060010606

DSE 12 Open Source Web Technology

Unit 1

PHP Introduction, Array and Functions

Difference Between HTML & PHP

- Type Of Language
- Code Type
- Easy to Learn
- Compatibility
- Developers
- Integration
- When to Use
- Syntax

Difference

HTML	PHP
HTML isn't a programming language	PHP however is a full-blown programming language
HTML is client-side scripting language	PHP is server-side scripting language
It is used to produce static website.	It is used to produce dynamic web pages.

Similarity

- **Both HTML and PHP is compatible with most browsers.**
- **Both HTML and PHP have been used in web development.**
- **Both HTML and PHP integrate with AJAX for dynamic webpages.**
- **Output of PHP script will be HTML**

HTML is a mark-up language which is used to create web pages. It is the backbone of the front end development	PHP is a server side scripting language which is an open source and popular among web developers who develop dynamic websites and web applications.
HTML is static, it always remains same everytime we open.	PHP files are dynamic and output will depend on the database contents and conditions applied in the code.
HTML is easy to learn, even a small mistake in the tags will get auto adjusted by it.	PHP is also easy to learn but mistake in code will populate error in HTML structure so PHP will relatively take more time to learn.

`<html !DOCTYPE html>`
`<head>Title</head>`
`<body>Hello World</body>`
`</html>`
	`<?php`
	
	`//php code here`
	
	`?>`

HTML files can be saved as .html or .htm extensions	PHP files can use .php, .php3, .php4, .php7, .phar extensions.

Introduction to PHP

- PHP stands 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.
- PHP syntax is very similar to C.
- PHP can be embedded directly into the HTML code.
- PHP supports many databases like Oracle, MySQL, etc.

Cont.

- PHP is an interpreted language, i.e. there is no need for compilation.
- PHP is an object-oriented language.
- PHP is simple and easy to learn language.
- PHP files are saved with the ".php" file extension, and the PHP development code is enclosed in tags.
- Current version is 7.4

Importance

- PHP runs on various platforms.
- PHP is compatible with almost all servers used today (Apache, IIS, etc.).
- PHP supports a wide range of databases.
- PHP is free.
- PHP is easy to learn and runs efficiently on the server side.
- Compatibility to upload into HTML.

History

- Started as a Perl hack in 1994 by Rasmus Lerdorf (to handle his resume), developed to PHP/FI 2.0
- By 1997 up to PHP 3.0 with a new parser engine by Zeev Suraski and Andi Gutmans
- Version 5.2.4 is current version, rewritten by Zend (www.zend.com) to include a number of features, such as an object model

