**PHP**

**Server-side Scripting**

## Dynamic vs. Static Web Sites

Roughly speaking, there are two kinds of Web sites: those with static content and those with dynamically generated content. These are also called static Web sites and dynamic Web sites, or Web sites with static pages versus Web sites with dynamic pages.

*What is Web content? Content is everything that can appear on a Web page: text, graphics, form fields, hyperlinks to other pages, navigation buttons, menus, etc.*

### [Static Web Sites](http://www.robertz.com/WebDesign/StaticWeb.htm)

For a static-content Web site, all content appearing on Web pages is placed manually by professional Web developers. This is also called "design-time page construction," because the pages are fully built while the site is being developed. Static-content Web site is developed and then maintained by experienced professionals. Such Web site usually costs less when initially developed, but then all future changes still have to be done by Web professionals. Therefore a static Web site can be more expensive to maintain, especially when you want to make frequent changes to your site.

### [Dynamic Web Sites](http://www.robertz.com/WebDesign/DynamicWeb.htm)

On the other hand, pages in a dynamic-content Web site are constructed "on the fly" when a page is requested from a Web browser. Dynamic-content Web site, while still developed by professionals, can be maintained directly by you, our customer. Such Web site initially costs more to develop, but then you don't have to pay Web professionals every time you need to change something on your site. If you plan to make frequent changes to your site, you most likely will be better off with a dynamic Web site.

**Static v Dynamic Website Design**

There are basically two main types of website - static and dynamic.  
A static site is one that is written in plain HTML and what is in the code of the page is what is displayed to the user.

A dynamic site is one that is written using a server-side scripting language such as PHP, ASP, JSP, or Coldfusion. In a dynamic site the content is called in by the scripting language from other files or from a database depending on actions taken by the user.

**Server**

In general, a server is a computer program that services to other computer programs in the same or other computers. The computer that a server programs runs in is also frequently referred to as a server (through it may contain a number of server and client programs). In the client/sever programming model, a server is a program that awaits and fulfills requests from client program in the same or other computers. A given application in a computer may function as a client with requests for services from other programs and also as a server of request from other programs.

Specific to the Web, a Web server is the computer program (housed in a computer) that serves requested HTML pages or files. A Web client is the requesting program associated with the user. The Web browser in your computer is a client that requests HTML files Web servers user.

**Web server**

A Web server is a program that, using the client/server model and the World Wide Web's Hypertext Transfer Protocol(HTML), serves the files that form Web pages to Web users (whose computers contain HTTP clients that forward their requests). Every computer on the Internet that contains a Web site must have a Web server, and Microsoft's Internet Information Server (IIS). Other Web servers include Novell's Web Server for users of its NetWare operating system and IBM's family of Lotus Domino servers, primarily for IBM's OS/390 and AS/400 customers.

Web servers often come as part of a larger package of Internet-and intranet-related programs for serving e-mail, downloading requests for File Transfer Protocol (FTP) files, and building and publishing Web pages. Considerations in choosing a Web server include how well it works with the operating system and other servers, its ability to handle server-side programming, security characteristics, and publishing, search engine, and site building tools that may come with it.

**IIS**

IIS (Internet Information Server) is a group of Internet servers (including a Web or Hypertext Transfer Protocol server and a File Transfer Protocol server) with additional capabilities for Microsoft's Windows NT and Windows 2000 Server operating systems. IIS is Microsystems, to compete in the Internet server market that is also addressed by Apache, Sun Microsystems, O'Reilly, and others. With IIS, Microsoft includes a set of programs for building and administering Web sites, a search engine, and support for writing Web-based applications that access databases. Microsoft points out that IIS is integrated with the Windows NT and 2000 Servers in a number of ways, resulting in faster Web page serving.

A typical company that buys IIS can create pages for Web sites using Microsoft's Font Pages product (with its WYSIWYG user interface). Web developers can use Microsoft's Active Server Page (ASP) technology, which means that applications- including Active controls –can be imbedded in Web pages that modify the content sent back to users. Developers can also write programs that filter requests and get the correct Web pages for different users by using Microsoft's Internet Server Application Program Interface (ISAPI) interface. ASPs and ISAPI programs run more efficiently than common gateway interface (CGI) and server-side include (SSI) programs, two current technologies. (however, there are comparable interfaces on other platforms.)

Microsoft includes special capabilities for server administrators designed to appeal to Internet service providers (ISPs). It includes a single window (or "console") from which all services and users can be administered. It's designed to be easy to add components as snap-ins that you didn't initially install. The administrative windows can be customized for access by individual customers.

**PHP**

# Introduction

PHP was developed in 1995 Rasmus Lerdof Asia asp supportive file but later viewing its efficiency to upload all the programming performances it is solely developed as server side script and was named Personal home page but later in 1997 it was revised to “Hypertext pre-processor”. It helps us to develop dynamic webpage or website which can easily interact with database and interface and has got easier syntax then asp syntax. It is a case sensitive scripting language. It’s extension is “.php”. Today PHP is everywhere you look in the web, with an estimated 100 million PHP pages. You can handle button clicks, radio button selections, and list box choices with ease using PHP. Java script can’t write files, and it can’t work with data on the server. Web server are Microsoft Internet Information Server (IIS), Apache, Xitami, and Sambar Server etc.

Wordpress, zoomla, Magento, Drupal etc software are developed from PHP.

## What is PHP?

* PHP stands for **P**HP: **H**ypertext **P**reprocessor
* PHP is a server-side scripting language, like ASP
* PHP scripts are executed on the server
* PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
* PHP is an open source software
* PHP is free to download and use

## Why PHP?

* PHP runs on different platforms (Windows, Linux, Unix, etc.)
* PHP is compatible with almost all servers used today (Apache, IIS, etc.)
* PHP is FREE to download from the official PHP resource: www.php.net
* PHP is easy to learn and runs efficiently on the server side

**Syntax of PHP:**

We can be used two tags of PHP code

1. . <?php….?> or <? ………. ?>

<?php

echo("The is the example one to show text");

?>

<?

echo ("this is the second syntax of php");

?>

2. <script language="php"> …………….. </script>

<script language="php">

echo ("this is the third syntax of php");

</script>

**HTML Embedding**

<html>  
    <head>  
        <title>Example</title>  
    </head>  
    <body>  
  
        <?php   
        echo "Hi, I'm a PHP script!";   
        ?>  
    </body>  
</html>

Each code line in PHP must end with a **semicolon**. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: **echo** and **print**.

**Note:** The file must have the .php extension. If the file has a .html extension, the PHP code will not be executed.

**Single or double quotes:** You can pass text to the echo statement using single or double quotes.

<?php

echo "welcome to PHP.";

echo 'welcome to Microsoft.';

?>

parentheses: ( ) : You can pass text inside parentheses to echo as well.

echo (“Welcome to PHP.”);

**Using HTML tag in PHP:**

<html>

<head>

<title>

display text

</title>

<body>

<h1>Using HTML tag inPHP

</h1>

<br>

<?php

echo “<i>Welcome </i> <br>”;

echo “<u>to </u><br>”;

echo “<b>MICROSOFT</b>”;

?>

</body>

</html>

Variable:

It is a name provided to the memory where we can store the data nd will be varied according to your program construction. It is also a case sensitive in nature. Variables are used for storing a values, like text strings, numbers or arrays, functions. $ sign must be at the beginning of the variable. In that case it will not work.

