relational Operators. (8 Nos).

<, >, >=, <=, !=, == and
===, !==.

→ Boolean Operators

and, or, xor, !, && and ||

- The precedence of 'and' and 'or' is lower than that of && and || (operators is same)

Downloaded from Ktunotes.in Scanned by CamScanner

→ Selection statements [explain with its format & example]

- if
- if - else
- if - elseif
- switch

→ Loop statements

- while
- do - - - while
- for
- foreach
- break
- continue

IMP

format: foreach (array-variable as $value)

= = =

(or) foreach (array-variable as $key => $value)

= = =

eg:-
```
<? php
$arr = array (1, 2, 3, 4);
foreach ($arr as $value)
{
    $value = $value * 2;
    print "$value";
}
?>
```

o/p :- 2, 4, 6, 8.

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

eg 2 :-

```
<?php
    $arr = array (1, 2, 3, 4);
    foreach ($arr as $key => $value)
        echo "$key => $value";
?>
```

o/p :-

```
0 => 1
1 => 2
2 => 3
3 => 4
```

ARRAYS

→ PHP arrays are the ultimate in flexible built-in data structures because

1) they have a logical structure similar to an array in other languages (ie, the key is nonnegative integers in descending order)
2) They have a logical structure similar to a hash (ie, its keys are strings and the order of elements is determined with a hashing function).

→ PHP Array names begin with a '$' symbol.

→ Each array element consist of 2 parts < key / value

→ Note :- PHP array can have some elements with integer keys & some with string keys

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

Array Creation – 2 methods

1) In PHP, an array is created when you assign some values to a array variable

→ eg: 1) $list [0] = 17 ; → index is explicitly given

2) $list [] = 17 ; → index is implicitly taken as 0.

3) $list [1] = "Today is my birthday" → index is implicitly taken as 2.

$list [] = 42

2) using the 'array' construct

→ The parameter can be
- none
- array elements
- key, value pair.

eg:- 1) $list = array (); → array with no elements

2) $list = array(17, 24, 45, 91); → array with 4 elements

0	17
1	24
2	45
3	91

3) $list = array(1=>17, 2=>24, 3=>42, 4=>91); ⇒

1	17
2	24
3	42
4	91

4) $ages = array("Joe"=> 42, "Mary"=> 41);

Joe	42
Mary	41

5) $stuff = array("make" => "Cessna", "model" => "C210", "year" => 1960, 3 => "sold"); ⇒ mixture of strings & integer array

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

Accessing Array Elements

→ Individual array elements are accessed by following the array variable name with an index enclosed in [ ].

eg:-
```
$first = array ();
$first[0] = "zero";
$first["May"] = 29;
```

→ The 'list' construct can be used to assign multiple elements of an array to scalar variable in one statement.

eg:-
```
$trees = array ("oak", "pine", "binary");
list ($hardwood, $softwood, $data_struct)
                = $trees;
```

then
```
$hardwood = "oak"
$softwood = "pine"
$data_struct = "binary".
```

Functions for dealing with arrays

→ unset ()

- delete the whole array or individual elements.

eg:-
```
$list = array (2, 4, 6, 8);
unset ($list[2]);        // deletes 6
unset ($list);           // deletes entire array.
```

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

→ **array_keys ()**

- It takes an array as its parameter and returns an array of the keys of the given array.

array_values ()

- returns an array of the values of the given array.

In both functions, index of resultant array starts from 0.

eg:- $high = array ("Mon" => 74", "Tue" => 70).

$days = array_keys ($highs);

$temps = array_values ($highs);

So here $days = ("Mon", "Tue") and

$temps = (74, 70)

→ **array_key_exists ()**

- used to check the existence of an element of a specific key.
- It returns a Boolean value.
- eg:- $highs = array ("Mon" => 74, "Tue" => 70, "Wed" => 67);

$check = array_key_exists ("Tue", $highs);

Here $check will have a value TRUE.

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

→ **is_array ()**

- It takes a variable as its parameter & returns TRUE if the variable is an array, FALSE otherwise.

→ **in_array ()**

- It takes 2 parameters – an expression & an array and returns TRUE if the value of the expression is a value in the array; otherwise returns FALSE.