Unique Features of PHP

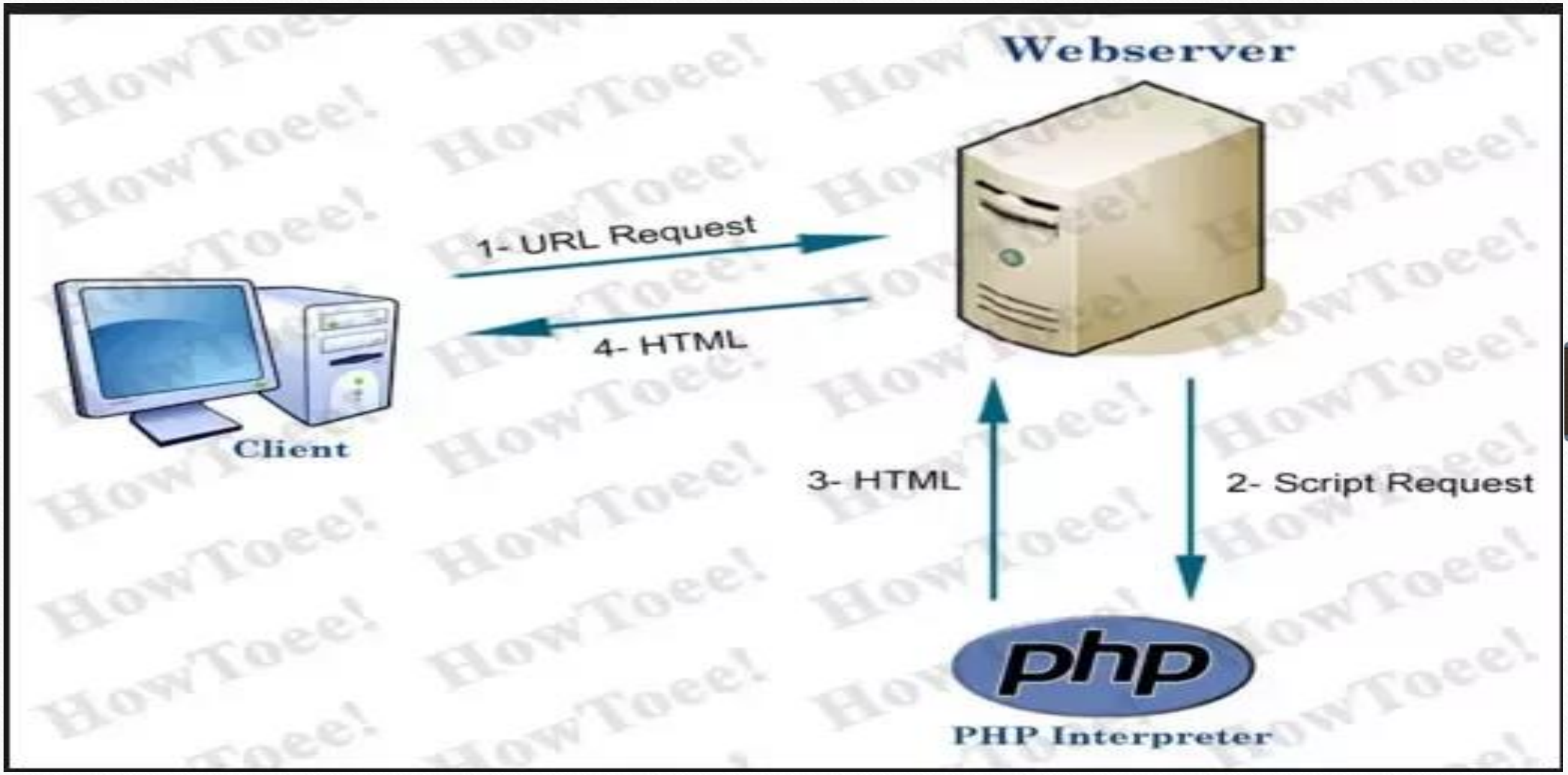
- **Performance :** faster than those written in other scripting languages.
- **Portability :** PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- **Ease of Use :** 5000+ functions included with the core distribution
- **Open Source :** free to download and use.
- **Embedded**: PHP code can be easily embedded within HTML tags and script.
- **Compatibility**: PHP is compatible with almost all local servers used today like Apache, IIS etc.

Other features

- Read and write the GIF, JPEG, and PNG image formats;
- Send and receive e-mail using the SMTP, IMAP, and POP3 protocols;
- Interface with Web services using the SOAP and REST protocols;
- Validate input using Perl regular expressions;
- Create and manipulate PDF documents.
- PHP can even access C libraries, Java classes, and COM objects and take advantage of program code written for these languages!

Do I need to compile PHP programs before executing them, as in Java or C++?

Justify - "PHP is a Loosely Typed Language"


Webserver
1- URL Request
4- HTML
Client
3- HTML
2- Script Request
php
PHP Interpreter

12

Basic Development Concepts

- Embed PHP code into one or more standard HTML documents using special "tags," or delimiters.

```
<html>
    <head></head>
    <body>
        <div>
            <?php PHP code  ?>
        </div>
    </body>
</html>
```

Executing PHP page

- To execute a PHP page, you need to put it on a server.
- Write a path of that page on a server and your page will get executed.
- For example, if you want to execute ‘index.php’ on domain ‘www.example.com’, then you need to write –

http://www.example.com/index.php

- **Single-line comments**
  - # This is a comment, and
  - // This is a comment too. Each style comments only
- **Multi-lines comments**
  - /* */
- PHP is whitespace insensitive
- PHP is case sensitive
- Statements are expressions terminated by semicolons

Outputting Data

- **Echo() & print()**
- **echo() and print() are not functions but language constructs in PHP.**
- **The print() Statement :**
  - The print() Statement will always return 1

```
<?php
   print("<p>I love India.</p>");
?>
```