## Variable Naming Rules

* A variable name must start with a letter or an underscore "\_"
* A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and \_ )
* A variable name should not contain spaces. If a variable name is more than one word, it should be separated with underscore ($my\_string), or with capitalization ($myString)

Eg. Of a variable with a string, and a variable with a number:

<?php

$txt = "Hello World!";

$number = 16;

?>

i.e.

$name="Kritagya";

echo="$Kritagya";

**<?php**

**$var="kritagya"; // use small v**

**$Var="krishna"; //use capital V**

**echo "$var, $Var";**

**?>**

<?

$name= "Ram";

$add= "Kathmandu";

echo "You placed the name" .$name. "and lives in" .$add;

$add="patan";

echo "<br> but sometimes lives in" .$add;

?>

# Data Type:

It is a type of data which is stored in the variable.

Types:

1. String :In this data type all the character are included, a-z, A-Z, 0-9 & all symbols.

Holds text like “Welcome to MICROSOFT”

1. Number : Integer (Whole number) Float – (Decimal Number)
2. Boolean : Holds true & false value. Where true means 1 and false means 0.
3. Null : Holds a value of NULL not zero(0)
4. integer : Hold numbers like -1, 0 5 and so on
5. array : Hold arrays of data items. $info=array{"Krishna",35,'Kathmandu',$gender,37}
6. object : Holds programming object.
7. float : Hold floating point numbers (“doubles”) like 3.232

# Comment:

Comment does not allow the browser to interperate the connected codes and will place it within the coding area as well as it is very much helpful in debugging the bugged codes.

Types of Comment

1. Single line comment (//……) or #
2. Multi line comment (/\*…….. \*/)

Note: “<!......>” this comment is used in Javascript or in HTML codes

Eg.

<?

$ab=”Ram”; // this is a variable definition

echo “The value is” .$ab;

echo “<br> This is first statement”;

/\* echo “<br> This is second statement”;

echo “<br> This is third statement.

Echo “<br> This is fourth statement”;

\*/

?>

# Operator:

Symbols used for logical as well as calculative operator in program are operator.

Types of Operator

1. String Operator (Concatnation operator)
2. Arthmetic Operator
3. Assignment Operator
4. Increment/Decrement Operator
5. Comparision Operator
6. Logical Operator
7. Combined operator

1. The Concatenation Operator

It is a kind of string operator in PHP. The concatenation operator (.)  is used to put two string values together. To concatenate two variables together, use the dot (.) operator:

<?php

$txt1="Hello World";

$txt2="1234";

echo $txt1 . " " . $txt2;

?>

**2. Arthmetic Operator**

a) + (addition)

b) -(substraction)

c) \* (multipplication)

d) / (division)

e) % (modulus or remainder)

a) addition

<?php

$a=10;

$b=15;

$c=$a+$b;

echo "$c";

?>

b) Substraction:

<?php

$a=50;

$b=15;

$c=$a-$b;

echo "$c";

?>

c) Multiplication

<?php

$a=10;

$b=15;

$c=$a\*$b;

echo "$c";

?>

d) Division:

<?php

$a=10;

$b=2;

$c=$a/$b;

echo "$c";

?>

e) Modulus

<?php

$a=10;

$b=2;

$c=$a%$b;

echo "$c";

?>

3. Assignment Operator

This operator assigns the value at the right after handling arthmetic operation with the variable value.

a) =

This operator assigns the value at the right to variable at the left.

Eg:

<?

$ab=10

echo “the assigned value of variable ab is” .$ab;

?>

b) +=

This operator performs addition operation to the variable value at the left with the value at the right and again assigns back the result to variable at the left.

Eg.

<?

$ab=80;

$ab+=20;

echo “the total of 80 and 20 is” .$ab;

?>

**4. Increment/Decrement operator:**

The operator increase or decrease the variable value by 1 default in programming.

|  |  |  |
| --- | --- | --- |
| operator | name | function |
| ++value | pre increment | its add value in in start |
| --value | pre decrement | its subtract value in start |
| value++ | post increment | its add value in end |
| value-- | post decrement | its decrease value in end |

<?

$num=8;

$num++;

echo “the increment value is” .$num;

$num+=10;

$num - - ;

echo “<br> the decrease value” .$num;

?>

**5. Comparision Operator:**

This operator checks the equality between two operands. If it satisfies will retrive the true output else false output.In php if true 1 display and if false nothing display.

a) = = (equal to)

This operator checks equality between the two operands

note: === triple equal to check string, integer and data type.

eg.

$a=5; // here 5 is known as integer.

$a="5"; //here 5 is known as character.

Eg.

<?

$num=1;

$num1=2;

$result= ($num==$num1);

echo “the equality check is” .$result;

$num=2;

$result=($num==$num1);

echo “The equality check is” .$result;

?>

b) < (less than)

If the operand at the left is less than the operand at the right will execute the true boolean value else will retrive the false boolean value.

Eg.

<?

$num=1;

$num1=2;

$result=($num<$num1);

echo “the equality check is” .$result;

$num=5;

$result=($num<$num1);

echo “The equality check is” .$result;

?>

c) > (greater than)

If the operand at the left is the greater than the operad at the right will execute the true boolean value else will retrive the false boolean value.

Eg.

<?

$num =1;

$num1=2;

$result=($num>$num1);

echo “The equality check is”

.$result;

$num=5;

$result=($num>$num1)

echo “the equality check is” .$result;

?>

d) >= (greater than or equal to)

If the operand at the left is greater or equal to, the operand at the right will execute the true boolkean value else will retrive the false boolean value.

Eg.

<?

$num=1;

$num1=2;

$result=($num>=$num);

echo “the equality check is” .$result;

$num=5;

$result=($num>=$num1);

echo “the equality check is” .$result;

$num1=5;

$result=($num>=$num1);

echo “The equality check is” .$result;

?>

**e) <= (Less than or equal to)**

If the operand at the left is less or equal to, to the operand at the righ will execute the true boolean value else will retrive the false boolean value.

Eg.

<?

$num=1;

$num1=2;

$result=($num<=$num1);

echo “The equality check is” .$result;

$num=5;

$result=($num<=$num1);

echo “The equality check is” .$result;

$num1=5;

$result=($num<=$num1);

echo “the equality check is” .$result

?>

f) | = (not equal to)

If the operand at the left is not equal to, the operand at the right will execute the true boolen value else will retrive the false boolean value.

Eg.

<?

$num=1;

$num1=2;

$result=($num1=$num1);

echo “the equality check is” .$result;

$num=5;

$result=($num|=$num1);

echo “The equality check is” .$result;

$num1=5;

$result=($num|=$num1);

echo “ the equality check is” .$result

?>

**6. Logical Operator:**

This operator is used to combine the multiple expressions in order to generate your desired output in programming.

