



# PHP

PHP (Hypertext Preprocessor /Personal Home Page)

# Contents

- PHP: PHP Fundamentals
- Data types
- Operators
- Control structures,
- PHP Arrays
- PHP String Handling
- PHP Functions
- File Handling
- Super global variables

# PHP

- PHP is a server scripting language, and a powerful tool for **making dynamic and interactive Web pages**.
- PHP scripts are executed on the **server**.
- PHP is a widely-used, free, and efficient alternative to competitors such as **Microsoft's ASP/JSP/Servlet**
- PHP is an acronym for **"PHP: Hypertext Preprocessor"**
- PHP is **Open source and Platform independent**.
- Example: Wordpress, facebook
- Originally developed by **Rasmus Lerdorf** in 1994

# What PHP can do?

- PHP files can contain **text, Images,HTML,CSS, JavaScript, and PHP code**
- PHP code are executed on the server, and the **response is returned to the browser as plain HTML**
- PHP files have extension **".php"**
- PHP **can create, open, read, write, delete, and close files on the server**
- PHP can **collect form data**
- PHP **can send and receive cookies**
- PHP can **add, delete, modify data in your database**
- PHP can **encrypt and decrypt data**

# PHP

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases MySQL, Oracle, PostgreSQL, Informix etc.
- PHP is open source software and free
- A PHP file normally contains HTML tags, and some PHP scripting code.

# Server side script Languages

- ASP
- JSP
- Pascal
- PHP
- Python
- R
- Ruby
- Groovy Server pages etc...

# Advantages of PHP language

- **Open source:** It is developed and maintained by a large group of PHP developers, this will help in creating a support community, abundant extension library.
- **Platform independent:** PHP is platform independent and can be run on all major operating systems.
- **Speed:** Faster than other scripting languages, for example, ASP and JSP.
- **Easy to use:** It uses C like syntax, so for those who are familiar with C, it's very easy for them to pick up and it is very easy to create website scripts.
- **Stable:** Since it is maintained by many developers, so when bugs are found, it can be quickly fixed.

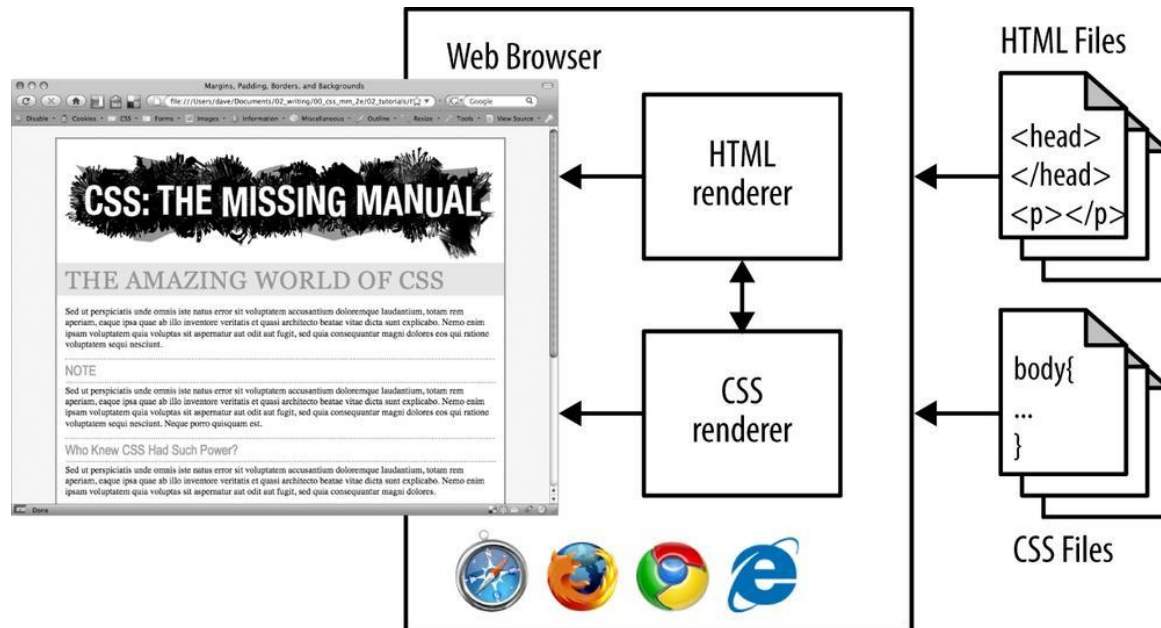
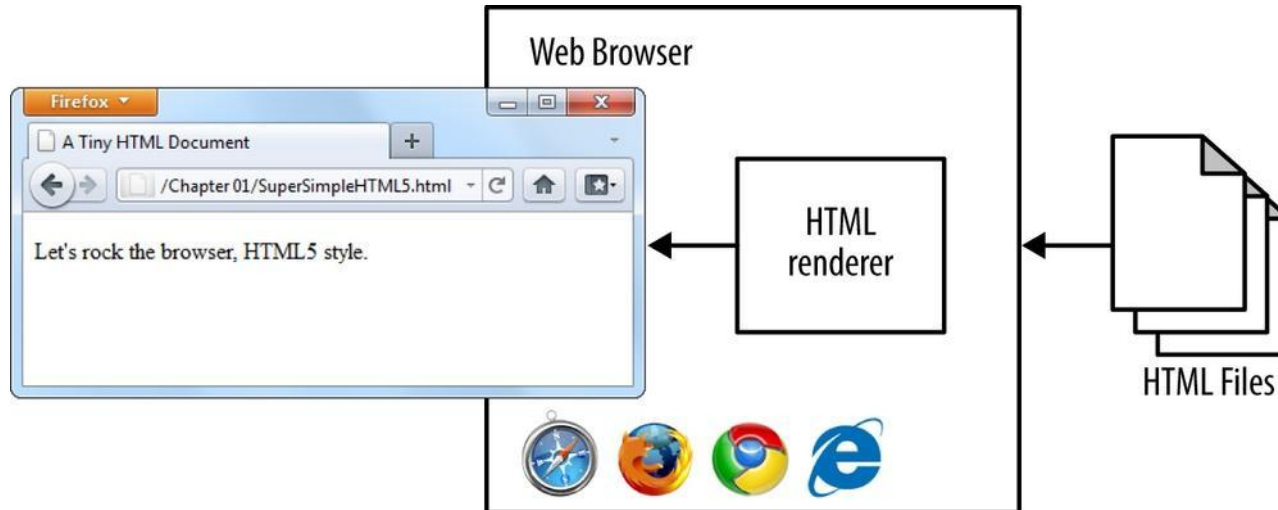
# Advantages of PHP language Cont...