→ **sizeof ()**

- returns the no. of elements in the array.
- eg:- $list = array (1, 2, 3, 4, 25);
  $len = sizeof ($_list). // $len = 5

KTUNOTES.IN

→ **explode ()**

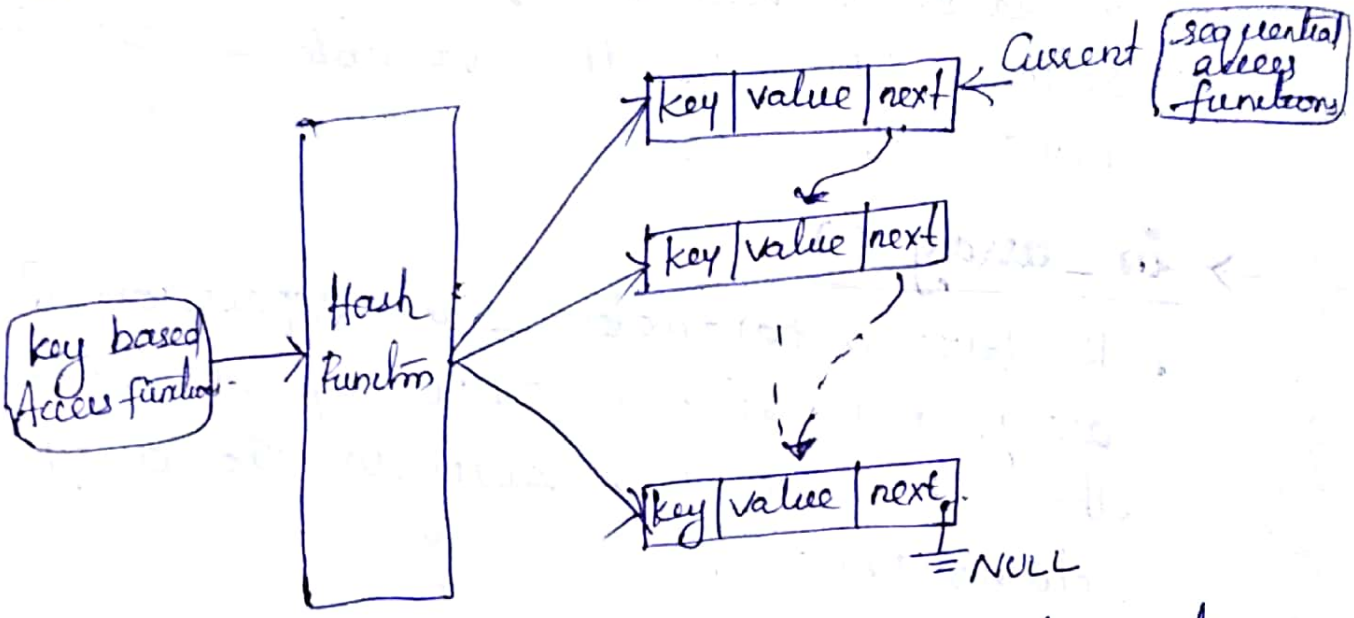
- explodes a string into substrings & store it in array.
  explode (delimiter, string);
- eg:- $str = "April in Paris";
  $words = explode (" ", $str);
  Here o/p is $words = ("April", "in", "Paris").

→ **implode ()**

- reverse operation.
- eg: $str = implode (":", $words);
  then $str = "April : in : Paris"

Downloaded from Ktunotes.in Scanned by CamScanner

Logical Internal structure of an array.



explain :- • linked list cell ⇒ array elements ie key, value.

• next → points to the next element in the array

• Access to elements through string keys are implemented through the hashing function.

• Sequential access is possible using 'next' pointer if the keys are integers.

Sequential Access to Array Elements

→ "current" pointer

Whenever an array is created, a pointer is initialized to reference the first element of the array.

→ current () function :- returns the element being referenced by the current pointer

KTUNOTES.IN

Downloaded from Ktunotes.in Scanned by CamScanner

eg:- $cities = array ("Bangalore", "Chennai", "Goa");
$city = current ($cities)
print ("The first city is $city
");

O/P :- The first city is Bangalore

→ next () function → used to move the current pointer. If current is pointing to last element of the array, then next () returns FALSE
eg:- The following code lists all elements of the array.

```
$city = current ($cities);          // $city = "Banglore"
print ("$city <br/>");
while ($city = next ($cities))
        print ("$city <br/>");
```

If the array element itself is 'FALSE', the loop ends. The solution to this problem is the use of for-each loop.