Types of Logical operator

|  |  |  |  |
| --- | --- | --- | --- |
| **operator** | **name** | **example** | **Result** |
| and | and | $a and $b | If $a and $b both condition is true this operator gives true result |
| or | or | $a or $b | If $a and $b any one condition is true this operator gives true result |
| xor | xor | $a xor $b | If $a and $b any one condition is true this operator gives true result but the both condition not be true |
| ! | not | !$a | If the condition is true it gives false result. |
| && | and | $a && $b | If $a and $b both condition is true this operator gives true result |
| || | or | $a || $b | If $a and $b any one condition is true this operator gives true result |

**Eg. 1**

<?php

$age=18;

if($age>17 && $age<78)

{

echo "Your age between 10 and 78";

}

?>

**7. Combined operator:** This operator are made from two different operator.

- =, \* = , / = , % =, .=

eg. $a=3;

$a-=3;

$a\*=3;

$a/=3;

$a%=3;

$a.='microsoft';

Control structure:

control structure n] program flow nfO{ control ug]{ sfd u5{ . n]lvPsf] k|f]u|fd sf]8nfO{ PlShSo'6 ug]{ jf gug]{ eGg] o;n] k/Lif0f u5{ . xfdLn] php df n]lvPsf] ;Dk"0f{ sf]8x? Pskl5 csf]{ x'Fb} PlShSo'6 x'Fb} hfG5 . t}klg slxn] sfxLF Ps cj:yfdf Ps sf]8sf] Ans / csf]{ cj:yfdf csf]{ sf]8sf] Ans PlShSo'6 ug'{kg]{ cj:yf cfpg ;S5 To:tf] cj:yfdf control structure n] k"/f ug{ ;lsG5 . cyf{t cfjZoStf cg';f/ xfdLn] sf]8sf] AnsnfO{ PlShSo'6 ug{ klg ;S5f}F jf /f]Sg klg ;S5f}F .

**Some important control structure are as follows:**

**if**

**else/else if**

**while**

**do while**

**for**

**for each**

**switch**

1. If

This statement holds only the true statement and if the condition satisfies will execute the true statement. The false statement can't display. if statements are the most common conditional constructs, and they exist in most programming languages.

**Syntax:**

if (*condition*)

*code to be executed if condition is true;*

Example 1

<?

$power=”on”;

if ($power – “on”)

{

echo “the electrical appliances can be used”;

}

?>

Example 2

<?php

$d=date("D");

if($d=="Mon")

{

echo "Have a nice weekend";

}

//if the condition is false it does not display nothing.

?>

Example 3

<?php

$minutes=61;

if($minutes>60)

{

echo "Your times is up! <br>";

echo "Leave the computer.";

}

?>

2. If ......else

<?php

$result = 70;

if ($result >= 57) {

echo "Pass <br />";

}

else {

echo "Fail <br />";

}

?>

When PHP evaluates your If...elseif...else statement it will first see if the If statement is true. If that test comes out false it will then check the first elseif statement. If that is false it will either check the next elseif statement, or if there are no more elseif statements, it will evaluate the else segment, if one exists.

This statement holds both true and false statement if the condition satisfies will execute the true statement else execute the false statement. If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax:

If(conditions)

{

statement 1;

statement 2;

........................

......................

statement n;

}

else

{

statement 1;

statement2;

................

statement n;

}

Eg.1

<?

$money=500;

if($money>=300)

{

echo “get the taxi and go home”;

}

else

{

echo “get the micro and go home”;

}

?>

<?php

$timecross=20;

if ($timecross>40)

{

$bill\_amt=1200;

$fine=100;

$total=$bill\_amt+$fine;

echo "Bill Pay Amount : $bill\_amt : Fine : $fine <br>";

echo "Total" .$total;

}

else

{

$bill\_amt=2000;

$fine=0;

$total=$bill\_amt+$fine;

echo "Bill Pay Amount : $bill\_amt : Fine : $fine <br> ";

echo "total ".$total;

}

?>

**<?php**

**$result = 70;**

**if ($result >= 75) {**

**echo "Passed: Grade A <br />";**

**}**

**elseif ($result >= 60) {**

**echo "Passed: Grade B <br />";**

**}**

**elseif ($result >= 45) {**

**echo "Passed: Grade C <br />";**

**}**

**else {**

**echo "Failed <br />";**

**}**

**?>**

**To display system date:**

<html>

<body>

<?php

$d=date("D");

echo $d

?>

</body>

</html>

Eg.2

<html>

<body>

<?php

$d=date("D");

if ($d=="Fri")

echo "Have a nice weekend!";

else

echo "Have a nice day!";

?>

</body>

</html>

**Note: date display the system day with 3 characters.**

3. if..............else if.....................else

This statement holds multiple condition of a same program which will have three and false statements: If you want to execute some code if one of several conditions are true use the elseif statement

**Syntax:**

If(conditions)

Statements;

Elese if (conditions)

{

statement;

}

else if (conditions)

{

statements;

}

else

{

statements;

}

eg.

<?

$trafficlight=”green”;

if($trafficlight= =”red”)

{

echo “stop the vehicles”;

}

else if ($trafficlight= =”yellow”)

{

echo “get ready to move or stop vehicles”;

}

else if ($trafficlight=”green”)

{

echo “move your vehicles”;

}

else

{

echo “the systems not ready”;

}

?>

questions:

Age Status

>0 - <=5 infant

>5 -<=12 child

>12 <=19 teen

>19 - <35 Youth

>35 <=55 Mature

>55 - <=100 old

<=0 ->100 Invalid age

Eg. 2

<?php

$d=date("D");

if ($d=="Fri")

echo "Have a nice weekend!";

elseif ($d=="Sun")

echo "Have a nice Sunday!";

else

echo "Have a nice day!";

?>

</body>

</html>

<?php

$result = 70;

if ($result >= 75) {

echo "Passed: Grade A <br />";

}

elseif ($result >= 60) {

echo "Passed: Grade B <br />";

}

elseif ($result >= 45) {

echo "Passed: Grade C <br />";

}

else {

echo "Failed <br />";

}

?>

**Switch Case:** It is similar to if..... else statement but it treats with value of the variable with its related statement. If you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

This is how it works:

* A single expression (most often a variable) is evaluated once
* The value of the expression is compared with the values for each case in the structure
* If there is a match, the code associated with that case is executed
* After a code is executed, **break** is used to stop the code from running into the next case
* The default statement is used if none of the cases are true

|  |
| --- |
| <html>  <body>  <?php  $num=1;  switch ($num)  {  case 1:  echo "Number 1";  break;  case 2:  echo "Number 2";  break;  case 3:  echo "Number 3";  break;  default:  echo "No number between 1 and 3";  }  ?>  </body>  </html> |

Syntax:

Switch (variable\_name)

{

case 1;

{

statement;

}

break;

case 2:

{

statements;

}

break;

case n:

{

statements;

}

break;

default:

{

statements;

}

break;

}

eg.

<?