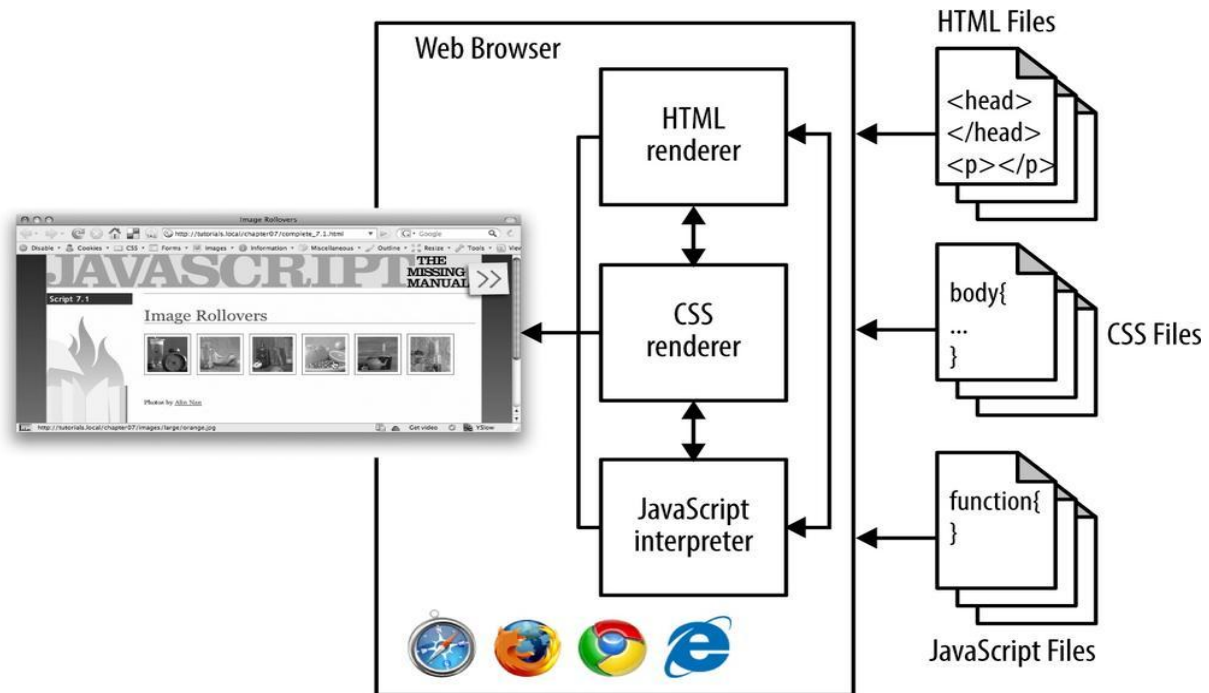
- **Powerful library support:** You can easily find functional modules you need such as PDF, Graph etc.
- **Built-in database connection modules:** You can connect to database easily using PHP, since many websites are data/content driven, so we will use database frequently, this will largely reduce the development time of web apps.
- **Compatible with almost all servers:**