- **The echo() Statement**
  - Represented like void echo(string argument1 [, ...string argumentN]).In actually it's not a function, it's a language construct. As such it does not required parentheses.

```
<?php    echo " BMIIT is the best institute.";   ?>
```

	"echo"	"print"
type	This is a type of output string.	This is a type of output string.
parenthesis	Not written with parenthesis.	It can or can not be written with parenthesis.
strings	It can output one or more strings.	It can output only one string at a time, and returns 1.
Functionality	echo is faster than print.	print is slower than echo.
argrument	echo($arg1[, $arg2....])	print($arg)

	echo	print
Parameters	echo can take more than one parameter when used without parentheses. The syntax is echo expression [, expression[, expression] ...]. Note that echo ($arg1,$arg2) is invalid.	print only takes one parameter.
Return value	echo does not return any value	print always returns 1 (integer)
Syntax	void echo (string $arg1 [, string $...])	int print (string $arg)
What is it?	In PHP, echo is not a function but a language construct.	In PHP, print is not a really function but a language construct. However, it behaves like a function in that it returns a value.

Outputting Data

- **The printf() Statement :**

  The printf() function outputs a formatted string.

- **The sprintf() Statement**
  - **The sprintf() function writes a formatted string to a variable.**
  - **Replace the percent (%) sign by a variable passed as an argument**

```php
<?php
$number = 5;
$str = ”BMIIT";
$txt = sprintf("%u Integrated MSc IT,%s.",$number,$str);
echo $txt;
?>
```

- **Print_r() Statement:-** Print the information about some variables in a more human-readable way

Possible format values:

- ✓ %% - Returns a percent sign
- ✓ %b - Binary number
- ✓ %c - The character according to the ASCII value
- ✓ %d - Signed decimal number (negative, zero or positive)
- ✓ %e - Scientific notation using a lowercase (e.g. 1.2e+2)
- ✓ %E - Scientific notation using a uppercase (e.g. 1.2E+2)
- ✓ %u - Unsigned decimal number (equal to or greather than zero)
- ✓ %f - Floating-point number (local settings aware)
- ✓ %F - Floating-point number (not local settings aware)
- ✓ %g - shorter of %e and %f
- ✓ %G - shorter of %E and %f
- ✓ %o - Octal number
- ✓ %s - String
- ✓ %x - Hexadecimal number (lowercase letters)
- ✓ %X - Hexadecimal number (uppercase letters)

- ✓ **Additional format values**
  - ✓ These are placed between the % and the letter (example %.2f):
  - ✓ + (Forces both + and - in front of numbers. By default, only negative numbers are marked)
  - ✓ ' (Specifies what to use as padding. Default is space. Must be used together with the width specifier. Example: %'x20s (this uses "x" as padding)
  - ✓ - (Left-justifies the variable value)
  - ✓ [0-9] (Specifies the minimum width held of to the variable value)
  - ✓ .[0-9] (Specifies the number of decimal digits or maximum string length)

20

Data Types

- **Integers** – are whole numbers, without a decimal point, like 4195.
- **Doubles** – are floating-point numbers, like 3.14159 or 49.1.
- **Booleans** – have only two possible values either true or false.
- **NULL** – is a special type that only has one value: NULL.
- **Strings** – are sequences of characters, like 'PHP supports string operations.'
- **Arrays** – are named and indexed collections of other values.
- **Objects** – are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** – are special variables that hold references to resources external to PHP (such as database connections).

Rules for variable declaration

1. Must start with letter or underscore.
2. Never use numeric value as an initial character of variable.
3. No need to defined datatype while declaring variable.
4. Use $ sign for declaration.
5. Variable names are case-sensitive.

$a	$_1345	$ABC
$NFC	$_ab	$abc
$1234	$_123&*()	$$

PHP Variables Scope

- PHP has three different variable scopes:
  - Local - A variable declared within a function and can only be accessed within that function.
  - Global - Declared outside a function and can only be accessed outside a function.
  - Static - The static keyword is used to declare variables in a function which keep their value after the function has ended.

```
<?php
   $a= 2.2888800;
   $b= 2.2111200;
   $c= $a+ $b;
      print("$a+ $b= $c<br>");
?>
```

For Creating Constant Variable:

bool define(String name,Mixed value,Case-Insensitive) – By default case-insensitive value is false.