→ prev () function → used to move the current pointer backward

→ reset () function → sets the current pointer to the first element of the array

→ end () function → returns the value of the last element.

Scanned by CamScanner

→ **key () function** :- The parameter is the name of the array & returns the key of the current element of the array.

→ **array_push & array_pop functions** :- implement stack in an array. NULL is returned if the array is empty.

→ **foreach** statement is used to build loops that process all of the elements of an array.
[format & eg :- refer control structures section]

→ eg:- printing array elements using reset (), key (), and next ()

```
<?php
$first = array ("1", "2", "3", "4");   //string array
for (reset ($first) ; $element = key ($first);
                next ($first))
{
    print ("$element is $first [$element]
                <br />);
}.
?>
```

Scanned by CamScanner

Sorting Arrays

→ **sort () function** :- It takes an array as parameter sorts the values in the array, replacing the keys with the numeric keys 0, 1, 2, ... The string values comes before numeric values (heterogeneous array)

→ **asort () function** :- used to sort arrays that correspond to hashes. It sorts the elements of a given array by their values, but keeps the original key-value associations. Here also string values comes before numeric values.

→ **ksort () function** :- sorts its given array by keys, rather than values.

→ **rsort (), arsort (), krsort ()**

They sort in reverse order.

eg:-
```
<?php
$arr = array ("Fred" => 31, "Ali" => 27,
              "John" => "Rich", "Betty" => 42);

foreach ($arr as $key => $value)
    print ("[$key] => $value <br/>");

$new = $arr;
sort ($new);

foreach ($new as $key => $value)
    print ("[$key] => $value <br/>");
```

→
[Fred] => 31
[Ali] => 27
[John] => Rich
[Betty] => 42.

→
[0] => Rich
[1] => 27
[2] => 31
[3] => 42

Scanned by CamScanner

```
$new = $arr;
asort ($new);
foreach ($new as $key => $value)
    print ("[$key] => $value <br/>");
```

→ [John] => Rich
[Ali] => 27
[Fred] => 31
[Betty] => 42.

```
$new = $arr;
ksort ($new);
foreach ($new as $key => $value)
    print ("[$key] => $value <br/>");
```

→ [Ali] => 27
[Betty] => 42
[Fred] => 31
[John] => Rich.

```
?>
```

FUNCTIONS IN PHP

→ The general form of PHP function is

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

→ Function overloading is not allowed & functions cannot be redefined.

→ Functions definitions can be nested.

→ Function names are not case sensitive. i.e, we cannot define two functions with name 'sum' & 'SUM'.

→ <u>return</u> statement specifies the value to be returned to the caller.

→ We can keep all the function definitions in a seperate file & then copy that file into those documents where they are requested. This can be done using <u>'include'</u> function.

Scanned by CamScanner

→ Parameters

- Actual parameters → appears in function call. It can be a variable / expression
- Formal " → " " definition. It must be a variable name.

- The no. of actual parameters & no. of formal parameters need not match. If less actual parameters they will be treated as unbound variables & if more actual parameters, the extras will be ignored.

- parameter passing mechanisms

1) pass by value :- only a copy of actual para. are copied to formal para. ~~so the~~ changes to the formal para, doesn't reflect in actual parameters (ie, a one way communication)

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

Suppose the function call is ~~max~~
$res = max_abs ($a, $b);

~~Then~~ Here only $first & $second are updated not a & b.

Scanned by CamScanner

2) Pass by reference

- provides a 2 way communication. Here the address of actual parameters are passed, rather than its value.
- So if the formal parameter is changed, then that change is reflected in the actual parameter also.
- In PHP, 2 ways to specify pass by reference
  1) Add a '&' to the beginning of formal parameters
  2) Add a '&' to the " " actual para.

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

→ Scope of variables

- local scope → The variables defined inside a function, we can use the same name for a variable as a variable used outside the function.

  eg:- (Write one eg code)
- global scope - visible throughout the program

  eg:- (Write an eg code)

Scanned by CamScanner

→ The lifetime of variables

- lifetime of local variables → the time from the variable is first used until the function's execution terminates

- In PHP, a local variable in a function can be specified as static by declaring it with the reserved word static

The lifetime of static variable → is until the browser leaves the document in which the PHP script is embedded.

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

This fn displays the no. of times it has been called.