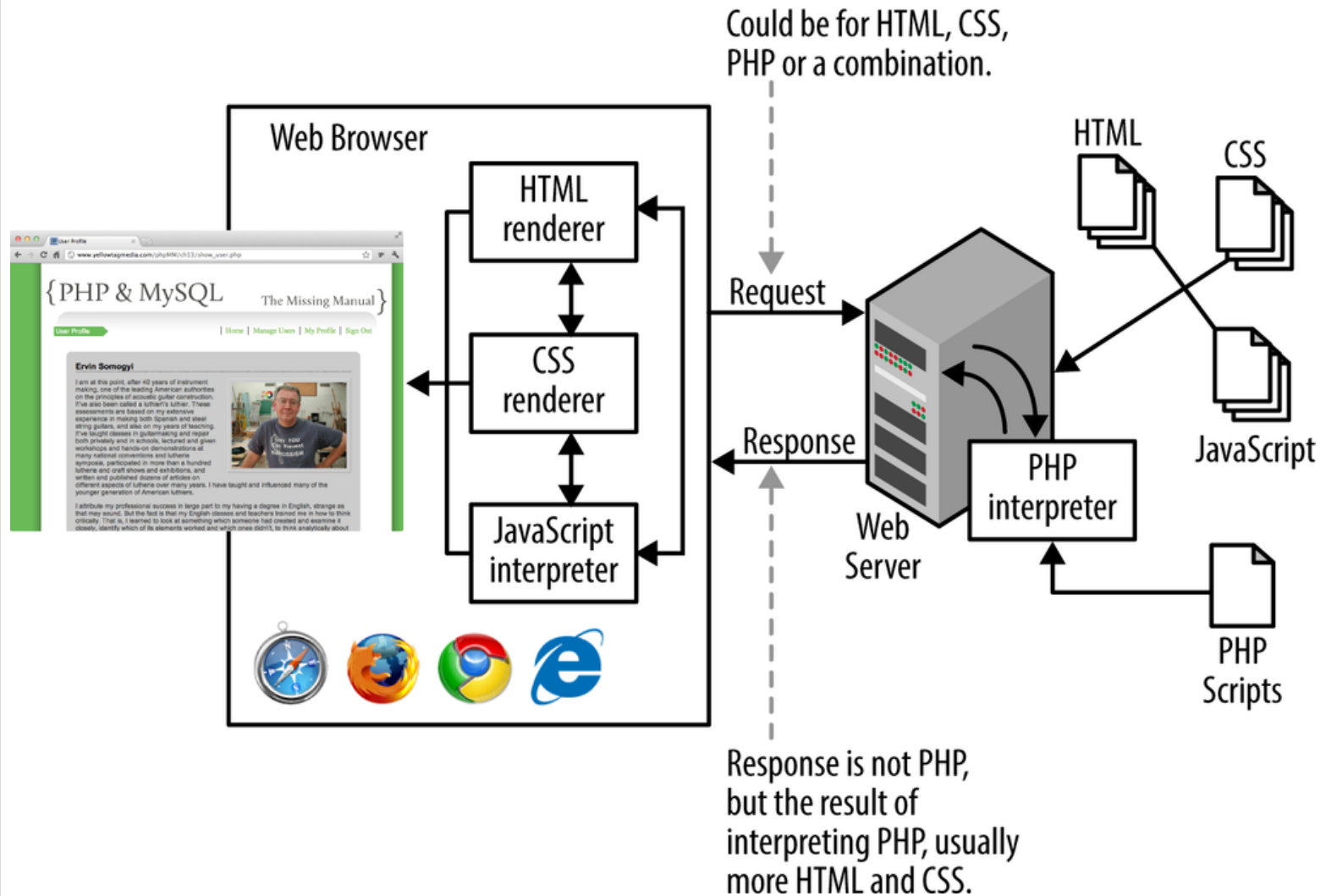


# Disadvantages of PHP

- Security : Since it is open sourced, so all people can see the source code, if there are bugs in the source code, it can be used by people to explore the weakness of PHP
- Not suitable for large applications: Hard to maintain since it is not very modular.
- Weak type: Implicit conversion may surprise unwary programmers and lead to unexpected bugs. For example, the strings “1000” and “1e3” compare equal because they are implicitly cast to floating point numbers.







# Basic PHP Syntax

```
<?php  
    // PHP code goes here  
?>
```

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**
- HTML script tags –  
`<script language = "PHP">...</script>`

# PHP Syntax

```
<!DOCTYPE html>  
<html>  
<body>  
<h1>My first PHP page</h1>  
<?php  
echo "Hello World!";  
?>  
</body>  
</html>
```

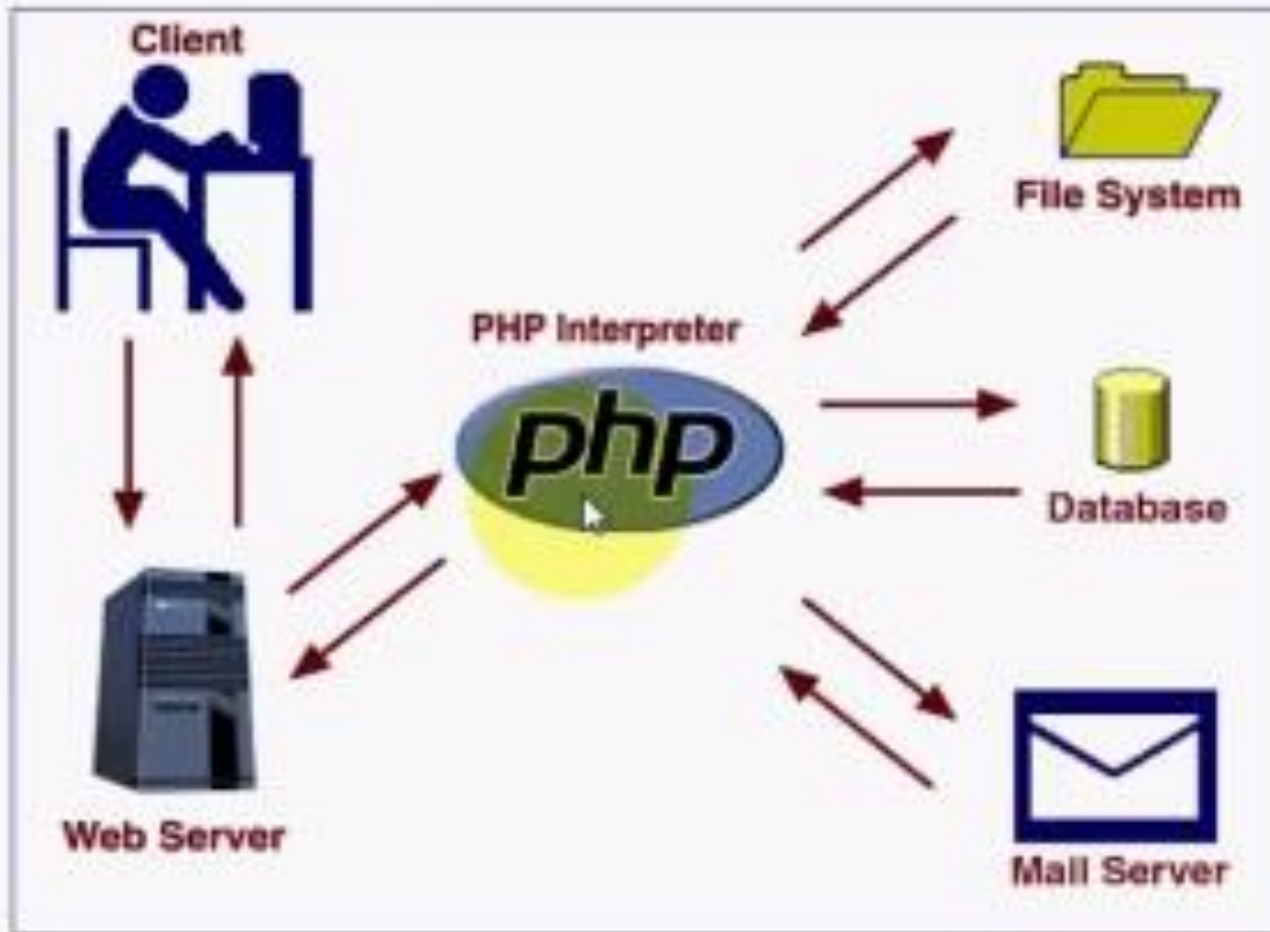
**Note:** PHP statements end with a semicolon (;).