e.g: define(“PI”,3.14);

PHP's Supported Data Types

Scalar Data Types – Stores Single Value

- Boolean
- Integer
- Float
- String

Compound Data Types – Store Multiple values

Array

Object

- The PHP var_dump() function returns the data type and value.

```
1.  <?php
    $x = 5985;
    var_dump($x);
    ?>
```

o/p : int(5985)

```
1.  <?php
    $s= array(“Student1”,”Student2”,“Strudent3");
    var_dump($s);
    ?>
```

o/p:array(3){[0]=> string(5) " Student1 " [1]=> string(3) " Student2 " [2]=> string(6) " Strudent3 "}

Control Structure and Looping Structure

Control Structure

- IF
- Break
- Continue
- Switch
- Include
- Include once
- Require
- Require once

Looping Structure

- While
- For
- Do While
- foreach

Conditional Statement

- The if Statement

```
if (expression) {
    statement
}
```

- Else if
- Switch

```
<?php
    switch($category) {
        case "news":
            echo "<p>What's happening around the world</p>";
            break;
        case "weather":
            echo "<p>Your weekly forecast</p>";
            break;
        case "sports":
            echo "<p>Latest sports highlights</p>";
            break;
        default:
            echo "<p>Welcome to my web site</p>";
    }
?>
```

While

- While Statement

```
while (expression) {
    statements
}
```

- Do While

```
do {
    statements
} while (expression);
```

For loop

```
for (expression1; expression2; expression3) {
    statements
}
```

- **For Each**

```
foreach (array_expr as $value) {
    statement
}
```

- The break and goto Statements

Array

An array is a data structure that stores one or more similar type of values in a single value.

- **Numeric array** − An array with a numeric index. Values are stored and accessed in linear fashion. Index start with 0.
- **Associative array** − An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
- **Multidimensional array** − An array containing one or more arrays and values are accessed using multiple indices

Numeric array

- A numeric array stores each element with a numeric ID key.

- 2 ways to write a numeric array.
  - Automatically
  - Manually

Numeric Array

- **Automatically**

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

- **Manually :**

```
$numbers[0] = 1;
$numbers[1] =2;
$numbers[2] = 3;
$numbers[3] = 4;
$numbers[] = 5;          // to append value in an array.
```

To access value

- To access specific value

```
echo $numbers[0] . " and " . $numbers[2] ;
```

- To access all values:

```
foreach( $numbers as $value ) {
        echo "Value is $value <br />";
    }
```

Associative Arrays

- Associative array will have their index as string.
- 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.

Associative Arrays

First Way

$variable= array(“Key1" => value1, " Key2" => value2, “Key3" => value3);

Example

$salaries = array(“Jugnoo" => 2000, “Swiggy" => 1000, “Zomato" => 500);

Second way :

$salaries[‘Jugnoo'] = "high";

$salaries[‘Swiggy'] = "medium";

$salaries[‘Zomato'] = "low";

Multidimensional Array

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

```
$AmazonProducts = array(
                array(“BOOK", "Books", 50),
                array("DVDs", “Movies", 15),
                array(“CDs", “Music", 20)
                );
```

Accessing multidimensional array

```
for ($row = 0; $row < 3; $row++) {
    for ($column = 0; $column < 3; $column++)
    {
        echo $AmazonProducts[$row][$column] . " <BR>";
    }
}
```

```
$AmazonProducts = array(
    array("Code" =>"BOOK", "Description" => "Books", "Price" => 50),
    array("Code" => "DVDs","Description" => "Movies","Price" => 15),
    array("Code" => "CDs", "Description" => "Music", "Price" => 20)
   );
for ($row = 0; $row < 3; $row++)
 {
   echo $AmazonProducts[$row]["Code"]. " " .
        $AmazonProducts[$row]["Description"]. " " .
        $AmazonProducts[$row]["Price"];
   }
```

Three Dimensional Array

- It is the form of multidimensional array.
- Initialization in Three-Dimensional array is same as that of Two-dimensional arrays.
- The difference is as the number of dimension increases so the number of nested braces will also increase.
- **Syntax**:

```
array (
    array (
        array (elements...),
        array (elements...),
        ...
    ),
    array (
        array (elements...),
        array (elements...),
        ...
    ),
    ...
)
```

-Three Dimensional Array

- It is the form of multidimensional array.
- Initialization in Three-Dimensional array is same as that of Two-dimensional arrays.
- The difference is as the number of dimension increases so the number of nested braces will also increase.

