PHP

What is PHP? [1]

- PHP is an acronym for "**PHP: Hypertext Preprocessor**"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

Figure 1 [1]

What is a PHP File? [1]

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do? [1]

- PHP can generate dynamic page content
- 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 be used to control user-access
- PHP can encrypt data
- With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP? [1]

- 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
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

PHP Syntax

Basic PHP syntax [1]

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**:

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

- Example 1

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

Comments in PHP [1]

- A comment in PHP code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is looking at the code.
- **Comments can be used to**:
  - **Let others understand what you are doing**
  - **Remind yourself of what you did** - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

- Example 2

PHP Case Sensitivity [1]

- In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.
- Example 3

- However; all variable names are case-sensitive.
- Example 4

PHP Variables

Variables are "containers" for storing information.

Creating (Declaring) PHP Variables [1]

- In PHP, a variable starts with the $ sign, followed by the name of the variable:

- Example 5

- After the execution of the example statements, the variable **$txt** will hold the value **Hello world!**, the variable **$x** will hold the value **5**, and the variable **$y** will hold the value **10.5**.
- **Note:** When you assign a text value to a variable, ***put quotes around the value***.
- **Note:** Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

PHP Variables [1]

- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
- Rules for PHP variables:
  - A variable starts with the **$ sign**, followed by the name of the variable
  - A variable ***name must start with a letter or the underscore character***
  - A variable name ***cannot start with a number***
  - A variable name can only contain ***alpha-numeric characters and underscores*** (A-z, 0-9, and _)
  - Variable names are ***case-sensitive*** ($age and $AGE are two different variables)

Output Variables [1]

- The PHP echo statement is often used to output data to the screen.

- Example 6

PHP is a Loosely Typed Language [1]

- In the previous example, notice that we did not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value.
- In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

PHP Variables Scope [1]

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
  - global
  - static

PHP Variables Scope: Global and Local Scope [1]

- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:
- Example 7

- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:
- Example 8

PHP Variables Scope: PHP The global Keyword [1]

- The global keyword is used to access a global variable from within a function.
- To do this, use the global keyword before the variables (inside the function):
- Example 9

- PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.
- Example 10

PHP The static Keyword [1]

- Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.
- To do this, use the **static** keyword when you first declare the variable:
- Example 11

- Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.
- **Note:** The variable is still local to the function.

PHP echo and print Statements

In PHP there are two basic ways to get output: **echo and print.**

PHP echo and print Statements [1]

- echo and print are more or less the same. They are both used to output data to the screen.
- The differences are small:
  - echo has no return value while print has a return value of 1 so it can be used in expressions.
  - echo can take multiple parameters (although such usage is rare) while print can take one argument.
  - echo is marginally faster than print.

The PHP echo Statement [1]

- The echo statement can be used with or without parentheses: echo or echo().
- **Display Text**
- The following example shows how to output text with the echo command (notice that the text can contain HTML markup): Example 12
- **Display Variables**
- The following example shows how to output text and variables with the echo statement: Example 12

The PHP print Statement [1]

- The print statement can be used with or without parentheses: print or print().
- **Display Text**
- The following example shows how to output text with the print command (notice that the text can contain HTML markup): Example 13
- **Display Variables**
- The following example shows how to output text and variables with the print statement: Example 13

PHP Data Types

PHP Data Types [1]

- Variables can store data of different types, and different data types can do different things.
- PHP supports the following data types:
  - String
  - Integer
  - Float (floating point numbers - also called double)
  - Boolean
  - Array
  - Object
  - NULL
  - Resource

PHP String Data Type[1]

- 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.
- Example 14

PHP Integer Data Type[1]

- 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
  - Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)
- The PHP var_dump() function returns the data type and value.
- Example 14

PHP Float Data Type [1]

- A float (floating point number) is a number with a decimal point or a number in exponential form.

- Example 14

PHP Boolean Data Type [1]

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

- Example 14

PHP Array Data Type [1]

- An array stores multiple values in one single variable.
- In the example 14, $cars is an array. The PHP var_dump() function returns the data type and value.
- Example 14

PHP Object Data Type [1]

- An object is a data type which stores data and information on how to process that data.
- In PHP, an object must be explicitly declared.
- First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods.
- Example 14

PHP NULL Value [1]

- 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.
- **Tip:** If a variable is created without a value, it is automatically assigned a value of NULL.
- Variables can also be emptied by setting the value to NULL:

- Example 14

PHP Resource [1]

- The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.
- A common example of using the resource data type is a database call.
- We will not talk about the resource type here, since it is an advanced topic.

