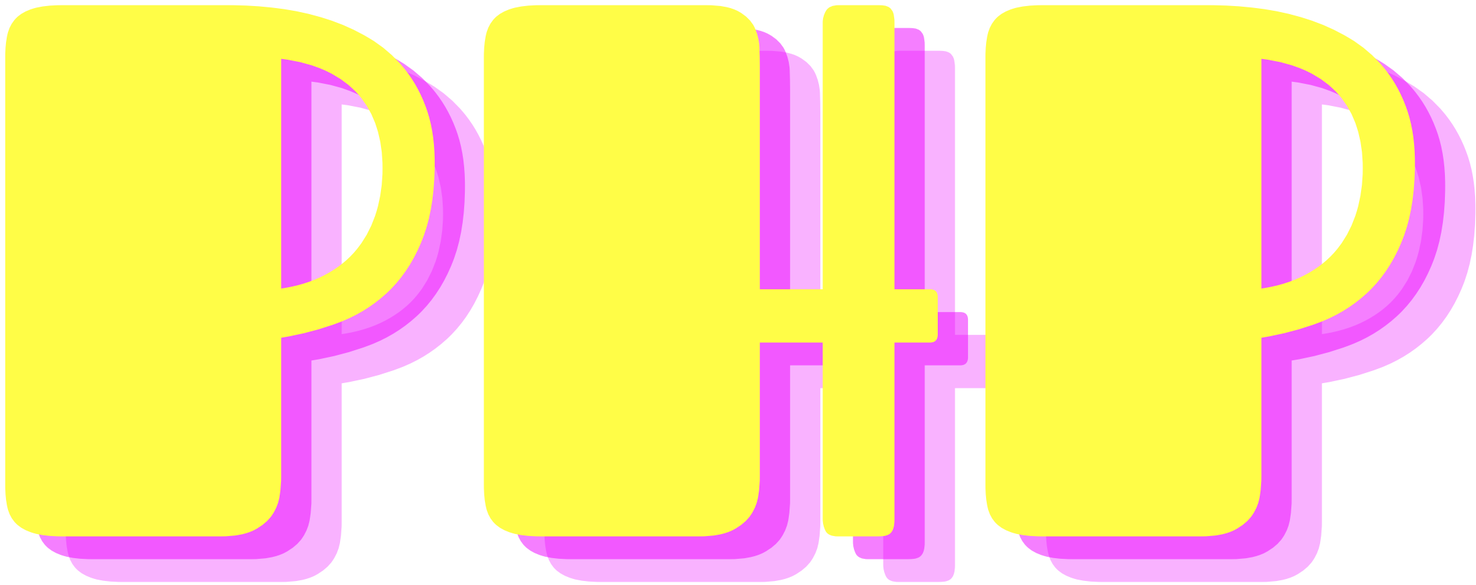
POP

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

- **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**

- **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**

A PHP script starts with <?php and ends with ?>

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

// This is a single-line comment

This is also a single-line comment

PHP is a Loosely Typed Language
In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

PHP Variables
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)
Remember that PHP variable names are case-sensitive!

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

- **PHP has three different variable scopes:**
-
- **local**
- **global**
- **static**

Global and Local Scope
A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<?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>";
?>
```

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

```
<?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>";
?>
```

PHP 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):

Example

```
<?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 The static Keyword

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

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

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

PHP echo and print Statements
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

PHP Data Types
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 Object
Classes and objects are the two main aspects of object-oriented programming.

A class is a template for objects, and an object is an instance of a class.

When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Let's assume we have a class named Car. A Car can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties.

When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

If you create a __construct() function, PHP will automatically call this function when you create an object from a class.

Example

```
<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
```

- **strlen()**
- **str_word_count()**
- **strrev()**
- **strpos()str_replace()**

- **round()**
- **rand()**

The real power of PHP comes from its functions.