- **Syntax**:

```
array (
    array (
        array (elements...),
        array (elements...),
        ...
    ),
    array (
        array (elements...),
        array (elements...),
        ...
    ),
    ...
)
```


41

Foreach loop

- Used to process all the elements of an array.
- There are two forms of foreach:

    1. foreach(*$array as $value)* {}
    2. foreach(*$array as $key => $value)*{}

For each as value

```
$snacks = array("abe" => "chips", "abby" => "pretzels",
                "darin" => "candy");

foreach($snacks as $val)
{
    print("$val<br/>");
}

# Output is:
# chips
# pretzels
# candy
```

Foreach as key

```
$snacks = array("abe" => "chips", "abby" => "pretzels",
                "darin" => "candy");

foreach($snacks as $partier => $snack)
{
    print("$partier wants $snack<br/>");
}

# Output is:
# abe wants chips
# abby wants pretzels
# darin wants candy
```

Sorting

- sort($array): Sorts the values in $array, but replaces the keys by 0,1,2,3, etc...
- asort($array) : Same as sort($array), but key-value associations are preserved.
- ksort($array): Sorts $array by its keys, rather than by its values. Key-value associations are preserved.
- **rsort($array)**
- **arsort($array)**
- **krsort($array)**

These functions behave the same as sort, asort,and ksort respectively, but sort in the reverse order.

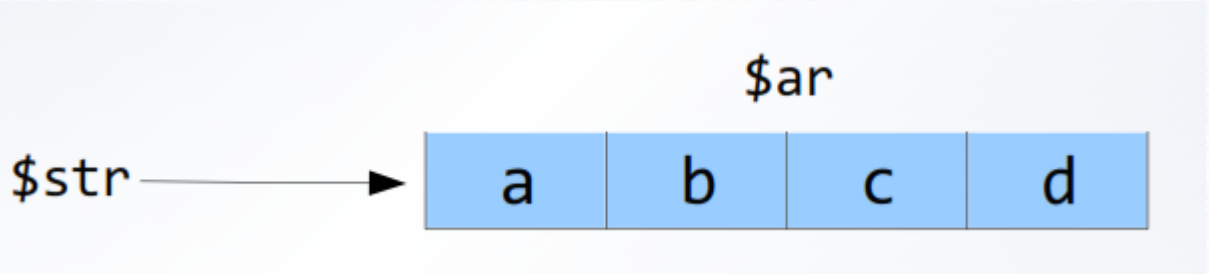
Array Function

Explode

- Explode is equivalent of split. It takes an argument of a string and a delimiter and returns an array consisting of substrings of the string.
- The explode() function breaks a string into an array.
- **Example**:

```
$str = "a:b:c:d";

$ar = explode(":",$str);
```



Array Function

Implode

- Implode does the inverse. It takes an array and returns a string where the entries are appended together using a delimiter.
- **Example**:

  $str = implode("^",$ar");

  $str has the value: "a^b^c^d".

Current pointer

- **Current : to fetch current pointer value or** Output the value of the current element in an array

```
$snacks = array("chips", "pretzels", "candy");
$snack = current($snacks);
print("The first snack is $snack");
```

Output: The first snack is chips

Moving Current Pointer Example

```
$snacks = array("chips", "pretzels", "candy");

$snack = next($snacks);
print("The 2nd snack is $snack");

$snack = prev($snacks);
print("The 1st snack is $snack");

# The 2nd snack is pretzels
# The 1st snack is chips
```

Moving pointer at start and end

- **Reset function :** Moves an array's "current" pointer to the **first element** in the array and then dereferences it.
- **End function:** Moves "current" to the **last element** in the array and then dereferences it.

```
$snacks = array("chips", "pretzels", "candy");

$first = reset($snacks);
print("The first snack is $first");

$last = end($snacks);
print("The last snack is $last");

# The first snack is chips
# The last snack is candy
```

Delete particular element from array

- **Unset Function :** Used to delete element as per assigned array index.

```
$a=array(1,2,3);
unset($a[0]);
Print_r($a);
```

Identify difference among array

- **Array_diff() :** The array_diff() function compares **the values** of two (or more) arrays, and returns the differences.This function compares the values of two (or more) arrays, and return an array that contains the entries from *array1* that are not present in *array2* or *array3*, etc.