$light="red";

switch($light)

{

case "red":

{

echo "stop vehicles";

{

break;

case "yellow":

{

echo "get ready";

}

break;

case "green";

{

echo "move vehicles";

}

break;

default:

{

echo "system failure";

}

break;

}

?>

Eg.2

<html>

<head>

<title>

switch

</title>

</head>

<body>

<?php

$day=5;

switch($day)

{

case 1:

{

echo "Today is Sunday";

break;

}

case 2:

{

echo "Today is Monday";

break;

}case 3:

{

echo "Today is Tuesday";

break;

}case 4:

{

echo "Today is Wednesday";

break;

}

case 5:

{

echo "Today is Thursday";

break;

}

case 6:

{

echo "Today is Friday";

break;

}

case 7:

{

echo "Today is Saturday";

break;

}

Default:

{

echo "Days are not longer than 7";

}

}

?>

</body>

</html>

<?php

$temperature=78;

switch($temperature)

{

case 70:

case 71:

case 72:

echo "Nice day outside";

break;

case 73:

case 74:

case 75:

echo "Ok, but a litle warm.";

break;

case 76:

case 77:

case 78:

echo "A little warmer.";

break;

default:

echo "Temperature outside the range this statement can handle.";

}

?>

**<?php**

**$myNumber = 5;**

**switch ($myNumber) {**

**case 0:**

**echo "Zero is not a valid value.";**

**break;**

**case $myNumber < 0:**

**echo "Negative numbers are not allowed.";**

**break;**

**default:**

**echo "Great! Ready to make calculations.";**

**break;**

**}**

**?>**

**<?php**

**$flower = "rose";**

**switch ($flower)**

**{**

**case "rose" :**

**echo $flower." costs $2.50";**

**break;**

**case "daisy" :**

**echo $flower." costs $1.25";**

**break;**

**case "orchild" :**

**echo $flower." costs $1.50";**

**break;**

**default :**

**echo "There is no such flower in our shop";**

**break;**

**}**

**?>**

**Loop:** It is the repetation of any statement until the condition satisfies: Very often when you write code, you want the same block of code to run a number of times.

Types of loop

1. For loop
2. While loop
3. do.... while loop
4. foreach loop. (PHP only)

**1. for loop:**

If you know the exact number of repetation than you can use for loop.

Parameters:

* **init**: Is mostly used to set a counter, but can be any code to be executed once at the beginning of the loop statement.
* **cond**: Is evaluated at beginning of each loop iteration. If the condition evaluates to TRUE, the loop continues and the code executes. If it evaluates to FALSE, the execution of the loop ends.
* **incr**: Is mostly used to increment a counter, but can be any code to be executed at the end of each loop.

Syntax:

for (initialization; condition; increment/decrement)

{

statement 1;

statement 2;

..............

statement n;

}

The for loop executes initialization before it starts; then it checks the value of condition if it;s true, the lop executes statement once. Then the loop executes increment or decrement (Which often increments the value in a loop counter variable) and after that checks the value of condition again (Which might ckeck the value in the loop counter variable). If condition is still true, the loop executes statement once again. Then the loop executes increment or decrement again and the process keeps going until condition evaluetes to false, when the loop ends. Note that condition is checked before the loop executes, and increment or decrement i sexecuted after every time the loop loops.

### Example

The following example prints the text "Hello World!" five times:

<html>

<body>

<?php

for ($i=1; $i<=5; $i++)

{

echo "Hello World!<br />";

}

?>

</body>

</html>

eg.

<?

for ($num=1; $num<=55; $num++)

{

echo $num. "<br>";

}

?>

<html>

<body>

<?php

for ($loop\_counter=0; $loop\_counter<6; $loop\_counter++)

{

echo "You're going to see this message six times. <br>";

}

?>

</body>

</html>

**Eg.2**

<?

for ($k=1; $k<=10; $k++)

{

print $k;

}

?>

**Eg. 3**

<?

for ($k=1; ;$k++)

{

if($k>10)

{

break;

}

print $k;

}

?>

**Eg. 4**

<?php

for($a=1; $a<=10; $a++)

{

echo "the square $a is" .$a\*$a ."<br>";

}

?>

**Multiple loop counters:**

**Eg.1**

<html>

<body>

<?php

for ($loop1=2, $loop2=2; $loop1<10 && $loop2< 10; $loop++, $loop2++)

{

echo "$loop1 x $loop2 = ", $loop1\*$loop2, "<br>";

}

?>

</body>

</html

Eg. 2

<html>

<body>

<?php

for ($loop1=2, $loop2=2; $loop1<10 && $loop2< 10; $loop1++, $loop2++)

{

echo "$loop1 x $loop2 = ", $loop1\*$loop2, "<br>";

}

?>

</body>

</html>

Multiplication table using loop:

<?php

echo "<h1>Multiplication table</h1>";

echo "<table border=2 width=50%";

for ($i = 1; $i <= 9; $i++ ) { //this is the outer loop

echo "<tr>";

echo "<td>".$i."</td>";

for ( $j = 2; $j <= 9; $j++ ) { // inner loop

echo "<td>".$i \* $j."</td>";

}

echo "</tr>";

}

echo "</table>";

?>

At last let's consider the example that uses 2 variables. One to add all the numbers from 1 to 10. The other to add only the even numbers.

<?php

$total = 0;

$even = 0;

for ( $x = 1, $y = 1; $x <= 10; $x++, $y++ ) {

if ( ( $y % 2 ) == 0 ) {

$even = $even + $y;

}

$total = $total + $x;

}

echo "The total sum: ".$total."<br />";

echo "The sum of even values: ".$even;

?>

1. **While loop**: Whle loop tells PHP to execute the nesteted statements repeatedly, as long as the while expression evaluates to TRUE. The value of the expression checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statement(s), execution will not stop until the end of the iteration (each time PHP runs the statements in the loop is one iteration) Sometimes, if the while expression evaluates to FALSE from the very beginning, the nested statements won't even be run once. If you don't know the exact repeation of statement you can use while loop.

Syntax:

While (condition)

{

statement 1;

statement 2;

..................

statement n;

increment/decrement;

}

eg. 1

<?

$ab=2;

while ($ab<=25)

{

echo $ab. "<br>";

$ab+=2;

}

?>

eg. 2

<?

$k=1;

while($i<=10)

{

print $i++;

}

?>

eg. 3

<?

$k=1;

while($k<=10):

print $k;

$k++;

endwhile;

?>

3. **do...... while loop:** This loop is similar to the while loop. But the difference is that in while loop first condition is checked then only treats with statement. In do.... while loop first treats with the statement then only checks the condition. The do...while loop is similar to the previous while loop, except that the truth expression is checked at the end of each iteration instead of at the beginning. **This means that the loop always runs at least once**.

Syntax:

do

{

statement 1;

statement 2;

..................

statement n;

increment/decrement;

}

while (condition);

eg.

<?

$aa=0;

do

{

echo $aa. "<br>";

$aa++;

}

while ($aa<=15);

?>

**Example:**

<?php

$a=5;

$b=1;

$c=$a\*$b;

do

{

echo $a."x".$b."=".$c."<br>";

$b++;

}

while($b<=20);

?>

Here, eventhough its condition evalueates to false, you’ll see that the do...while loop executes its body once but the while loop does not: The do..... while loop executed once but the while loop didn’t.

<html>

<body>

<?php

$variable=20;

do

{

echo "The do... while loop says \$variable= ", $variable, "<br>";

}

while ($variable<10);

while ($variable<10)

{

echo "The while loop says \$variable = ", $variable, "<br>";

}

?>

</body>

</html>

## The foreach Statement

The foreach statement is used to loop through arrays.

For every loop, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop, you'll be looking at the next element.

