ADVANCED WEB DEVELOPMENT

An Introduction to PHP

Dr. Safar M. Asaad

Koya University
Faculty of Engineering
Software Engineering Department

Outline

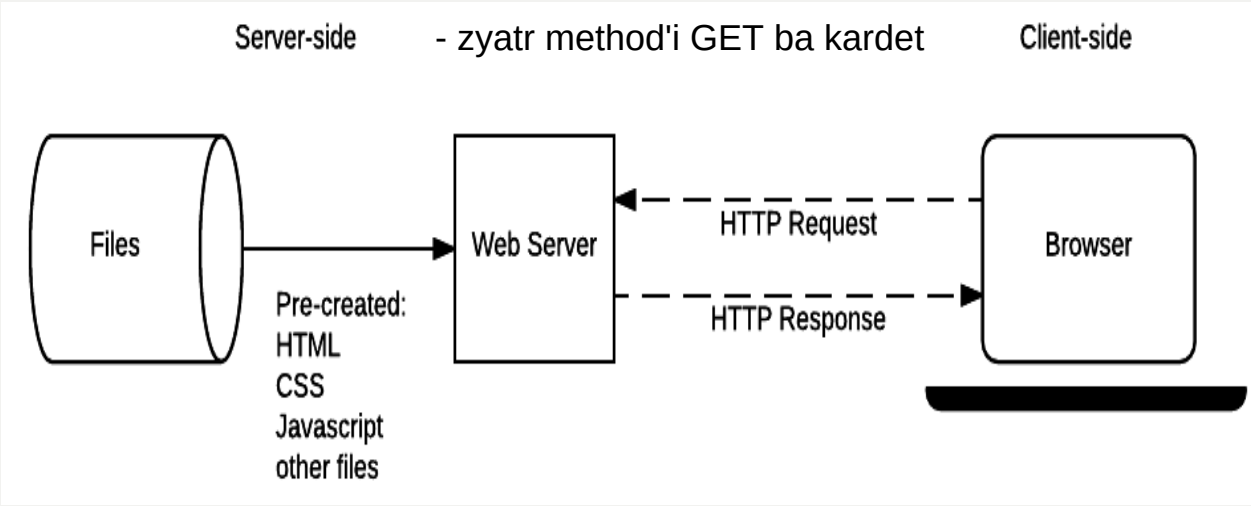
- Server-Side Web programming
  - *Static and Dynamic websites*
- PHP functions.
- Why Learn PHP?
- What do You Need?
- Comments
- Variables & Values.
  - *String*
  - *Numbers*
  - *Booleans*
  - *Null*
  - *Array*
- Operations
- Functions

Server-Side Web programming

- Web browsers communicate with web servers using HTTP.
  - *When user **click a link** on a web page, **submit a form**, or **run a search**, an **HTTP** request is sent from your browser to the target server.*
- The request includes
  - ***URL** identifying the affected resource,* - aw file'ay ka dawat krdwa
  - ***method** that defines the required action (get or post mothods),*
  - *and may include additional information encoded in **URL parameters** (the field-value pairs sent via a **query string**),*
  - *and/or **cookie data*** handek zanyarya la sar client'aka xazn dakre w dwaya ba kardetawa bo nmwna ka login dakay la site'ek daley remember me, etr bo jari dahatw ka deytawa yaksar datwani daxli bbi
- Web servers wait for client request messages, process them when they arrive, and reply to the web browser with an HTTP response message.
  - *The response contains a **status line** indicating **whether or not the request succeeded** (e.g. "HTTP/1.1 200 OK" for success).*

Static sites

- A static site is one that returns the same hard-coded content from the server whenever a particular resource is requested).
- When a user wants to navigate to a page, the browser sends an **HTTP "GET"** request specifying its URL.
- The server retrieves the requested document from its file system and returns an **HTTP response** containing the document and a **success status** (usually 200 OK).
- web server architecture for a static site