PHP Strings

PHP Strings [1]

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

- **PHP String Functions**
- **Get The Length of a String**
  - The PHP **strlen()** function returns the length of a string.

- **Count The Number of Words in a String**
  - The PHP **str_word_count()** function counts the number of words in a string.

PHP Strings [1]

- **PHP String Functions**
- **Reverse a String**
  - The PHP **strrev()** function reverses a string

- **Search For a Specific Text Within a String**
  - The PHP **strpos()** function searches for a specific text within a string.
  - If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

PHP Strings [1]

- **PHP String Functions**
- **Replace Text Within a String**
  - The PHP **str_replace()** function replaces some characters with some other characters in a string.

- Example 15

PHP Constants

PHP constants [1]

- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (***no $ sign before the constant name***).
- **Note:** Unlike variables, constants are automatically global across the entire script.

PHP constants [1]

Create a PHP Constant

- To create a constant, use the **define()** function.
- Syntax
  - **define(*name*, *value*, *case-insensitive*)**
- Parameters:
  - *name*: Specifies the name of the constant
  - *value*: Specifies the value of the constant
  - *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is **false**

PHP constants [1]

Create a PHP Constant

- The example 16 creates a constant with a **case-sensitive** name.
- The example 16 creates a constant with a **case-insensitive** name.

- Example 16

PHP constants [1]

Constants are Global

- Constants are automatically global and can be used across the entire script.
- The example 17 uses a constant inside a function, even if it is defined outside the function

- Example 17

PHP Operators

PHP Operators [1]

- Operators are used to perform operations on variables and values.
- PHP divides the operators in the following groups:
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Increment/Decrement operators
  - Logical operators
  - String operators
  - Array operators

PHP Operators [1]

- **Arithmetic Operators Example 18**

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
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP 5.6)

[1]

PHP Operators [1]

- **Assignment Operators**

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

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

[⊥]

PHP Operators [1]

- **Comparison Operators** **Example 19**

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

PHP Operators [1]

- **Increment / Decrement Operators**

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

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

PHP Operators [1]

- **Logical Operators** **Example 20**

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP Operators [1]

- **String Operators** **Example 21**

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Operators [1]

- **Array Operators** → We will learn array operators after learning array concepts in detail.
- Example 39

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

PHP 5 if...else...elseif Statements

Conditional statements are used to perform different actions based on different conditions.

PHP Conditional Statements [1]

- Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.
- In PHP we have the following conditional statements:
  - **if statement** - executes some code if one condition is true
  - **if...else statement** - executes some code if a condition is true and another code if that condition is false
  - **if...elseif....else statement** - executes different codes for more than two conditions
  - **switch statement** - selects one of many blocks of code to be executed

PHP - The if Statement [1]

- The if statement executes some code if one condition is true.

- Syntax

```
if (condition) {
    code to be executed if condition is true;
}
```

- The example 22 will output "Have a good day!" if the current time (HOUR) is less than 20:
- Example 22

PHP - The if...else Statement

- The if....else statement executes some code if a condition is true and another code if that condition is false.
- Syntax

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

- The example 23 will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:
- Example 23

PHP - The if...elseif....else Statement

- The if....elseif...else statement executes different codes for more than two conditions.
- Syntax

```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

- The example 24 will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":
- Example 24

PHP - The switch Statement

- The switch statement is used to perform different actions based on different conditions.
- Use the switch statement to **select one of many blocks of code to be executed**.

PHP - The switch Statement

- Syntax

```
switch (n) {
   case label1:
      code to be executed if n=label1;          break;
   case label2:
      code to be executed if n=label3;          break;
   ...
   default:
      code to be executed if n is different from all labels;
}
```

- This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically. The **default** statement is used if no match is found.
- **Example 25**

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

PHP - Loops

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

```
while (condition is true) {
   code to be executed;
}
```

  - The example 26 first sets a variable $x to 1 ($x = 1). Then, the while loop will continue to run as long as $x is less than, or equal to 5 ($x <= 5). $x will increase by 1 each time the loop runs ($x++):
  - **Example 26**

PHP - Loops

- **The PHP do…while Loop**
  - The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.
  - Syntax