Scanned by CamScanner

PATTERN MATCHING

→ PHP includes 2 kinds of string pattern matching using regular expressions

1) Based on POSIX (Portable OS Interface) regular expression
2) Based on Perl regular expression.

→ PHP provides functions ereg() and preg_match() which use regular expressions to search a string for a specified pattern.

→ preg_match()

- It takes 2 arguments < 1) Perl style regular expression 2) string to be searched.

```
eg:- if (preg_match("/^PHP/", $str))
        print ("\$str ~~does not~~ begins with PHP <br/>");
     else
        print ("\$str ~~begins~~ does not begin with PHP <br/>")
```

→ preg_split()

- 2 arguments < 1) Perl style pattern as a string 2) string to be split.
- return value is an array

```
eg:- $string1 = "apple : orange : banana";
     $fruits = preg_split("/:/", $string1);
```

Now the array $fruits = ("apple", "orange", "banana")

Scanned by CamScanner

→ ereg ()

eg:- $search = "Now is the time"; // test string

if (ereg ("Now", $search)) // search 'Now' in string $ search. If found return true

if (ereg ("^Now", $search)) // Caret symbol (^) search for pattern 'Now' in the beginning of string

if (ereg ("Now$", $search)) // $ sign search for pattern 'Now' at the end of string

→ Bracket Expressions

- lists of characters enclosed in [ ] that match a single character from the list. Ranges can be specified by supplying the beginning and the end of the range seperated by a dash (-)

[a-z] → matches any lowercase letters

[A-Z] → " " uppercase "

[[:<:]] and [[:>:]] → match the beginning and end of a word.

eg:- if (ereg ("[[:<:]] ([a-z A-Z]*ow [[:>:]]", $search, $match))

This stmt search for any word ending in 'ow'

stores the matched result in this array

Scanned by CamScanner

- <u>Quantifiers</u> → denote how often a particular character can appear in a match.

Some of the PHP quantifiers are.

$\{n\}$	exactly n times	eg:- $p^{\{2\}}$ → Any string containing a sequence of 2 P's
$\{m,n\}$	Between m & n times	$p^{\{2,3\}}$ → 2 or 3 p's
$\{n,\}$	n or more times	$p^{\{2,\}}$ → atleast 2 P's.
+	One or more times	(same as $\{1,\}$)
*	zero or more times	(" $\{0,\}$)
?	zero or one time	(" $\{0,1\}$)

- <u>Searching for multiple instances of a pattern</u>

* The ereg () matches only the first instance of the pattern.
* So remove the matched instance before calling ereg () again. ereg_replace () is used for this.

eg:- Search for any words beginning with 't'

```
while (eregi ("[[:<:]] (t [[:alpha:]]) [[:>:]]
                    $search, $match))
{
    print ($match[1]. " ");
    $search = ereg_replace ($match[1], " ",
                    $search);
}
```

Scanned by CamScanner

* Here alpha is character class indicating alphabets
Other character classes are

[:alnum:]	Alphanumeric characters [a-z A-Z] or [0-9]
[:alpha:]	alphabets [a-z A-Z]
[:digits:]	digits [0-9]
[:space:]	whitespace
[:lower:]	lowercase letters.
[:upper:]	upper " "

* eregi → specifying case insensitive matches
* ereg_replace() → 3 arguments - the pattern to match, a string to replace the matched string & the string to search.

VIEWING CLIENT/SERVER ENVIRONMENT VARIABLES

→ Environment Variables contain information about a script's environment such as the client's Web browser, the HTTP host & the HTTP connection

→ Some useful global arrays are.

variable name	description
$_SERVER	data about the currently running server
$_ENV	data about the client's environment

Scanned by CamScanner

$_GET	data posted to the server by the GET method.
$_POST	data posted to the server by the POST method
$_COOKIE	Data contained in cookies on the client's computer.
$GLOBALS	Array containing all global variables

FORM HANDLING

→ XHTML forms enable webpages to collect data from users and send it to a webserver for processing

→ The PHP script that processes the form data may reside in the same HTML page or in another seperate file.