# WAMP,LAMP and XAMP

- WAMP (Window Based Apache, MySQL and PHP)
- LAMP (Linux Based Apache, MySQL and PHP)
- XAMP (Cross-Platform Apache, MySQL and PHP)
- AMP refers to Apache, MySQL, and PHP, all of which work together to help you develop dynamic web sites.



# How PHP Works ?





# Creating Your First Program

```
< html >  
< head >  
< title > My First PHP Program < /title >  
< /head >  
< body >  
< ?php  
echo "Hello everyone.";  
? >  
< /body >  
< /html >
```

## Comments can be used to:

- Let others understand what you are doing
- Comments can remind you of what you were thinking when you wrote the code

// This is a single-line comment

OR

# This is also a single-line comment

/\*

This is a multiple-lines comment block  
that spans over multiple  
lines

\*/

# PHP 5 Variables

- Rules for PHP variables:
  1. A variable starts with the \$ sign, followed by the name of the variable.
  2. A variable name must start with a letter or the underscore character.
  3. A variable name cannot start with a number
  4. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
  5. Variable names are case-sensitive (\$age and \$AGE are two different variables)

# PHP Case Sensitivity

- In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.
- However all variable names are case-sensitive.

Example

```
<!DOCTYPE html>
<html>
<body>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
</body>
</html>
```

Output: Hello World!  
Hello World!  
Hello World!

```
<!DOCTYPE html>
<html>
<body>
<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR .
"<br>";
echo "My boat is " . $coLoR . "<br>";
?>
</body> </html>
```

Output: My car is red  
My house is  
My boat is

# PHP Data Types

- PHP supports the following data types:
    - String
    - Integer
    - Float (floating point numbers - also called double)
    - Boolean
    - Array
    - Object
    - NULL
    - Resource
- 
- The diagram uses blue curly braces to group the data types into three categories:
- Scalar**: Includes String, Integer, Float (floating point numbers - also called double), and Boolean.
  - Compound**: Includes Array and Object.
  - Special**: Includes NULL and Resource.

# PHP String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:
- ```
<?php  
$x = "Hello world!";
```

```
echo "hello $x";  
echo "<br>";  
echo 'hello $x';  
?>
```

# PHP Integer

- An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
- Rules for integers:
  - An integer must have at least one digit
  - An integer must not have a decimal point
  - An integer can be either positive or negative
- ```
<?php  
$x = 5985;  
var_dump($x);  
?>
```
- Note: The PHP `var_dump()` function returns the data type and value.
- ```
echo PHP_INT_MAX;
```

# PHP Float

- A float (floating point number) is a number with a decimal point or a number in exponential form.
- In the following example \$x is a float. The PHP var\_dump() function returns the data type and value:

```
<?php  
    $x = 10.365;  
    var_dump($x);  
?>
```



# PHP Boolean

- A Boolean represents two possible states: TRUE or FALSE.
- `$x = true;`  
`$y = false;`
- Booleans are often used in conditional testing

# PHP NULL Value

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.
- Variables can also be emptied by setting the value to NULL:
- ```
<?php  
$x = "Hello world!";  
$x = null;  
var_dump($x);  
?
```

- Creating PHP variables:

```
<?php
```

```
$txt = "Hello world!";
```

//When you assign a text value to variable, put quotes around the value.

```
$x = 5;
```

```
$y = 10.5;
```

```
echo $txt;
```

```
echo "<br>";
```

```
echo $x;
```

```
echo "<br>";
```

```
echo $y;
```

```
?>
```

Output:

Hello world!

5

10.5

# Local and Global Scope variables

```
<?php
$x=5; // global scope

function myTest()
{
    $y=10; // local scope
    echo "<p>Test variables inside the function:<p>";
    echo "Variable x is: $x";
    echo "<br>";
    echo "Variable y is: $y";
}

myTest();

echo "<p>Test variables outside the function:<p>";
echo "Variable x is: $x";
echo "<br>";
echo "Variable y is: $y";
?>
```

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function.

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function.

# PHP Constants

- It defines constant at run time
- `define(name, value, case-insensitive)`
- 1. **name:** It specifies the constant name.
- 2. **value:** It specifies the constant value.
- 3. **case-insensitive:** Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.

```
1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP",true);//not case sensitive
3. echo MESSAGE, "</br>";
4. echo message;
5. ?>
```

```
1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP",false);//case sensitive
3. echo MESSAGE;
4. echo message;
5. ?>
```

# PHP Constants

- Only scalar data (boolean, integer, float and string) can be contained in constants.

**<?php**

```
const MESSAGE="Hello const by Javapoint PHP";  
echo MESSAGE;
```

**?>**

**1.<?php**

2. define("MSG", "JavaTpoint");

3. echo MSG, "**</br>**";

4. echo constant("MSG");

5. //both are similar

**6.?>**

# Differences between constants and variables

- There is no need to write a dollar sign (\$) before a constant, where as in Variable one has to write a dollar sign.
- Constants cannot be defined by simple assignment, they may only be defined using the define() function.
- Constants may be defined and accessed anywhere without regard to variable scoping rules.
- Once the Constants have been set, may not be redefined or undefined.
- By default, constants are global.