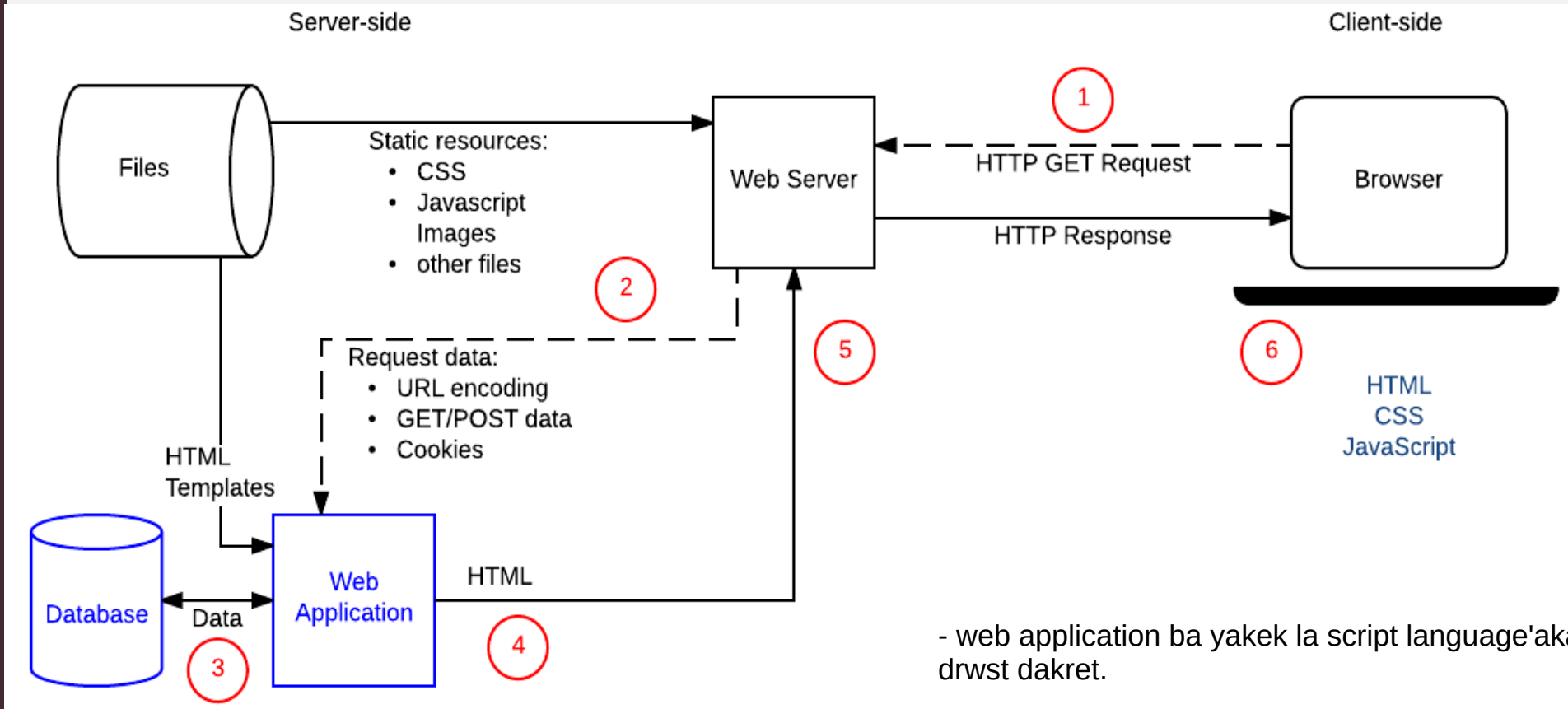
Dynamic Sites

- A dynamic website is one where some of the response **content is generated dynamically**, only when needed.
- On a dynamic website HTML pages are normally created by inserting **data from a database** into **placeholders** in HTML templates.
  - *this is a much more efficient way of **storing large amounts** of content **than** using static websites.*
- Most of the code to support a dynamic website must run on the server.
  - *Creating this code is known as "**server-side programming**" or "**back-end scripting**".*

Dynamic Sites

- The diagram below shows a simple architecture for a dynamic website.

boya lerash file system haya lawanaya basheki dynamic stie'akaman static bet bo nmwna about us page.



- web application ba yakek la script language'akan drwst dakret.