### Syntax

|  |
| --- |
| foreach (*array* as *value*)  {  *code to be executed;*  } |

### Example

The following example demonstrates a loop that will print the values of the given array:

|  |
| --- |
| <html>  <body>  <?php  $arr=array("one", "two", "three");  foreach ($arr as $value)  {  echo "Value: " . $value . "<br />";  }  ?>  </body>  </html> |

**Eg.2**

<html>

<body>

<?php

$arr=array("turkey", "ham", "beef");

foreach ($arr as $value)

{

echo "Current sandwich: $value<br>";

}

?>

</body>

</html>

**Loop Control: break and continue**

**break*;***

**continue*;***

Sometimes, you want to terminate the execution of a loop in the middle of an iteration. For this purpose, PHP provides the break statement. If break appears alone, as in break; the innermost loop is stopped. break accepts an optional argument of the amount of nesting levels to break out of, break *n*; which will break from the n innermost loops (break 1; is identical to break;). n can be any valid expression. In other cases, you may want to stop the execution of a specific loop iteration and begin executing the next one. Complimentary to break, continue provides this functionality. continue alone stops the execution of the innermost loop iteration and continues executing the next iteration of that loop.

Continue n can be used to stop execution of the n innermost loop iterations. PHP goes on

executing the next iteration of the outermost loop. As the switch statement also supports break, it is counted as a loop when you want to break out of a series of loops with break n.

<?php

$i=0;

while($i<10)

{

if($i==5)

{

break;

}

echo $i;

$i++;

}

?>

**EG. 2**

<?php

for($loop\_counter=0; $loop\_counter<1000; $loop\_counter++)

{

echo ("I'm going to do this a thousand times unless you stop me! <br>");

if($loop\_counter==5)

{

echo "Alright, I'm quitting.<br>";

break;

}

}

?>

**Continue:**

<?php

$number=1;

for($number=-2; $number<3; $number++)

{

if($number==0)

{

continue;

}

echo "1/$number=", 1/$number, "<br>";

$number++;

}

?>

**PHP Alternate Syntax:**

PHP also supports an alternative syntax for if, while, for, foreach, and switch. In each case, the form of the alternate syntax changes the opening curly brace to a colon (:) and the closing brace to endif;, endwhile;, endfor;, endforeach;, or endswitch;, respectively.

Eg. of endif

<?php

$temperature=61;

if($temperature==60):

echo "The temperature is 60";

elseif ($temperature==70):

echo "The temperature is 70";

else:

echo "The temperature is not 60 or 70";

endif;

?>

**Eg. of endfor**

<?php

for($loop\_counter=0; $loop\_counter<6; $loop\_counter++):

echo "You're going to see this message six times. <br>";

endfor;

?>

Eg. of endswitch:

<?php

$temperature=65;

switch ($temperature):

case 60:

echo "The temperature is 60.";

break;

case 70:

echo "The temperature is 70.";

break;

case 80:

echo "The temperature is 80.";

break;

default:

echo "The temperature is not 60,70, or 80.";

endswitch;

?>

# String Functions in PHP

PHP String function helps to work with string. PHP is string oriented, and it comes packed with many string functions. PHP programs are often text intensive the data you get from the user is in text forml for example, and you might want to strip exra spaces from the begining and end of such test. or you might want to search that text ofor command words that the user can use. Or you might want to capitalize that text, or any of a hundred other things. You can perform all those operations with the PHP string functions.

**Some important php string function are given there.**

**1. strlen():** Returns the length of variable

Eg.

<?

$name="krishna";

$len=strlen($name);

print $len;

?>

**2. strtoupper(): Conver the upper case**

<?php

echo "In upper case:", strtoupper("Microsoft"), "<br>";

?>

**3. strrev: It reverse the character**

<?php

echo "Reversed: ", strrev("MICROSOFT"), "<br>";

?>

**4. strtolower: it changes the lower cash**

<?php

echo "in lower case:", strtolower("MICROSOFT"), "<br>";

?>

**<?php**

$name='MICROSOFT EDUCATIONAL INSTITUTE';

$change\_case=strtlower($name);

echo $change\_case;

?>

**5. ucfirst: first letter capital letter of each word.**

**<?php**

**echo "First letter capital:", ucfirst("Microsoft institute. bagbazar"), ?>**

**<?php**

**$name='microsoft educational institute';**

**$change\_case=ucfirst($name);**

**echo $change\_case;**

**?>**

**6. chr: the chr functions converts ASCII numeric codes into ther corresponding characters.**

**<?php**

**echo "using ascii codes:", chr(65), chr(66), chr(67), "<br>";**

**?>**

**7. strpos: It search the position:**

**<?php**

**echo "The word 'problem' is at position", strpos("there is No problem",**

**"problem"), "<br>";**

**// ans 12**

**?>**

**8. substr: It extract substrings**

**<?php**

**echo "the substring is ", substr("No problem", 1,8), "<br>";**

**?>**

**9. substr\_replace: You can replace substrings:**

**<?php**

**echo "Replacing 'problem' with 'problems' gives:", substr\_replace("No problem", "problems", 3, 8), "<br>";**

**?>**

**10. str\_replace():**

**<?php**

**$name='I like pop';**

**$string=str\_replace('pop','sentimental',$name);**

**echo $string;**

**?>**

# PHP Arrays

An array can store one or more values in a single variable name.

**What is an array?**

When working with PHP, sooner or later, you might want to create many similar variables.Instead of having many similar variables, you can store the data as elements in an array.Each element in the array has its own ID so that it can be easily accessed. Arrays are collection of data items stored under a single name. For eg. If you are going to display a many students name. You might create an array named $student and assign the name “ “ fo the first place in that array, emement 0:

There are three different kind of arrays:

1. **Indexed / Numeric array** - An array with a numeric ID key
2. **Associative array** - An array where each ID key is associated with a value
3. **Multidimensional array** - An array containing one or more arrays

**PHP Indexed Arrays**

**There are two ways to create indexed arrays:**

**Example: 1**

The index can be assigned automatically (index always starts at 0):

$tv=array("sony","lg","samsung");

**Example: 2**

**The index can be assigned manually:**

$tv[0]="sony";

$tv[1]="lg";

$tv[2]="samsung";

**The following example creates an indexed array named $tv, assigns three elements to it, and then prints a text containing the array values:**

**Example**

<?php

$tv=array("sony","lg","samsung");

echo "I like " . $tv[0] . ", " . $tv[1] . " and " . $tv[2] . ".";

?>

**Example: 3**

**// we can start from any number like 5.**

**<?php**

**$arr=array();**

**$arr[5]="Microsoft";**

**$arr[6]="Institute";**

**echo $arr[5]."<br>";**

**echo $arr[6]."<br>";**

**?>**

**Example 4: array using while loop**

<?php

$arr=array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday");

$i=0;

while($i<6)

{

echo $arr[$i]."<br>";

$i++;

}

?>

**Example 5: array using for loop**

<?php

$arr=array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday");

for($i=0; $i<6; $i++)

{

echo $arr[$i]."<br>";

}

?>

**Associative Arrays**

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 array you use a string name in index number to make the programming easier to handle the multiple datas.

Syntax:

Arrayname="array(key=>value,key=>value.....)