E.g:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("e"=>"red","f"=>"black","g"=>"purple");
$a3=array("a"=>"red","b"=>"black","h"=>"yellow");

$result=array_diff($a1,$a2,$a3);
print_r($result);
?>
```

Output: Array ([b] => green [c] => blue)

More Array Function

Function name	Description
array_chunk()	function splits an array into chunks of new arrays. <?php $cars=array("Volvo","BMW","Toyota","Honda","Mercedes","Opel"); print_r(array_chunk($cars,2)); ?> o/p: Array ([0] => Array ([0] => Volvo [1] => BMW) [1] => Array ([0] => Toyota [1] => Honda) [2] => Array ([0] => Mercedes [1] => Opel))
array_column()	The array_column() function returns the values from a single column in the input array. Example in next slide
array_count_values()	function counts all the values of an array. <?php $a=array("A","Cat","Dog","A","Dog"); print_r(array_count_values($a)); ?> o/p: Array ([A] => 2 [Cat] => 1 [Dog] => 2)

Example of Array_column & array_combine

- ```
  <?php
  $a = array(
    array(
      'id' => 5698,
      'first_name' => 'Ms.',
      'last_name' => 'Gifty',
    ),
    array(
      'id' => 4767,
      'first_name' => 'Mr.',
      'last_name' => 'Smit',
    )
  );

  $last_names = array_column($a, 'last_name');
  print_r($last_names);
  ?>
  ```

- **O/P**

```
Array
(
  [0] => Gifty
  [1] => Smit
)
```

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

  $c=array_combine($fname,$age);
  print_r($c);
  ?>
  ```

O/p:

Array ([Peter] => 35 [Ben] => 37 [Joe] => 43)

- The **array_combine()** function creates an array by using the elements from one "keys" array and one "values" array.

Note: Both arrays must have equal number of elements!

Create a PHP Constant

- To create a constant, use the define() function.

define(name, value, case-insensitive)

- ✓ 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

- Constants are Global

```php
<?php
define("GREETING", "Welcome to W3Schools.com!“, true);
echo GREETING;
?>
```

PHP - Strings

- Sequences of characters
- $string_1 = "This is a string in double quotes";
- $string_0 = ""; // a string with zero characters
- Difference Between ‘ ’ and “ ”:
  - In a ‘ ‘variables value will be not evaluated. It will be consider as a string. Print as it is.
  - In a “ ” variables value will be evaluated and it will be give value of that variable.

```
$a= 'Indian';
$b = 'Hi I am $a';
$c = “Hi I am $a”;
echo $b , “<BR> , $c”;
```

String Function

- String Concatenation Operator – dot (.)

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

- strlen(): Returns length of string.

String Function

- **str_word_count():** It is used to counts the number of words in a string:.
  - echo str_word_count("Hello world!"); // outputs 2
- **strrev() :** Used to reverse string.
  - echo strrev(“BMIIT"); // outputs TIIMB

  Replace Text Within a String
- **strpos():** It is used to search for a string or character within a string.
  - // Output 6
- **str_replace():** Replace Text Within a Stringecho strpos("Hello world!","world");
  - str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!

String Function

- **Ucwords():** is used to convert first alphabet of every word into uppercase.
  - echo ucwords("welcome to the php world"); // Welcome To The Php World
- **Strtoupper()** is used to convert a whole string into uppercase.
- **Strtolower()** is used to convert a string into lowercase.
- **Strcmp(string1,string2) :** You can compare two strings by using strcmp().
  - It returns output either greater than zero, less than zero or equal to zero.
  - If string 1 is greater than string 2 then it returns greater than zero.
  - If string 1 is less than string 2 then it returns less than zero.
  - It returns zero, if the strings are equal.