Frontend and Backend Engineering

- The components of a webserver is divided into two distinct parts:
  - *front-end*
  - *back-end.*
- front-end developer
  - *The Front End consists of the HTML, CSS, and any Client-Side Programs (i.e., JavaScript).*
- back-end developer
  - *The Back End consists of Server-Side Programs and the Database*
  - *Back-end engineers would also work on*
    - server-side configuration,
    - load balancing, bo nmwna ba kar henani zyatr la server'ek bo away load la sar server'ek ko nabetawa
    - content delivery network (CDN) and
    - server infrastructure issues.

la har shwena w server'ek dabin dakret ka to request'i data'ayak dakay gar to yakam kas bwy aw data'aya dawa bkay awa la server'i raysyawa deta nziktrin server'i lay to w etr bo kasi dwam law server'a nzikawa data'aka danerdretawa boy

Basic Models of Server-Side Programming

- Two traditional methods for creating a webpage server-side.

1. *HTML with Server-Side Code Added*

   haman HTML template'a balam la server-side content'akay dagorin, PHP la naw HTML ba kardenin

   - With this approach a webpage on the server-side looks almost like a tradition HTML file, except in a few places, we ask the server to insert new information based on code executed on the server.

2. *Server-Side Code Generating HTML*

   la server-side HTML file'i nwe drwst dakain, HTML la naw PHP ba kardenin

   - In this approach we generate a new HTML file from scratch.

Server-Side Programs

PHP

PHP

- PHP stands for "PHP Hypertext Preprocessor”
- PHP is a widely-used, open source scripting language
- An HTML-embedded server-side scripting language
- used to make web pages dynamic
  - *process form information*
  - *authenticate users* and authorization
  - *provide different content depending on context interface with other services: database, e-mail, etc.*
- generates HTML and/or client-side scripts sent to client browsers
- similar syntax to JavaScript

What Can PHP Do?

- 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.
- Session Management
- PHP can add, delete, modify data in your database.
- PHP can be used to control user access.
- PHP can encrypt data
- PHP supports object-oriented programming principles
- 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?

- Many other options: ASP.NET, ColdFusion, JSP...
- PHP is:
  - ***free and open source**: anyone can run a PHP--enabled server.*
  - ***Wide Adoption**: Many popular websites and web applications, including **Facebook**, **Wikipedia**, **WordPress**, and **Magento**, are **built** using PHP.*
  - *compatible with almost **all servers** used today (Apache, IIS, etc.)*
  - ***simple**: lots of **built-in** functionality;*
  - ***familiar syntax**: Ease of Learning and Use*
  - *PHP supports a wide range of **databases***
  - *PHP runs on **various platforms** (Windows, Linux, Unix, Mac OS X, etc.)*
  - ***Community Support**: PHP has a large and active community of **developers**, **contributors**, and users who provide support, share knowledge, and contribute to the growth of the PHP ecosystem..*
  - ***Performance**: recent versions (PHP 7 and later) have introduced significant performance improvements, including **faster execution times**, **reduced memory consumption**, and **improved error handling**.*

Why use PHP instead of JavaScript?

- PHP has access to server's important and/or private data
- avoids many browser JS compatibility issues
- faster for users (browsers) (doesn't have to run a script to view each page) la sar browser run nabet la sar server run dabet
- client can't see your source code.
- fewer security restrictions (can write to files, open web pages on other servers, connect to databases, ...)

Similarities between PHP and JavaScript

- Interpreted
- Relaxed syntax and rules
  - Both PHP and JavaScript have C-style syntax
  - variables don't need to be declared
  - Dynamic Typing: "loose" data types.
- variable names case sensitive
- built--in regular expressions