Example 1

<?php

$students=array('krishna'=>98,'ramita'=>80,'kritagya'=>88);

print\_r($students); //print\_r is use to display array

echo "<br>";

echo $students['krishna']; //here krishna is key and it display the value 98

?>

fore each loop in array

<?php

$name = array('ram','shyam','hari');

foreach ($name as $store) { // array values are save in store variable

//$name and $store is a variable name.

echo $store.'<br>';

}

echo "Assosiative array with foreach";

echo "<br>";

$arrList = array(1=>'ram',2=>'shyam',3=>'hari');

foreach ($arrList as $k=>$value) {

echo $k . ": " . $value; echo "<br>";

}

**?>**

**Exampl e 2:**

**<?php**

**$students=array('krishna'=>98,'ramita'=>70,'kritagya'=>85);**

**foreach($students as $name=>$value)**

**{**

**echo $name. " has secured".$value."marks<br>";**

**echo $name."<br>"; // to display name only**

**echo $value."<br>"; // to display value only.**

**}**

**?>**

**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.

**Example**

<?php

$students=array('kritagya'=>array('maru','10','class4'),

'krishna'=>array('kathmandu','38','master'));

print\_r ($students);

echo "<br>";

echo $students['krishna']['2'];

?>

**Extracting Data from Arrays:**

You can use the extract function to extract data from arrays and store it in variables.

You can use extract to create variables whose names will be taken from the keys in the array, and those variables will be assigned the values in the array.

<?php

$icecream["good"]="orange";

$icecream["better"]="Vanilla";

$icecream["best"]="rum raisin";

extract($icecream);

echo $better;

?>

**Sorting Arrays:** Sort function sorts arrays in ascending or descending array.

ascending order: sort

desceending order : rsort

eg.:

<?php

$arrList = array('ram','anil','tiger','chandra');

// sort($collection);

sort($arrList);

print\_r($arrList);

?>

Split Array: We can split array using array\_slice function.

<?php

$icecream[0]="orange";

$icecream[1]="vanilla";

$icecream[2]="rum raisin";

$icecream[3]="lime";

$subarray=array\_slice($icecream, 1,2);

foreach($subarray as $value)

{

echo "$value <br>";

}

?>

merge array: We can merge array using array\_merge function.

<?php

$icecream[0]="orange";

$icecream[1]="vanilla";

$icecream[2]="rum raisin";

$icecream[3]="lime";

$cold\_drinks[0]="Coke";

$cold\_drinks[1]="Fanta";

$cold\_drinks[2]="Sprite";

$cold\_drinks[3]="Soda";

$subarray=array\_merge($icecream, $cold\_drinks);

foreach($subarray as $value)

{

echo "$value <br>";

}

?>

sum array: The array\_sum total the numeric elements in an array.

<?php

$scores=array(10, 20, 30, 40, 50);

echo "the total is ", array\_sum($scores);

echo "the average is", array\_sum($scores)/count($scores);

?>

Length of Array => array.name.length()

Array is similar to the variable but in variable you can store only a single value at time, in array you can store multiple number of value of a time and can be eecuted all the value or a single value. Array collect values into list. You refer to an element in an array using an index, whic often an integer but can also be a string. And the value of the element can be text, a number or event another array.

**EG. 1**

<?

$Employee\_Name[0]="Pramila Nepali";

$Employee\_Name[1]="Alisha Nakarmi";

$Employee\_Name[2]="Shyam Karki";

print("$Employee\_Name[2]");

?>

**EG. 2**

<?

$Employee\_Name[0]="Pramila Nepali";

$Employee\_Name[1]="Alisha Nakarmi";

$Employee\_Name[2]="Shyam Karki";

$indexlimit=count($Employee\_Name);

for ($index=0; $index<$indexlimit; $index++)

{

print("Name of Employee is $Employee\_Name[$index].<br>\n"); // \n is line break

}

?>

**EG. 3**

<?

$monthname=array(1=>"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December");

print ("Month 5 is $monthname[5] <br>\n");

?>

<?php

$month\_array = array( "January", "February", "March", "April", "May", "June",

"July", "August", "September", "October", "November", "December");

echo "<select name=\"day\">";

$i = 1;

while ( $i <= 31 ) {

echo "<option value=".$i.">".$i."</option>";

$i++;

}

echo "</select>";

echo "<select name=\"month\">";

$i = 0;

while ( $i <= 11 ) {

echo "<option value=".$i.">".$month\_array[$i]."</option>";

$i++;

}

echo "</select>";

echo "<select name=\"year\">";

$i = 1900;

while ( $i <= 2015 ) {

echo "<option value=".$i.">".$i."</option>";

$i++;

}

echo "</select>";

?>

**Some Built-in functions about Array in PHP**

**1. Count:** Counts the number of elements in array.

<?

$sample=array(1,2,3,4,);

$num=count($sample);

echo "the count is" .$num;

?>

2. Sizeof: The sizeof function counts the number of elements in an array.

<?

$sample=array(1,2,3,4,5,6);

$num=sizeof($sample);

echo "the size of array is" .$num;

?>

**Deleting Array Elements:**

set an array element to an empty string: or use the **unset** function.

eg.

<?php

$student[0]="Ramesh";

$student[1]="Prakash";

$student[2]="Raju";

$student [1]= “ "; or unset($student[1]);

echo "students", $student[0], "<br>";

echo "students", $student[1], "<br>";

echo "students", $student[2], "<br>";

?>

**The print\_r Function**

There’s a simple function for displaying the contents of an array the print\_r function:

<?php

$student[0]="Ramesh";

$student[1]="Prakash";

$student[2]="Raju";

print\_r($student);

?>

Implode and Explode: Converting between Strings and Array:

The implode function implodes an array into a string, and the explode function explode a string into an array. That’s useful if you’ve stored your data in strings in files and want to convert those strings into array elements when your web application runs.

Implode:

<?php

$icecream[0]="chocolate";

$icecream[1]="pecan";

$icecream[2]="strawberry";

$text= implode(",", $icecream);

echo $text;

?>

Explode:

<?php

$text="chocolate, pecan, strawberry";

$icecream=explode(",", $text);

print\_r($icecream);

?>

**PHP Functions**

The real power of PHP comes from its functions.In PHP - there are more than 1000 built-in functions available.A function is a block of code that can be executed whenever we need it. A Function is a reusable piece of code. You will write it once and you can use it many times. For example, We can make a function to perform addition or subtraction or calculation of VAT, And we can use it as many times as we want.

We can divide function in two groups.

a) Built In function: It is php built in function.

b) User define function: we create a function by giving name.  
  
**Creating PHP functions:**

All functions start with the word "function()"

Name the function - It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)

Add a "{"  - The function code starts after the opening curly brace Insert the function code

Add a "}"  - The function is finished by a closing curly brace

Syntax:

<?php

Function function\_name()

{

Type function code blocks

}

Function calling statement

?>

Example I: A simple function that writes my name when it is called:

<?php

function mei () // no argument in parenthesis ().