Variables can be local, global, or static

# Magic Constants

- Magic constants are the predefined constants in PHP which get changed on the basis of their use. They start with double underscore (\_\_) and ends with double underscore.
- They are similar to other predefined constants but as they change their values with the context, they are called **magic** constants.
- Magic constants are case-insensitive.

1. \_\_LINE\_\_
2. \_\_FILE\_\_
3. \_\_DIR\_\_
4. \_\_FUNCTION\_\_
5. \_\_CLASS\_\_
6. \_\_METHOD\_\_
7. \_\_NAMESPACE\_\_
8. ClassName::class




```
<?php
    echo "<h3>Example for __LINE__</h3>";
    // print Your current line number i.e;4
    echo "You are at line number " . __LINE__ . "<br><br>";
?>
```

```
<?php
    echo "<h3>Example for __FILE__</h3>";
    //print full path of file with .php extension
    echo __FILE__ . "<br><br>";
?>
```

```
<?php
    echo "<h3>Example for __DIR__</h3>";
    //print full path of directory where script will be placed
    echo __DIR__ . "<br><br>";
    //below output will equivalent to above one.
    echo dirname(__FILE__) . "<br><br>";
?>
```

```
<?php
    echo "<h3>Example for __FUNCTION__</h3>";
    //Using magic constant inside function.
    function test(){
        //print the function name i.e; test.
        echo 'The function name is ' . __FUNCTION__ . "<br><br>";
    }
    test();
```



```
<?php
```

```
echo "<h3>Example for __FUNCTION__</h3>";
```

```
//Using magic constant inside function.
```

```
function test(){
```

```
    //print the function name i.e; test.
```

```
    echo 'The function name is '. __FUNCTION__ . "<br><br>";
```

```
}
```

```
test();
```

# PHP Arithmetic Operators

Operator	Name	Example	Result
+	Addition	$\$x + \$y$	Sum of $\$x$ and $\$y$
-	Subtraction	$\$x - \$y$	Difference of $\$x$ and $\$y$
*	Multiplication	$\$x * \$y$	Product of $\$x$ and $\$y$
/	Division	$\$x / \$y$	Quotient of $\$x$ and $\$y$
%	Modulus	$\$x \% \$y$	Remainder of $\$x$ divided by $\$y$
**	Exponential	$\$x ** \$y$	$\$x$ raised to the power $\$y$

# PHP Assignment Operators

Assignment	Same as...	Description
<b><code>x = y</code></b>	<b><code>x = y</code></b>	The left operand gets set to the value of the expression on the right
<b><code>x += y</code></b>	<b><code>x = x + y</code></b>	Addition
<b><code>x -= y</code></b>	<b><code>x = x - y</code></b>	Subtraction
<b><code>x *= y</code></b>	<b><code>x = x * y</code></b>	Multiplication
<b><code>x /= y</code></b>	<b><code>x = x / y</code></b>	Division

# PHP String Operators

Operator	Name	Example	Result
.	Concatenation	<code>\$txt1 = "Hello"</code> <code>\$txt2 = \$txt1 . "world!"</code>	Now \$txt2 contains "Hello world!"
.=	Concatenation assignment	<code>\$txt1 = "Hello"</code> <code>\$txt1 .= " world!"</code>	Now \$txt1 contains "Hello world!"

# PHP Increment / Decrement Operators

Operator	Name	Description
<b>++\$x</b>	<b>Pre-increment</b>	Increments \$x by one, then returns \$x
<b>\$x++</b>	<b>Post-increment</b>	Returns \$x, then increments \$x by one
<b>--\$x</b>	<b>Pre-decrement</b>	Decrements \$x by one, then returns \$x
<b>\$x--</b>	<b>Post-decrement</b>	Returns \$x, then decrements \$x by one

# PHP Comparison Operators

Operator	Name	Example	Result
<b>==</b>	<b>Equal</b>	<b>\$x == \$y</b>	True if \$x is equal to \$y
<b>===</b>	<b>Identical</b>	<b>\$x === \$y</b>	True if \$x is equal to \$y, and they are of the same type
<b>!=</b>	<b>Not equal</b>	<b>\$x != \$y</b>	True if \$x is not equal to \$y
<b>&lt;&gt;</b>	<b>Not equal</b>	<b>\$x &lt;&gt; \$y</b>	True if \$x is not equal to \$y
<b>!==</b>	<b>Not identical</b>	<b>\$x !== \$y</b>	True if \$x is not equal to \$y, or they are not of the same type
<b>&gt;</b>	<b>Greater than</b>	<b>\$x &gt; \$y</b>	True if \$x is greater than \$y
<b>&lt;</b>	<b>Less than</b>	<b>\$x &lt; \$y</b>	True if \$x is less than \$y
<b>&gt;=</b>	<b>Greater than or equal to</b>	<b>\$x &gt;= \$y</b>	True if \$x is greater than or equal to \$y
<b>&lt;=</b>	<b>Less than or equal to</b>	<b>\$x &lt;= \$y</b>	True if \$x is less than or equal to \$y

# PHP Conditional Statements

In PHP we have the following conditional statements:

- **if statement** - executes some code only if a specified condition is true
- **if...else statement** - executes some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - specifies a new condition to test, if the first condition is false
- **switch statement** - selects one of many blocks of code to be executed



# The if Statement

- executes some code only if a specified condition is true

```
<?php
$per = 65
if ($per > 60) {
    echo "Congrats, You are qualified!";
}
?>
```

# The if...else Statement

- executes some code if a condition is true and another code if the condition is false

```
<?php
$per = 65;
if ($per > 60)
{
    echo "Congrats, You are qualified! 😊";
}
else
{
    echo "Sorry, You are not qualified 😞"
}
?>
```

# The if...elseif...else Statement

- specifies a new condition to test, if the first condition is false

```
<?php
$t = date("H");

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

# switch Statement

- selects one of many blocks of code to be executed

```
<?php
```

```
$favcolor = "red";
```

```
switch ($favcolor) {
```

```
case "red": echo "Your favorite color is red!"; break;
```

```
case "blue": echo "Your favorite color is blue!"; break;
```

```
case "green": echo "Your favorite color is green!"; break;
```

```
default: echo "Your favorite color is neither red, blue, nor  
green!";
```

```
}
```

```
?>
```



# **PHP FLOW CONTROL AND LOOPS**

# PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

# While Loop

- The while loop executes a block of code as long as the specified condition is true.