PHP files

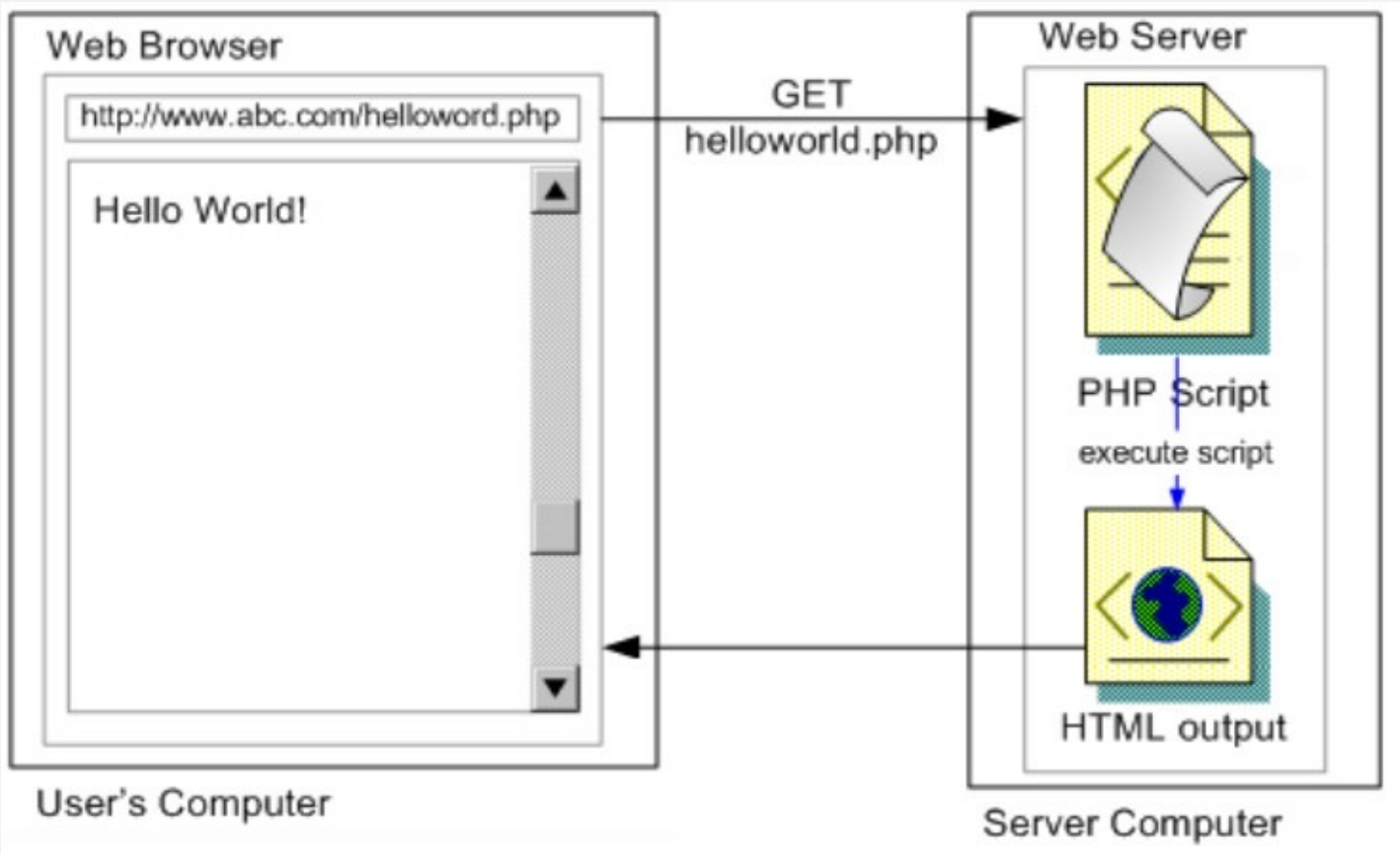
- generally have .php
- generally contain both HTML and PHP
- when a client views the source, only HTML is visible.
- all PHP script blocks start with **<?php** and end with **?>,** which can be put any where in the file.

```
<?php

// PHP code goes here

?>
```

A typical web server request using PHP



- browser requests a .html file (static content): server just sends that file.
- browser requests a .php file (dynamic content): server reads it, runs any script code inside it, then sends result across the network
- script produces output that becomes part of the HTML page

What Do You Need?

- To start using PHP, you can:
  - *Find a **web host with PHP and MySQL** support, OR*
  - *install a web **server on your own PC**, and then install PHP and MySQL.*
- Alternatively, You can Install **WAMP Server**, For Windows OS, which contains PHP and MySQL as well.
- Or You can use **MAMP**, For Mac OS.
- Or YOU CAN USE **XAMMP** FOR BOTH OPERATING SYSTEMS.