- Maths Function
- Date Function
- User Define Function

Function

```
function functionName(parameters)
{
    function-body
}
```

```
function generateFooter()
{
    echo "Copyright 2010 W. Jason Gilmore";
}
```

Once defined, you can call this function like so:

```
<?php
    generateFooter();
?>
```

Passing Arguments by Value

```
function calcSalesTax($price, $tax)
{
    $total = $price + ($price * $tax);
    echo "Total cost: $total";
}
```

Passing Arguments by Reference

```
<?php

    $cost = 20.99;
    $tax = 0.0575;

    function calculateCost(&$cost, $tax)
    {
        // Modify the $cost variable
        $cost = $cost + ($cost * $tax);

        // Perform some random change to the $tax variable.
        $tax += 4;
    }
    calculateCost($cost, $tax);
    printf("Tax is %01.2f%% ", $tax*100);
    printf("Cost is: $%01.2f", $cost);

?>
```

Date Function

- Syntax:

date (format , timestamp) ;

- Characters which represents format are:
  - **d : day of life (01 to 31)**
  - **m:current month (01 to 12)**
  - **Y : current year in 4 digits**
  - **/ . - : used to separate**
- A timestamp is the number of seconds since January 1, 1970 at 00:00:00 GMT.
- This is also known as the Unix Timestamp.

Some characters that are commonly used for dates:
•d - Represents the day of the month (01 to 31)
•m - Represents a month (01 to 12)
•Y - Represents a year (in four digits)
•l (lowercase 'L') - Represents the day of the week

Some characters that are commonly used for times:
•H - 24-hour format of an hour (00 to 23)
•h - 12-hour format of an hour with leading zeros (01 to 12)
•i - Minutes with leading zeros (00 to 59)
•s - Seconds with leading zeros (00 to 59)
•a - Lowercase Ante meridiem and Post meridiem (am or pm)

Parameter	Description
format	•Required. Specifies the format of the outputted date string. The following characters can be used: •d - The day of the month (from 01 to 31) •D - A textual representation of a day (three letters) •j - The day of the month without leading zeros (1 to 31) •l (lowercase 'L') - A full textual representation of a day •N - The ISO-8601 numeric representation of a day (1 for Monday, 7 for Sunday) •S - The English ordinal suffix for the day of the month (2 characters st, nd, rd or th. Works well with j) •w - A numeric representation of the day (0 for Sunday, 6 for Saturday) •z - The day of the year (from 0 through 365) •W - The ISO-8601 week number of year (weeks starting on Monday) •F - A full textual representation of a month (January through December) •m - A numeric representation of a month (from 01 to 12)

- M - A short textual representation of a month (three letters)
- n - A numeric representation of a month, without leading zeros (1 to 12)
- t - The number of days in the given month
- L - Whether it's a leap year (1 if it is a leap year, 0 otherwise)
- o - The ISO-8601 year number
- Y - A four digit representation of a year
- y - A two digit representation of a year
- a - Lowercase am or pm
- A - Uppercase AM or PM
- B - Swatch Internet time (000 to 999)
- g - 12-hour format of an hour (1 to 12)
- G - 24-hour format of an hour (0 to 23)
- h - 12-hour format of an hour (01 to 12)
- H - 24-hour format of an hour (00 to 23)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds, with leading zeros (00 to 59)
- u - Microseconds (added in PHP 5.2.2)

•e - The timezone identifier (Examples: UTC, GMT, Atlantic/Azores)
•I (capital i) - Whether the date is in daylights savings time (1 if Daylight Savings Time, 0 otherwise)
•O - Difference to Greenwich time (GMT) in hours (Example: +0100)
•P - Difference to Greenwich time (GMT) in hours:minutes (added in PHP 5.1.3)
•T - Timezone abbreviations (Examples: EST, MDT)
•Z - Timezone offset in seconds. The offset for timezones west of UTC is negative (-43200 to 50400)
•c - The ISO-8601 date (e.g. 2013-05-05T16:34:42+00:00)
•r - The RFC 2822 formatted date (e.g. Fri, 12 Apr 2013 12:01:05 +0200)
•U - The seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)

Code

```php
<?php
   echo date("Y/m/d");
   echo "<br />";
   echo date("Y.m.d");
   echo "<br />";
   echo date("Y-m-d");
?>
```

```
2013/03/28
2013.03.28
2013-03-28
```

Include

- One of the most useful tools is to insert another php script from a file into the current php script.
- The command

```
include("filename");
```

- This will import contents of a text file called filename and insert it at the include spot.
- Any PHP in the included text must be inside the <?php tags.