→ The steps are

- The browser presents a form to be filled by the user
- The user fills the form and click the submit button
- The php page specified in the 'action' attribute of <form> get executed.
- Within the PHP script, the elements of the form are accessed & processed (using $_POST or $_GET)
- The result is sent back to the browser

Scanned by CamScanner

→ The value of form elements are accessed in PHP using the following methods

1) $_POST ["form element name"]
2) $_GET ["form element name"]
3) extract ($_POST);
   extract ($_GET);
   → Creates a variable-value pair corresponding to each key_value pair in the associative Array $_POST.

eg:- <u>data.html</u> → This html doc. contains a form to request user details and the form datas are processed by a 'form.php' script.

```
<html>
    <head><title> - - - - - - - </title>
    </head>
    <body>
        <form method="post" action="form.php">
            Name: <input type="text" name="fname"/>
            <br />
            Email: <input type="text" name="email"/>
            <br />
            Book: <select name="book">
                <option> C++ </option>
                <option> Java </option>
            </select>
            <input type="submit" value="Register"/>
        </form>
    </body>
</html>
```

o/p

Name :
Email :
Book :
Submit

Scanned by CamScanner

form.php

```
<?php
        extract ($_POST);
        print (" $fname . " " . $email . " " .
                $book ");
?>
```

Another implementation of form.php is

```
<?php
        $first_name = $_POST ["fname"];
        $email_id   = $_POST ["email"];
        $book_name  = $_POST ["book"];
```

Scanned by CamScanner

Connecting to a DB

→ DB connectivity allows sym. admin to maintain and update ~~such~~ information such as user accounts, passwords, credit card numbers, mailing list etc.

→ PHP offers built-in support for a wide variety of DB.

→ steps are
- The client selects a DB field name that is sent to the Web server.
- The PHP script is then executed; the script builds the query, queries the DB and sends a record set in the form of XHTML to the client.

→ Example

data.html → posts ~~data~~ form data containing a DB column to the server.

database.php → processes the form data.

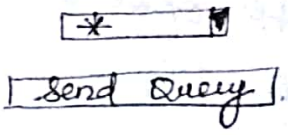
DB contains details of books such as ID, Title, Category.

Scanned by CamScanner

data.html

```
<html>
    <head> <title> Sample DB Query </title>
    </head>
    <body>
        <form method="post" action="database.php">
        <select name="select">
            <option selected="selected> * </option>
            <option> ID </option>
            <option> Title </option>
            <option> Category </option>
        </select>
        <input type="submit" value="Send Query"/>
        </form>
    </body>
</html>
```

→ displays all details.



database.php

```
<?php
    extract ($_POST);
    // Connect to MySQL (parameters are servers host name, a username & a password).
    if (!($database = mysql_connect ("localhost",
                "httpd", " ")))
        die ("Could not connect to database");
    // The database handle is assigned to $database.

    // open the product DB to be queried.
    if (!mysql_select_db ("Products", $database))
        die ("could not open Products database");
```

Scanned by CamScanner

```
// Build the query.
$query = "SELECT". $select . "FROM Books";

// Query Products DB.
if (!($result = mysql_query ($query, $database)))
{   print ("could not execute query ! <br/>");
    die (mysql_error ());
}

// Search Results.
for ($counter = 0; $row = mysql_fetch_row ($result);
    $counter ++)
{   foreach ($row as $key => $value)
        print (" $value ");
        print (" ");
}

// close the connection
    mysql_close ($database);

?>
```

Scanned by CamScanner

COOKIES

→ A Cookie is a text file that a web site stores on a clients' computer to maintain information about the client during and between browsing sessions.

→ It records user preferences and other information that the website can retrieve during the clients' subsequent visits.

→ uses

- Many websites use cookies to store clients' zip codes. and provides weather reports and news updates tailored to the user's region
- Websites use cookies to track information about the client activity.
- Analysis of information collected via cookies can reveal the popularity of websites or products.
- Marketers can use cookies to determine the effects of advertising campaigns.

→ Security & Privacy of Cookies

- Cookies are text files stored on user's hard drives by the websites. and any program can read it. Thus critical information such as credit card numbers and passwords should not be stored in cookies.

Scanned by CamScanner

- A server can access only the cookies that it has placed on the client.
  ie, a cookie placed by www.dietel.com cannot be accessed by www.prenhall.com.
- A cookie has an expiration date, after which the web browser deletes it.
- Users can disable cookies in their web browsers. But sometimes it becomes impossible for the user to interact with websites that rely on cookies to function properly.