Installing WAMP Server to run PHP

1. Goto www.wampserver.com and download WAMP Server.
2. install it like other softwares by just clicking next, next…
3. Now go to START menu of windows and start wampserver.
4. Generally, the path is Start --> WapmServer --> start WampServer
5. Open your web browser and type http://localhost (or http://127.0.0.1)
6. If you see a default WampServer home page, you installation is success.
7. Put your php code in **www** folder in your wamp installation directory.
8. Usually this is **C:\wamp\www**
9. Now type **http://localhost/filename.php** in your browser. (where filename is your php file name). It will execute your php code.. Thats it...

- You can start or stop your server via an icon in the bottom right, where the clock is.

Comments in PHP

- Standard C, C++, and shell comment symbols:

```
// C++ and Java-style single line comment

# Shell-style single line comment

/* C-style comments
     These can span multiple lines */
```

PHP echo and print Statements

- print are more or less the same.
  - *They are both used to output data to the screen.*
  - *They can be used with or without parentheses.*
    - *Echo "hi"; or echo("hi");*
  - *They can contain HTML markup.*
- 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, while print can take one argument. (**print** produces a parse error with multiple parameters)*

la zorbay 7altakan hasti pe nakret

✓ *echo is marginally faster than print (it is negligible in most cases)*

Creating (Declaring) PHP Variables

- In PHP, a variable starts with the $ sign, followed by the name of the variable.
- 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).*

Creating (Declaring) PHP Variables

- Example:

```
<?php
    $txt = "Hello world!";          //holds the value Hello world!
    $x = 5;                         //holds the value 5
    $y = 10.5;                      //holds the value 10.5
    echo $txt ;
    echo $x, $y;
    ?>
```

- **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 Case Sensitivity

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

  - For example, all three echo statements are legal (equal):

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

  - However; all variable names are case-sensitive.
    - $color, $COLOR, and $coLOR are treated as three different variables

```
<?php
$color = "red";
echo "My car is " . $color . "<br>";
 echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

PHP Data Types

- Variables can store data of different types, and different data types can do different things.
- PHP supports the following data types.

1. *String*
2. *Numbers*
3. *Boolean*
4. *Array*
5. *Object*
6. *NULL*

String

A string is any characters between two quotes,

For Example:

- *"This is a string ",*
- *'This is also a string'*

- Example:

```
<?php
$fName="Azad";
$sName='Ahmed'
Echo "Hello ". $fName." ".$sName;
?>
```

Numbers

- There are only two types of numbers:
  - ✓ *Integer number*
  - ✓ *Floating point number.*
- You don't have to worry about those types, because PHP will convert back and forth when necessary.
- Example:

```
<?php
$Number=10;
$Number2=10.3;
Echo $Number.”<br>”.$Number2 ;
?>
```

Boolean

- Boolean has two values:
  1. *True*
  2. *False*
- Those are keywords in PHP which means you cannot use them for anything other than Boolean values.
- Example:

```
<?php
$students=TRUE;
 Echo $students;
?>
```

Null

- This is the data type that represents nothing: the valueless value.
- That is a keyword too.
- Example:

```
<?php
$will_be_null;
//or
$will_be_null_too = null;
?>
```

Array

- It is the most complex type that we have looked at yet, and that is because it can be made up of other variable types that we have looked at.
- Example:

```
<?php
$cars = array("Volvo","BMW",2.6, true, null, array(6,7));
 var_dump($cars);
?>
```

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

Array

There are three types of array:

1. ***Indexed arrays*** *-- Arrays with a numeric index*
2. ***Associative arrays*** *-- Arrays with named keys*
3. ***Multidimensional arrays*** *-- Arrays containing one or more arrays*

Indexed arrays

- By default, arrays use numerical indices;
- The indices start at 0, not 1.
- Example:

```
<?php
$an_array= array(”PHP","BMW",2.6, true, null,);
?>
```

$an_array[0] holds the value PHP.

$an_array[2] holds the value 2.6.

- This is also a way to get the values out of an array.

Associative arrays

- Use strings as their indices; Of course, you have to define a string for each value you put in the array.
- Example:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
 echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Or