{

echo "Microsoft Educational Institute";

}

mei();

?>

# Note: mei is function name, mei(); calling the function

<html>

<body>

<?php

function calculation(){

$time=5;

$rate=.8;

$principal=1000;

echo "The interest is ".$time\*$rate\*$principal;

}

calculation(); // function calling

?>

</body>

</html>

Example:

<?php

function address()

{

echo "Microsoft Educational Institute <br>";

echo "Bagbazar <br>";

echo "Kathmandu Nepal <br>";

echo "4231108 <br>";

}

echo "The address is";

address();

address()// we can call many times

?>

**PHP Functions - Adding parameters**

Our first function (writeMyName()) is a very simple function. It only writes a static string.To add more functionality to a function, we can add parameters. A parameter is just like a variable.

You may have noticed the parentheses after the function name, like: writeMyName(). The parameters are specified inside the parentheses.

**Single parameter function:**

**<?php**

**function multiply($number)**

**{**

**echo $number\*5;**

**}**

**multiply(4);**

**?>**

**Example 1**

<html>

<head>

<title>Writing PHP Function with Parameters</title>

</head>

<body>

<?php

function addFunction($num1, $num2)

{

$sum = $num1 + $num2;

echo "Sum of the two numbers is : $sum";

}

addFunction(10, 20);

?>

</body>

</html>

Examples 2:

The following example will write different first names, but the same last name:

The output of the code above will be:

<html>

<body>

<?php

function writeMyName($fname)

{

echo $fname . " Maharjan"."<br>";

}

echo "My name is ";

writeMyName("Krishna");

echo "My name is ";

writeMyName("kritagya");

echo "My name is ";

writeMyName("ramita singh");

?>

</body>

</html>

**Example 2**

<?php

//function declaration (the function asks for 2 parameters)

function displayfullname($first, $last)

{

echo "<p>my first name is <strong>$first</strong>" .

" and my last name is <strong>$last</strong></p>";

}

//call the function parsing two strings

displayfullname("Krishna", "Maharjan"); //my first name is John and my last name is Doe

?>

**Example 3**

The following function has two parameters:

|  |
| --- |
| <html>  <body>  <?php  function writeMyName($fname,$punctuation)  {  echo $fname . " Maharjan" . $punctuation . "<br />";  }  echo "My name is ";  writeMyName("Krishna",".");  echo "My name is ";  writeMyName("Ramita","!");  echo "My name is ";  writeMyName("Kritagya","...");  ?>  </body>  </html> |

**Example: 4**

Here other samples of user defined functions:

**Function Return Value:**

**Function nfO{ call ubf{ pSt function n] lbg] value nfO{ g} function sf] return value elgG5 . function n] return ug]{ value agfpg return keyword sf] k|of]u ug'{k5{ .**

**add()**

<?php //functions.php

//function declaration

function add($a, $b)

{

//declaring a variable to hold result of the addition

$result;

//calculating...

$result = $a + $b;

//return the result

return $result;

}

//function call

echo "<p>" . add(1, 2) . "</p>"; //3

//another function call

echo "<p>" . add(10, 20) . "</p>"; //30

?>

**Example 5 Subtraction**

<?php //functions.php

//function declaration

function subtract($a, $b)

{

//declaring a variable to hold result of the subtraction

$result;

//calculating...

$result = $a - $b;

//return the result

return $result;

}

//function call

echo "<p>" . subtract(4, 2) . "</p>"; //2

//another function call

echo "<p>" . subtract(100, 20) . "</p>"; //80

?>

Example 6: vat()

<?php //functions.php

//function declaration

function vat($price)

{

//declaring a variable to hold result of the addition

$result;

//calculating...

$result = ($price \* 20) / 100;

//echoing the result

echo "The VAT of $price is: $result";

}

**function with defining variable outside**

<?php

$a=20;

$b=50;

$c=70;

function cal($num1, $num2, $num3)

{

$sum=$num1+$num2+$num3;

$multiply=$num1\*$num2\*$num3;

echo "Sum".$sum."<br>";

echo "Multiplication qube".$multiply."<br>";

}

cal($a,$b,$c);

?>

function with table data

<?php

$a="s.no";

$b="name";

$c="address";

function table($num1,$num2)

{

echo "<table border=1><tr><td>".$num1."<td>".$num2;

}

table($a,$b);

?>

//function call

echo "<p>" . vat(100) . "</p>";

?>

**Working with date function:**

<?php //functions.php

/\*

Date() is a built-in php function for showing datas.

we can pass it many different values/arguments.

Here some examples:

\*/

echo "<p>". date("D") ."</p>"; //Day name in three letters => Tue

echo "<p>". date("M") ."</p>"; //Month name in three letters => Dec

echo "<p>". date('l, jS F Y')."</p>"; //Day, date Month Year => Monday, 23 March 1920

?>

## PHP Form Handling

Website or web application agfpFbf olb visitor cyjf user af6 s]xL data jf hfgsf/L lng'kof]{ eg] HTML Form sf] k|of]u ul/G5 . HTML Form df el/Psf data nfO{ email or database df store ug{ PHP Code af6 ;lhn} ug{ ;lsG5 . HTML Form nfO{ e/]/ Submit ubf{ PHP sf] Global array variable x? $\_POST or $\_GET or $\_REQUEST df data x? cfkm} store x'G5g\ .

**Attributes of form tag:** Action, method, enctype:

Action: file name eg. Process.php

Method: Get, post

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will **automatically** be available to your PHP scripts.

## The $\_POST Variable

The $\_POST variable is an array of variable names and values sent by the HTTP POST method.

The $\_POST variable is used to collect values from a form with 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.

## The $\_GET Variable

The $\_GET variable is an array of variable names and values sent by the HTTP GET method.

The $\_GET variable is used to collect values from a form with method="get". Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and it has limits on the amount of information to send (max. 100 characters).

**Form example: form.html**

<html>

<body>

<form action="welcome.php" method="post">

Name: <input type="text" name="name" />

Age: <input type="text" name="age" />

<input type="submit" />

</form>

</body>

</html>

The example HTML page above contains two input fields and a submit button. When the user fills in this form and click on the submit button, the form data is sent to the "welcome.php" file.

The "welcome.php" file looks like this:

|  |
| --- |
| <html>  <body>  Welcome <?php echo $\_POST["name"]; ?>.<br />  You are <?php echo $\_POST["age"]; ?> years old.  </body>  </html> |

Data submit ul/;s]kl5 tL data x?nfO{ lng $\_POST[‘name’] k|of]u ug{ ;lsG5 . olb method df get eP $\_GET k|of]u ul/G5 cyjf s'g} klg gePsf] v08df $\_REQUEST k|of]u ul/G5 .

**Connection with form (create 2 file, form.html & result.php)**

**File name: form.html**

<html>

<head>

<title>Background</title>

</head>

<body>

<form method ="post" name = "form" action = "result.php">

<input type ="text" name ="num" "size=10" maxlength="10"> Type Number

<input type ="submit" value ="Get result">

</form>

</body>

</html>

**File Name: Resut.php**

<?php

$num=$\_POST['num'];

for ($i= 1; $i<=20; $i++)

{

$result=$num\*$i;

echo $num."\*".$i."=".$result."<br>";

}

echo "<a href ='form.html'> another number </a>";

?>

Examples:

File 1: welcomes.php

<html>

<body>

Welcome <?php echo $\_POST["name"]; ?><br>

Your email address is: <?php echo $\_POST["email"]; ?>

</body>

</html>

File 2: login.php

<html>

<body>

<form action="welcomes.php" method="post">

Name: <input type="text" name="name"><br>

E-mail: <input type="text" name="email"><br>

<input type="submit">

</form>

</body>

</html>

# PHP $\_GET

The $\_GET variable is used to collect values from a form with method="get".