Code

menu.php

```
<a href="default.php">Home</a>
<a href="about.php">About Us</a>
<a href="contact.php">Contact Us</a>
```

default.php

```
<html>
<body>
<?php include("menu.php");?>
</br><h1>Welcome to my home page</h1>
</body>
</html>
```

Final Outcome

default.php

```
<html>
<body>
<a href="default.php">Home</a>
<a href="about.php">About Us</a>
<a href="contact.php">Contact Us</a>
</br><h1>Welcome to my home page</h1>
</body>
</html>
```

include()

- In given example we have included menu.php in default.php.
- If menu.php is not found then also remaining echo statement in script will be executed.

require()

- Syntax and uses is as same as include() but the difference is that, if the file is not found the remaining script is also not executed.

include vs. require

- One big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute.
- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue
- Use require when the file is required by the application.
- Use include when the file is not required and application should continue when file is not found.

Functional Difference :

include vs include_once : There is only one difference between include() and include_once(). If the code from a file has been already included then it will not be included again if we use include_once(). Means include_once() include the file only once at a time.

include vs require : if include() is not able to find a specified file on location at that time it will throw a warning however, it will not stop script execution. For the same scenario, require() will throw a fatal error and it will stop the script execution.

require vs require_once : There is only one difference between require() and require_once(). If the code from a file has been already included then it will not be included again if we use require_once(). Means require_once() include the file only once at a time.

76

PHP Superglobal Variables

- "superglobals” :- which means that they are always accessible, regardless of scope - and can access them from any function, class or file without having to do anything special.
    - $GLOBALS
    - $_SERVER
    - $_REQUEST
    - $_POST
    - $_GET
    - $_FILES
    - $_COOKIE
    - $_SESSION

PHP Superglobal Variables

- PHP $GLOBALS
  - $GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script.
  - PHP stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable.
- PHP $_SERVER
  - $_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.
  - All the server and execution environment related information is available in this associative array.

Important elements that can go inside $_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)
$_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 script

$_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

Globals and Server

- PHP $GLOBALS

```php
<?php
$x = 75; $y = 25;
function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}
 addition();
echo $z;
?>
```

- PHP $_SERVER

```php
<?php
echo $_SERVER['PHP_SELF'], "<br>";;
echo $_SERVER['SERVER_NAME'], "<br>";;
echo $_SERVER['HTTP_HOST'], "<br>";;
echo $_SERVER['HTTP_REFERER'], "<br>";;
echo $_SERVER['HTTP_USER_AGENT'], "<br>";;
echo $_SERVER['SCRIPT_NAME'], "<br>";;
?>
```

Static Keyword

- ```
  <?php
  function myTest() {
      static $x = 0;
      echo $x;
      $x++;
  }

  myTest();
  myTest();
  myTest();
  ?>
  ```
- **O/P** : 0 1 2

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.
  ```

Request

PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

Before you can use the the $_REQUEST variable you have to have a form in html that has the method equal to GET and POST.

Then in the php, you can use the $_REQUEST variable to get the data that you wanted. Depending on what you wrote for the method in the form and using $_REQUEST in the php, $_REQUEST will use $_Get if GET is written for the method and $_REQUEST will use $POST if POST is written in the method.

The $_REQUEST syntax is ($_REQUEST['name of the form field goes here']).

Request

```
<html>
<body>
<form method="post" action=“”>
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
</body>
</html>
```

Where do PHP scripts reside?

- PHP scripts are text files residing on a server.
- When a web browser requests a document on the server that has a .php extension the file is first sent to a PHP processor.

How does PHP processor work?

- When a client requests a document that has embedded PHP, the Web server calls a **PHP processor which**
  - Copies XML code and client-side scripts verbatim to an output document as it finds them
  - Interprets the PHP scripts when they occur
  - Writes the PHP script output to the output document

The Web server then sends the output XHTML document to the client.

The output of a PHP script is an XHTML page!!

Key Point

- Tells the server where to find the PHP processor.
- The actual script is between <?php and ?>
- The stuff that is not between those tags is copied verbatim to the output document.
- The stuff that is in between the tags has to evaluate to XHTML.
- The final output of this document is an XHTML document.
- The browser only renders the XHTML document.
- It is not possible to see the PHP source if you view source on an XHTML document created from a PHP document.