```
<?php
$x = 1;
while($x <= 5) {
echo "The number is: $x <br>";
$x=$x+1;
}
?>
```

# do...while Loop

- loops through a block of code once, and then repeats the loop as long as the specified condition is true

```
<?php
    $x = 1;

    do {
        echo "The number is: $x <br>";
        $x++;
    } while ($x <= 5);
?>
```



```
<?php
    $x = 6;

    do {
        echo "The number is: $x <br>";
        $x++;
    } while ($x<=5);
?>
```

What will be output of this program???

# For Loops

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

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

The example below displays the numbers from 0 to 10

# PHP foreach Loop

- The foreach loop is used to traverse the array elements. It works only on array and object. It will issue an error if you try to use it with the variables of different datatype.
- The foreach loop works on elements basis rather than index. It provides an easiest way to iterate the elements of an array.
- In foreach loop, we don't need to increment the value.

1. **foreach** (\$array **as** \$value) {

2.     //code to be executed

3. }

or

1. **foreach** (\$array **as** \$key => \$element) {

2.     //code to be executed

3. }

# PHP foreach Loop

```
<?php
    $colors=array("red", "green", "blue", "yellow");
    foreach ($colors as $value){
    echo "$value <br>";
    }
?>
```

For every loop iteration, the value of the current array element is assigned to \$value and the array pointer is moved by one, until it reaches the last array element.

```
<?php
$a = 42;
PHP
34
$b = 0;
if( $a && $b ){
echo "TEST1 : Both a and b are true<br/>";
}else{
echo "TEST1 : Either a or b is false<br/>";
}
if( $a and $b ){
echo "TEST2 : Both a and b are true<br/>";
}else{
echo "TEST2 : Either a or b is false<br/>";
}
if( $a || $b ){
echo "TEST3 : Either a or b is true<br/>";
}else{
echo "TEST3 : Both a and b are false<br/>";
}
if( $a or $b ){
echo "TEST4 : Either a or b is true<br/>";
}else{
echo "TEST4 : Both a and b are false<br/>";
}
}
```

```
<?php
```

```
$x = "abc";
```

```
$$x = 200;
```

```
echo $x."<br/>";
```

```
echo $$x."<br/>";
```

```
echo $abc;
```

```
?>
```

abc

200

200



```
<?php
```

```
$name="vit";
```

```
${$name}="viit";
```

```
${${$name}}="vu";
```

```
echo $name. "<br>";
```

```
echo ${$name}. "<br>";
```

```
echo ${${$name}}. "<br>";
```

```
echo $vit. "<br>";
```

```
echo $viit. "<br>";
```

```
?>
```

```
vit  
viit  
vu  
viit  
vu
```

<?php

\$name="100";

\${\$name}="200";

\${\${\$name}}="vu";

echo \$name. "<br>";

echo \${\$name}. "<br>";

echo \${\${\$name}}. "<br>";

echo \$vit. "<br>";

echo \$viit. "<br>";

?>

100

200

vu



# Functions

- The real power of PHP comes from its functions; it has more than 1000 built-in functions.
- Besides the built-in PHP functions, we can create our own functions.
- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

# Functions

- Create a User Defined Function in PHP  
*function functionName() {  
code to be executed;  
}*
- Function names are NOT case-sensitive.
- Example:

```
<?php  
function writeMsg() {  
    echo "Hello world!";  
}  
  
writeMsg(); // call the function  
?>
```

# PHP Function Arguments

- Information can be passed to functions through arguments. An argument is just like a variable.

```
<?php  
function familyName($fname)  
{echo "$fname <br>";}
```

```
familyName("Jadhav");  
familyName("Deshpande");  
familyName("Joshi");  
familyName("Chandollikar");  
familyName("Godbole");  
?>
```

# PHP Function Arguments

```
<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>
```

Sum of the two numbers is :30

# PHP Function Arguments:by value

```
<?php
function familyName($fname, $year) {
echo "$fname, Born in $year <br>";}

familyName("Kulkarni", "1975");
familyName("Joshi", "1978");
familyName("Jadhav", "1983");
?>
```

O/P:

Kulkarni, Born in 1975

Joshi, Born in 1978

Jadhav, Born in 1983

# PHP Function Arguments: by reference

```
<?php
function addvalue($num)
{ $num += 5;          echo "$num\n";      }
  function addref(&$num)
{ $num += 6;          echo "$num\n";      }
```

```
$orignum = 10;
    addvalue( $orignum );
    echo "Original Value is $orignum\n";
    addref( $orignum );
    echo "Original Value is $orignum\n";
```

```
?>
```

15

Original Value is 10

16

Original Value is 16

# PHP Default Argument Value

```
<?php
```

```
function setHeight($minheight = 50) {  
    echo "The height is : $minheight <br>";  
}  
  
setHeight(350);  
setHeight(); // will use the default value of 50  
setHeight(135);  
setHeight(80);?>
```

O/P:

The height is : 350

The height is : 50

The height is : 135

The height is : 80

# Functions - Returning values

```
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;}

$s=sum(5, 10);
echo "5 + 10 = $s <br>";
$s=sum(7, 13);
echo "7 + 13 = $s <br>";
$s=sum(2, 4);
echo "2 + 4 = $s <br>" ;
?>
```

O/P:

5 + 10 = 15

7 + 13 = 20

2 + 4 = 6



# Recursion

```
<?php
function factorial($n)
{
    if ($n < 0)
        return -1; /*Wrong value*/
    if ($n == 0)
        return 1; /*Terminating condition*/
    return ($n * factorial ($n - 1));
}

echo factorial(5);
?>
```

# Dynamic Function Calls

- It is possible to assign function names as strings to variables and then treat these variables exactly as the function name itself.