```
do {
    code to be executed;
} while (condition is true);
```

  - The example 31 first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:
  - **Example 26**

PHP - Loops

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

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

  - Parameters:
  - *init counter*: Initialize the loop counter value
  - *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
  - *increment counter*: Increases the loop counter value
  - The example 27 displays the numbers from 0 to 10:
  - **Example 27**

PHP - Loops

- **The PHP foreach Loop**
  - The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
  - Syntax

```
foreach ($array as $value) {
    code to be executed;
}
```

  - For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.
  - The example 27 demonstrates a loop that will output the values of the given array ($colors):
  - **Example 27**

PHP Functions

PHP Functions [1]

- **PHP User Defined 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.

PHP Functions [1]

- **Create a User Defined Function in PHP**
  - A user defined function declaration starts with the word "function":
  - Syntax

```
function functionName() {
    code to be executed;
}
```

  - **Note:** A function name can start with a letter or underscore (not a number).
  - Function names are NOT case-sensitive.
  - In the example 28 below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name:
  - **Example 28**

PHP Functions [1]

- **PHP Function Arguments**
  - Information can be passed to functions through arguments. An argument is just like a variable.
  - Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
  - The example 29 has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

  - **Example 29**

PHP Functions [1]

- **PHP Default Argument Value**
  - The example 30 shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:
  - **Example 30**

- **PHP Functions - Returning values**
  - To let a function return a value, use the return statement:
  - **Example 30**

- **Pass by Reference Example**

PHP Arrays

PHP Array [1]

- An array stores multiple values in one single variable:

Create an Array in PHP

- In PHP, the array() function is used to create an array:

  array();

- In PHP, there are three types of arrays:
  - **Indexed arrays** - Arrays with a numeric index
  - **Associative arrays** - Arrays with named keys
  - **Multidimensional arrays** - Arrays containing one or more arrays

PHP Array [1]

PHP Indexed Arrays

- There are two ways to create indexed arrays:
- The index can be assigned automatically (index always starts at 0), like this:

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

or the index can be assigned manually:

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

- The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:
- **Example 31**

PHP Array [1]

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

- The count() function is used to return the length (the number of elements) of an array:
- **Example 31**

Loop Through an Indexed Array

- To loop through and print all the values of an indexed array, you could use a for loop.
- **Example 31**

PHP Array [1]

PHP Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.
- There are two ways to create an associative array:

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

- or:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

- The named keys can then be used in a script:
- **Example 32**

PHP Array [1]

Multidimensional Arrays

- A multidimensional array is an array containing one or more arrays.
- PHP understands multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.
- **The dimension of an array indicates the number of indices you need to select an element.**
  - For a two-dimensional array you need two indices to select an element
  - For a three-dimensional array you need three indices to select an element

PHP Array [1]

PHP - Two-dimensional Arrays

- A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).
- First, take a look at the following table: refer next slide

PHP Array [1]

PHP - Two-dimensional Arrays

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
```

- Example 33 Example 33D

PHP Array [1]

PHP - Sorting 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

Example 34

PHP Global Variables - Superglobals

Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes.

PHP Global Variables – Superglobals [1]

- Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- The PHP superglobal variables are:
  - $GLOBALS → already learn previously
  - $_SERVER
  - $_REQUEST
  - $_POST
  - $_GET
  - $_FILES
  - $_ENV
  - $_COOKIE
  - $_SESSION

PHP Global Variables – Superglobals [1]

PHP $_SERVER

- $_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.
- The example 35 shows how to use some of the elements in $_SERVER:
- **Example 35**

PHP Global Variables – Superglobals [1]

PHP $_SERVER

Element/Code	Description
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)

PHP Global Variables – Superglobals [1]

PHP $ SERVER

$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)
$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing

PHP Global Variables – Superglobals [1]

PHP $_SERVER

$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@w3schools.com)
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80)
$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages
$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page

PHP Global Variables – Superglobals [1]

PHP $_REQUEST

- PHP $_REQUEST is used to collect data after submitting an HTML form.
- The example 36 shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:
- **Example 36**

PHP Global Variables – Superglobals [1]

PHP $_POST

- PHP $_POST is widely used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.
- The example 37 shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:
- **Example 37**

PHP Global Variables – Superglobals [1]

PHP $_GET

- PHP $_GET is used to collect form data after submitting an HTML form with method=“get". $_GET is also widely used to pass variables.
- **Example 38**

References

1. https://www.w3schools.com/php/

Thank you….