### Example

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

When the user clicks the "Submit" button, the URL sent could look something like this:

The "welcome.php" file can now use the $\_GET variable to catch the form data (notice that the names of the form fields will automatically be the ID keys in the $\_GET array):

|  |
| --- |
| Welcome <?php echo $\_GET["name"]; ?>.<br />  You are <?php echo $\_GET["age"]; ?> years old! |

## Why use $\_GET?

**Note:** When using the $\_GET variable all variable names and values are displayed in the URL. So this method should not be used when sending passwords or other sensitive information! However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

**Note:** The HTTP GET method is not suitable on large variable values; the value cannot exceed 100 characters.

# PHP $\_POST

The $\_POST variable is used to collect values from a form with method="post".

### Example of post:

|  |
| --- |
| <form action="welcome.php" method="post">  Enter your name: <input type="text" name="name" />  Enter your age: <input type="text" name="age" />  <input type="submit" />  </form> |

When the user clicks the "Submit" button, the URL will not contain any form data, and will look something like this:

The "welcome.php" file can now use the $\_POST variable to catch the form data (notice that the names of the form fields will automatically be the ID keys in the $\_POST array):

|  |
| --- |
| Welcome <?php echo $\_POST["name"]; ?>.<br />  You are <?php echo $\_POST["age"]; ?> years old! |

## Why use $\_POST?

* Variables sent with HTTP POST are not shown in the URL
* Variables have no length limit

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

1. isset( ) :- Determine whether a variable is set. Returns TRUE if var exists; FALSE otherwise.

If a variable has unset with **unset( )**, it will no longer be **isset( ). Isset ( )** will return **FALSE** if testing a variable that has been set to **NULL**. Also note that a **NULL** ("\0") is not equivalent to the PHP **NULL** constant.

Warning : isset( ) only works with variables as passing anything else will result in a parse error. For checking if constants are use the defined ( ) function

# Isset:

## Add using isset function

<form method=post action=#>

Number 1 <input type=text name=n1>

Number 2 <input type=text name=n2>

<input type=submit value=add name=add>

<input type=submit value=sub name=sub>

</form>

<?php

if(isset($\_POST['add']))

{ //if click on add button the first block code execute.

$a=$\_POST['n1']+$\_POST['n2'];

echo $a;

}

if(isset($\_POST['sub']))

{

$a=$\_POST['n1']-$\_POST['n2'];

echo $a;

}

?>

**The $\_REQUEST variable**

The PHP $\_REQUEST variable contains the contents of both $\_GET, $\_POST, and $\_COOKIE.

The PHP $\_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

<?php

if( $\_REQUEST["name"] || $\_REQUEST["age"] )

{

echo "Welcome ". $\_REQUEST['name']. "<br />";

echo "You are ". $\_REQUEST['age']. " years old.";

exit();

}

?>

<html>

<body>

<form action="<?php $\_PHP\_SELF ?>" method="POST">

Name: <input type="text" name="name" />

Age: <input type="text" name="age" />

<input type="submit" />

</form>

</body>

</html>

Here $\_PHP\_SELF variable contains the name of self script in which it is being called.

**Using “isset”**

**You can use the “isset” function on any variable to determine if it has been set or not. You can use this function on the $\_POST array to determine if the variable was posted or not. This is often applied to the submit button value, but can be applied to any variable.**

**For example:<?php**

**if(isset($\_POST['submit'])**

**{**

**echo("First name: " . $\_POST['firstname'] . "<br />\n");**

**echo("Last name: " . $\_POST['lastname'] . "<br />\n");**

**}**

**?>**

**<form action="myform5.php" method="post">**

**<p>First name: <input type="text" name="firstname" /></p>**

**<p>Last name: <input type="text" name="firstname" /></p>**

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

**</form>**

**The above code will only display the submitted values if the submit button was clicked.**

**include**: The include statement includes and evaluates the specified file.

<?

echo "A $color $fruit"; include 'vars.php';

echo "$color $fruit";

?>

**6. Handling Password Control**

**A common use of PHP is to check passwords on the server, giving the user access to a resource if they have the right password, and you can use password controls for that.**

**Phppassword.html**

<html>

<head>

<title>Untitled Document</title>

</head>

<body>

Entering data with Password

<form method="post" action="phppassword.php">

Enter your Password:

<input name="password" type="password" />

<br />

<br />

<input type="submit" value="send"

</form>

</body>

</html>

**Phppassword.php**

<html>

<head>

<title>Untitled Document</title>

</head>

<body>

<h1> Reading data from Password

</h1>

<?php

if($\_REQUEST["password"]=="admin"){

echo "Password accepted ";

}

?>

</body>

</html>

**Note: If you type wrong password nothing display on browser.**

**Phppassword2.php**

<html>

<head>

<title>Untitled Document</title>

</head>

<body>

<h1> Reading data from Password

</h1>

<?php

if($\_REQUEST["password"]=="admin"){

?>

<h2>

Password Accepted</h2>

Welcome

<?php

}

else {

?>

<h2> Password denied </h2>

You did not enter the correct password. <br />

<?php

}

?>

</body>

</html>

FILE UPLOAD:

File.html

<form action="process2.php" method="post" enctype="multipart/form-data">

<label for="file"> upload file: </label>

<input type="file" name="file">

<input type="submit">

</form>

Dflysf] form df form element input sf] k|sf/ form sf] attribute enctype 5 olb of] 5}g eg] file upload x'Fb}g . ha file sumbit u5f}+ kmfOnsf] ;a} data php global variable $\_FILES df store x'G5 .

Process2.php

<?php

if($\_FILES["file"]["error"]>0)

{

echo "Error: " . $\_FILES["file"] ["error"] . "<br>";

}

else

{

echo "upload" . $\_FILES["file"] ["name"]. "<br>"; //file name

echo "Type" . $\_FILES["file"]["type"]. "<br>"; // file type

echo "Size". ($\_FILES["file"]["size"] / 1024)."KB<br>"; // file size

echo "stored in:". $\_FILES["file"]["temp\_name"]; // temporary directory file name

}

?>

All the uploaded data uploaded into web server temporary directory we can move the temporary directory from required directory using PHP code.

<?php

if($\_FILES["file"] ["error"] >0)

{

echo "Error: " . $\_FILES["file"] ["error"] . "<br>";

}

else

{

$temporary\_location=$\_FILES["file"] ["temp\_name"] ;

$destination\_location="uploaded";

$file\_name=$\_FILES ["file"] ["name"];

move\_uploaded\_file($temporary\_location, $destination\_location . '/' . $file\_name);

echo "stored in: " . $destination\_location . '.' . $file\_name;

}

?>

File ;fg{ PHP sf] move\_uploaded\_file() function sf] k|of]u u5f}+ h;n] b'Oj6f parameter lnG5 Pp6f source directory / csf]{ destination directory.