```
<?php
```

```
function sayHello() {
```

```
    echo "Hello<br />";
```

```
}
```

```
    $function_holder = "sayHello";
```

```
sayHello()
```

```
$function_holder();
```

```
?>
```

Hello

Hello

**PHP does not support Function Overloading.**

# PHP Arrays

```
$cars1 = "Volvo";  
$cars2 = "BMW";  
$cars3 = "Toyota";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300? The solution is to create an array!

```
<?php  
    $cars = array("Volvo", "BMW", "Toyota");  
    echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] .  
        " .";  
?>
```

O/P:

I like Volvo, BMW and Toyota.

# PHP Array Types

- There are 3 types of array in PHP
  1. Indexed Array: These are arrays with numeric index.
  2. Associative Array: These are arrays which have a named key as index, the key can be numeric or text.
  3. Multidimensional Array: These are arrays which contain one or more arrays.

- PHP Indexed Arrays

```
$cars = array("Volvo", "BMW", "Toyota");  
$cars[0] = "Volvo";  
$cars[1] = "BMW";  
$cars[2] = "Toyota";
```

```
<?php  
$cars = array("Volvo", "BMW", "Toyota");  
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2]  
  . ".";  
?>
```

O/P:

I like Volvo, BMW and Toyota.

# Indexed Array

```
<html>
```

```
<body>
```

```
<?php
```

```
    /* First method to create array. */
```

```
    $numbers = array( 1, 2, 3, 4, 5);
```

```
    foreach( $numbers as $value ) {
```

```
        echo "Value is $value <br />";
```

```
    }
```

```
        /* Second method to create array. */
```

```
    $numbers[0] = "one";
```

```
    $numbers[1] = "two";
```

```
    $numbers[2] = "three";
```

```
    $numbers[3] = "four";
```

```
    $numbers[4] = "five";
```

```
    foreach( $numbers as $value ) {
```

```
        echo "Value is $value <br />";
```

```
    }
```

```
?>
```

```
</body>
```

```
</html>
```

Value is 1

Value is 2

Value is 3

Value is 4

Value is 5

Value is one

Value is two

Value is three

Value is four

Value is five

# The **count()** Function

- Get The Length of an Array -The **count()** Function

```
<?php $cars = array("Volvo", "BMW", "Toyota");  
echo count($cars) ;?>
```

**O/P: 3**

- Loop Through an Indexed Array

```
<?php  
$cars = array("Volvo", "BMW", "Toyota");  
$arlength = count($cars);  
  
for($x = 0; $x < $arlength; $x++)  
{  
    echo $cars[$x];  
    echo "<br>";  
}  
?>
```

# Associative Arrays [key=>value]

- We can associate name with each array elements in PHP using => symbol.
- will have their index as string so that you can establish a strong association between key and values.
- To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.



```
<?php
```

```
/* First method to create associative array. */
```

```
$salaries = array("mohammad" => 2000, "qadir" => 1000, "zara"  
=> 500);
```

```
echo "Salary of mohammad is ". $salaries['mohammad'] . "<br/>";
```

```
echo "Salary of qadir is ". $salaries['qadir']. "<br />";
```

```
echo "Salary of zara is ". $salaries['zara']. "<br />";
```

```
/* Second method to create array. */
```

```
$salaries['mohammad'] = "high";
```

```
$salaries['qadir'] = "medium";
```

```
$salaries['zara'] = "low";
```

```
echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
```

```
echo "Salary of qadir is ". $salaries['qadir']. "<br />";
```

```
echo "Salary of zara is ". $salaries['zara']. "<br />";
```

```
?>
```

Salary of mohammad is 2000

Salary of qadir is 1000

Salary of zara is 500

Salary of mohammad is high

Salary of qadir is medium

Salary of zara is low

<html>

<body>

<?php

```
$salary = array("mohammad" => 2000,  
"qadir" => 1000, "zara" => 500);
```

```
foreach($salary as $k => $v) {  
echo "Key: ".$k." Value: ".$v."<br/>";  
}
```

?>

</body></html>

```
<?php
$scars = array("Hondacity", "Baleno", "Creta");
// getting the array of keys/index strings
$keys = array_keys($scars);
foreach($scars as $keys=>$val )
{
    echo $keys."=>".$val."<br>";
}
?>
```

```
0=>Hondacity
1=>Baleno
2=>Creta
```

# Multi-dimensional Array

- In multi-dimensional array, each element in the main array can also be an array.
- It allows you to store tabular data in an array.
- And each element in the sub-array can be an array, and so on.
- Values in the multi-dimensional array are accessed using multiple index.