```
<?php
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43”;
?>
```

Multidimensional arrays

- A multidimensional array is an array containing one or more arrays.
- PHP understands multidimensional arrays that are two, three, 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

```
$cars = array(    array("Volvo",22,18),
                  array("BMW",15,13),
                  array("Saab",5,2),
                 array("Land Rover",17,15)
);
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2]."<br>”;
```

Objects

More About PHP Objects in the next lectures.

PHP Operators

- Operators are used to perform operations on variables and values.
- **PHP divides the operators in the following groups:**

1. *Arithmetic operators*
2. *Assignment operators*
3. *Comparison operators*
4. *Increment/Decrement operators*
5. *Logical operators*
6. *String operators*
7. *Array operators*

Arithmetic operators

- Arithmetic operators are used with numeric values to perform common arithmetical operations:

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

Assignment operators

- assignment operators are used with numeric values to write a value to a variable:

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

Comparison operators

- 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

Increment/Decrement operators

- They are used to increment/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

Logical operators

- 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

String operators

- 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

Array operators

- 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 Conditional Statements

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

PHP Conditional Statements

1. if statement:

```
<?php
$a=5;  $b=2;
if ($a > $b) {
   echo "a is bigger than b";
}
?>
```

2. if...else statement:

```
<?php
$a=5;       $b=2;
if ($a > $b) {
echo "a is bigger than b";
} else {
echo "a is smaller than b"; }
?>
```

3. if...elseif....else statement:

```
<?php
$a=5;       $b=2;
if ($a > $b) {
        echo "a is bigger than b";
} elseif ($a == $b) {
        echo "a is equal to b";
} else {
        echo "a is smaller than b"; }
?>
```

PHP Conditional Statements

- **switch statement**

```
<?php
$destination = "Tokyo";
echo "Traveling to". $destination."<br />";
switch ($destination){
        case "Las Vegas":     echo "Bring an extra $500";
                        break;
        case "Amsterdam":  echo "Bring an open mind";
                        break;
        case "Egypt":              echo "Bring 15 bottles Water";
                       break;
        case "Tokyo":             echo "Bring lots of money";
                        break;
        case "Caribbean Islands":  echo "Bring a swimsuit";
                        break;
        Default: echo " Have a good journey";
}
?>
```

PHP Loops

- In PHP, we have the following looping statements:
  1. ***while*** *-- loops through a block of code as long as the specified condition is true.*
  2. ***do...while*** *-- loops through a block of code once, and then repeats the loop as long as the specified condition is true.*
  3. ***for*** *-- loops through a block of code a specified number of times*
  4. ***foreach*** *-- loops through a block of code for each element in an array*

while Loops

1. **While:**

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

2. **do...while:**

```
<?php
$x = 6;
do {
echo "The number is:". $x." <br>";
$x++;
} while ($x <=  10);
?>
```

for Loops

3. for:

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

4.foreach:

Syntax

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

Example

```
<?php
$fruits = array ("Orange", "Apple", "Banana",  "Cherry", " Mango");
foreach ( $fruits as $value )
{
echo  "$value<br />";
}?>
```

Loop Through an Array

Indexed Array:

The count() function is used to return the length (the number of elements) of an array

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

❖ **Associative Array:**

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $x_value) {
echo "Key=" . $x . ", Value=" . $x_value; echo "<br>";
}
?>
```

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- PHP has three different variable scopes:

1. *Global*
   - A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function
2. *Local*
   - A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:
3. *Static*
   - 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

Global and Local Scope

Note:

You can have local variables **with the same name** in different functions, because local variables are only recognized by the function in which they are declared.

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

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
```

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

The global Keyword

- 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)

```
<?php
$x = 5;  $y = 10;
function myTest() {
   global $x, $y;
   $y = $x + $y;
}
myTest();
echo $y; // outputs 15
?>
```

- 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.

```
<?php
$x = 5;   $y = 10;
function myTest() {
 $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}
myTest();
echo $y; // outputs 15
?>
```

PHP The static Keyword

- 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
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
myTest();
myTest();
?>
```