→ The information stored in the cookie is sent back to the Web server from which is originated whenever the user requests a webpage from that particular server. The Web server can send the client XHTML output that reflects the preferences or information that is stored in the cookie.

Writing Cookies

eg:- cookie.html - display an XHTML form that allows a user to enter a name, height and favourite color. When the user clicks the 'Write Cookie' button, the Cookie.php script executes.

Name : [LunaTic]
Height : [5 feet 7 inches]
favorite color : [Blue.]
[Write Cookie]

[Write cookie.phtml file]

Scanned by CamScanner

cookies.php

```
<?php
    // extract the values posted by client.
        extract ($_POST);

    // Write each form field's value to a cookie &
    // set the cookie's expiration date.

    setcookie ("Name", $NAME, time () + 60 * 60 * 24 * 5)
    setcookie ("Height", $HEIGHT, time () + 60 * 60 * 24 * 5)
    setcookie ("color", $COLOR, time () + 60 * 60 * 24 * 5)

?>
<html>
    <head> <title> . - - - . </title> </head>
    <body>
        <p> The cookie has been set with the
            following data : </p>.

        <b> Name : </b>
        <?php print ($NAME) ?> <br />
        <b> Height : </b>
        <?php print ($HEIGHT) ?> <br />
        <b> Favorite color : </b>
        <?php print ($COLOR) ?> <br />

        <p> click <a href = "readCookies.php" > here </a>
            to read the saved cookie </p>
    <body>
</html>.
```

Scanned by CamScanner

setcookie ( )

setcookie (Name of the cookie to be set ①, value to be stored in the cookie ②, [Expiration date of the cookie ③])

- If no expiration date is specified, the cookie only lasts until the end of the current session [ie, until the user closes the browser].
  In this eg, 5 days is set.
- If only the 'name' argument is passed to function, the cookie is deleted from the client's computer.
- setcookie ( ) must appear before <html> tag.

Reading an Existing Cookie

readCookies.php

```
<html> <head> <title> ..... </title> </head>
    <body>
        <b> The following data is saved in a
            cookie on your computer </b>
<?php
    foreach ($_COOKIE as $key => $value)
    {
        print ("$ key<br>");
        print ("$ value<br>");
    }
?>
</body>
</html>
```

print all cookie values (time & value) is

echo $_COOKIE ["Name"];

or echo $_COOKIE ($NAME);

Scanned by CamScanner

→ PHP creates the superglobal array $_COOKIE which contains all the cookie values indexed by their names

Scanned by CamScanner

SESSION TRACKING

→ A session is a way to store information (in variables) to be used across multiple pages.

→ ~~Unlike a cookie, the information is not stored on the users computer.~~

→ Session variables hold information about one single user and are available to all pages in one application.

→ A visitor accessing a web application is assigned a unique id called session id, which is either stored in a cookie on the client side or is propogated in the URL. ~~Rather they can be stored on the server side.~~

→ In PHP, a session ID is an internal value that identifies the session.

→ In PHP, a session is started with the session_start () function. A call to this function causes a session id to be created & recorded.

→ Session variables are set with the PHP global variable $_SESSION. Session key-value pairs are created or changed by assignments to the $_SESSION array.

→ A session variable can be destroyed with the unset operator.

Scanned by CamScanner

→ The ~~entire~~ PHP session can be destroyed by session_destroy function.

example

session1.php (In this page, we start a new PHP session & set some session variables)

```
<?php
    // Start the session
        session_start();
?>
<!DOCTYPE html>
<html>
    <body>
        <?php
            // Set session variables.
            $_SESSION["favcolor"] = "green";
            $_SESSION["favanimal"] = "cat";
            echo "Session variables are set";
        ?>
    </body>
</html>
```

Scanned by CamScanner

session2.php (From this page, we access the session info we set on the first page)

The session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page [session_start ()].

```
<?php
    session_start ();
?>
<!DOCTYPE html>
<html>
    <body>
        <?php
            // Printing session variables that were set on previous page
            echo ("Favourite color is". $_SESSION["favcolor"]);
            echo ("Favourite animal is". $_SESSION["favanimal"]);
        ?>
    </body>
</html>
```

Scanned by CamScanner

→ Another method to print the values of all session variables is

```
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
    <body>
        <?php
            print_r($_SESSION);
        ?>
    </body>
</html>
```

→ Destroying a PHP session

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

Scanned by CamScanner