# Multi-dimensional Array

```
$emp = array
```

```
(
```

```
array(1,"sonoo",400000),
```

```
array(2,"john",500000),
```

```
array(3,"rahul",300000)
```

```
);
```

Id	Name	Salary
1	sonoo	400000
2	john	500000
3	rahul	300000

```
<?php
```

```
$emp = array
```

```
(
```

```
  array(1,"sonoo",400000),
```

```
  array(2,"john",500000),
```

```
  array(3,"rahul",300000)
```

```
);
```

```
for ($row = 0; $row < 3; $row++) {
```

```
  for ($col = 0; $col < 3; $col++) {
```

```
    echo $emp[$row][$col]." ";
```

```
  }
```

```
  echo "<br/>";
```

```
}
```

```
?
```

1 sonoo 400000

2 john 500000

3 rahul 300000

```
<?php
```

```
$marks = array(  
    "mohammad" => array (  
        "physics" => 35,  
        "maths" => 30,  
        "chemistry" => 39  
    ),  
    "qadir" => array (  
        "physics" => 30,  
        "maths" => 32,  
        "chemistry" => 29  
    ),  
    "zara" => array (  
        "physics" => 31,  
        "maths" => 22,  
        "chemistry" => 39  
    )  
);
```

```
/* Accessing multi-dimensional array  
values */
```

```
    echo "Marks for mohammad in  
physics : " ;  
    echo $marks['mohammad']['physics']  
    . "<br />";  
    echo "Marks for qadir in maths : ";  
    echo $marks['qadir']['maths'] . "<br  
/>";  
    echo "Marks for zara in chemistry  
: " ;  
    echo $marks['zara']['chemistry'] . "<br  
/>";  
    ?>
```

Marks for mohammad in physics : 35  
Marks for qadir in maths : 32  
Marks for zara in chemistry : 39

```
<?php
$cars = array(
    "Urus" => array(
        "type"=>"SUV",
        "brand"=>"Lamborghini"
    ),
    "Cayenne" => array(
        "type"=>"SUV",
        "brand"=>"Porsche"
    ),
    "Bentayga" => array(
        "type"=>"SUV",
        "brand"=>"Bentley"
    ),
);
```



```
$size = count($cars);  
$keys = array_keys($cars);  
// using the for loop  
for($i = 0; $i < $size; $i++)  
{  
    echo $keys[$i]."\n";  
    foreach($cars[$keys[$i]] as $key => $value)  
    {  
        echo $key . " : " . $value . "\n";  
    }  
    echo "\n";  
}  
?>
```

```
Urus  
type : SUV brand : Lamborghini  
Cayenne  
type : SUV brand : Porsche  
Bentayga  
type : SUV brand : Bentley
```

# PHP - Sort Functions For Arrays

- `sort()` - sort arrays in ascending order
- `rsort()` - sort arrays in descending order
- `asort()` - sort associative arrays in ascending order, according to the value
- `ksort()` - sort associative arrays in ascending order, according to the key
- `arsort()` - sort associative arrays in descending order, according to the value
- `krsort()` - sort associative arrays in descending order, according to the key

# Sort Functions For Arrays

```
<?php  
$numbers = array(4, 6, 2, 22, 11);  
sort($numbers);
```

```
$arrlength = count($numbers);
```

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

2  
4  
6  
11  
22

## Sort Array in Descending Order - rsort()

```
<?php
    $numbers = array(4, 6, 2, 22, 11);
    rsort($numbers);

    $arrrlength = count($numbers);

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

22  
11  
6  
4  
2

## Sort Array (Ascending Order), According to Value - asort()

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

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ",Value=" . $x_value;
    echo "<br>";
}
?>
```

Key=Peter, Value=35

Key=Ben, Value=37

Key=Joe, Value=43

## Sort Array (Ascending Order), According to Key - ksort()

- ```
<?php
$age
= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ",Value=" . $x_value;
    echo "<br>";
}
?>
```

# array\_reverse() function

- PHP array\_reverse() function returns an array containing elements in reversed order.

```
<?php
$numbers = array(4, 6, 2, 22, 11);
$reversearr=array_reverse($numbers);
foreach( $reversearr as $s )
{
    echo "$s<br />";
}
?>
```

# array\_search() function

- PHP array\_search() function searches the specified value in an array. It returns key if search is successful.

```
<?php
```

```
$numbers = array(4, 6, 2, 22, 11);
```

```
$key=array_search(22,$numbers);
```

```
echo "$key";
```

```
?>
```



# array\_intersect()

PHP array\_intersect() function returns the intersection of two array. In other words, it returns the matching elements of two array.

```
1. <?php
2. $name1=array("sonoo","john","vivek","smith");
3. $name2=array("umesh","sonoo","kartik","smith");
4. $name3=array_intersect($name1,$name2);
5. foreach( $name3 as $n )
6. {
7.     echo "$n<br />";
8. }
9. ?>
```

# String

- Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?php
```

```
$variable = "ABC";
```

```
$literally = 'My $variable will not print!\\n';
```

```
echo $literally;
```

```
echo "<br />";
```

```
$literally = "My $variable will print!\\n";
```

```
echo $literally;
```

```
?>
```

```
My $variable will not print!\\n
My ABC will print
```

# String

- A string is a sequence of characters, like "Hello world!".

```
<?php
$txt = "Hello All";
echo "I love " . $txt . "!";
?>
```

- PHP String Functions:

## 1. String Concatenation Operator:

```
<?php $string1="Hello World";
$string2="1234";
echo $string1 . " " . $string2; ?>           //Hello World 1234
```

## 2. Get The Length of a String : strlen

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

## 3. Count The Number of Words in a String

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

### 3. Reverse a String

```
<?php  
echo strrev("Hello world!"); // outputs !dlrow olleH  
?>
```

### 4. Search For a Specific Text Within a String

```
<?php  
echo strpos("Hello world!", "world"); // outputs 6  
?>
```

- NOTE: If no match is found, it will return FALSE.

### 5. Replace Text Within a String

```
<?php  
echo str_replace("world", "Dolly", "Hello world!"); //  
outputs Hello Dolly!  
?>
```

- NOTE: function replaces some characters with some other characters in a string.

6. string in lowercase letter.

```
<?php
```

```
$str="My name is KHAN"; $str=strtolower($str); echo $str;
```

```
?>
```

7. string in uppercase letter.

```
<?php
```

```
$str="My name is KHAN"; $str=strtoupper($str); echo $str;
```

```
?>
```

8. lcfirst() function returns string converting first character into lowercase.

```
<?php
```

```
$str="MY name IS KHAN"; $str=lcfirst($str); echo $str;
```

```
?>
```

9. ucfirst() function returns string converting first character into uppercase.