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

PHP Global Variables - Superglobals
Some 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
$_SERVER
$_REQUEST
$_POST
$_GET
$_FILES
$_ENV
$_COOKIE
$_SESSION

PHP
LOGIN FORM

connection.php

```
<?php

$dbhost = "127.0.0.1:3305";
$dbuser = "root";
$dbpass = "";
$dbname = "login_sample_db";

if(!$con = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname))
{
    echo "suck";
 die("failed to connect!");
}
```

login.php

```php
<?php

session_start();

 include("connection.php");
 include("functions.php");


 if($_SERVER['REQUEST_METHOD'] == "POST")
 {
  //something was posted
  $user_name = $_POST['user_name'];
  $password = $_POST['password'];

  if(!empty($user_name) && !empty($password) && !is_numeric($user_name))
  {

   //read from database
   $query = "select * from users where user_name = '$user_name' limit 1";
   $result = mysqli_query($con, $query);

   if($result)
   {
    if($result && mysqli_num_rows($result) > 0)
    {

     $user_data = mysqli_fetch_assoc($result);

     if($user_data['password'] === $password)
     {

      $_SESSION['user_id'] = $user_data['user_id'];
      header("Location: index.php");
      die;
     }
    }
   }

   echo "wrong username or password!";
  }else
  {
   echo "wrong username or password!";
  }
 }

?>
```

```html
<!DOCTYPE html>
<html>
<head>
 <title>Login</title>
</head>
<body>

 <style type="text/css">

 #text{

  height: 25px;
  border-radius: 5px;
  padding: 4px;
  border: solid thin #aaa;
  width: 100%;
 }

 #button{

  padding: 10px;
  width: 100px;
  color: white;
  background-color: lightblue;
  border: none;
 }

 #box{

  background-color: grey;
  margin: auto;
  width: 300px;
  padding: 20px;
 }

 </style>

 <div id="box">

  <form method="post">
   <div style="font-size: 20px;margin: 10px;color: white;">Login</div>

   <input id="text" type="text" name="user_name"><br><br>
   <input id="text" type="password" name="password"><br><br>

   <input id="button" type="submit" value="Login"><br><br>

   <a href="signup.php">Click to Signup</a><br><br>
  </form>
 </div>
</body>
</html>
```

signup.php

```php
<?php
session_start();

 include("connection.php");
 include("functions.php");


 if($_SERVER['REQUEST_METHOD'] == "POST")
 {
  //something was posted
  $user_name = $_POST['user_name'];
  $password = $_POST['password'];

  if(!empty($user_name) && !empty($password) && !is_numeric($user_name))
  {

   //save to database
   $user_id = random_num(20);
   $query = "insert into users (user_id,user_name,password) values ('$user_id','$user_name','$password')";

   mysqli_query($con, $query);

   header("Location: login.php");
   die;
  }else
  {
   echo "Please enter some valid information!";
  }
 }
?>
```

```html
<!DOCTYPE html>
<html>
<head>
 <title>Signup</title>
</head>
<body>

 <style type="text/css">

 #text{

  height: 25px;
  border-radius: 5px;
  padding: 4px;
  border: solid thin #aaa;
  width: 100%;
 }

 #button{

  padding: 10px;
  width: 100px;
  color: white;
  background-color: lightblue;
  border: none;
 }

 #box{

  background-color: grey;
  margin: auto;
  width: 300px;
  padding: 20px;
 }

 </style>

 <div id="box">

  <form method="post">
   <div style="font-size: 20px;margin: 10px;color: white;">Signup</div>

   <input id="text" type="text" name="user_name"><br><br>
   <input id="text" type="password" name="password"><br><br>

   <input id="button" type="submit" value="Signup"><br><br>

   <a href="login.php">Click to Login</a><br><br>
  </form>
 </div>
</body>
</html>
```

functions.php

```
<?php

function check_login($con)
{

 if(isset($_SESSION['user_id']))
 {

  $id = $_SESSION['user_id'];
  $query = "select * from users where user_id = '$id' limit 1";

  $result = mysqli_query($con,$query);
  if($result && mysqli_num_rows($result) > 0)
  {

   $user_data = mysqli_fetch_assoc($result);
   return $user_data;
  }
 }

 //redirect to login
 header("Location: login.php");
 die;

}

function random_num($length)
{

 $text = "";
 if($length < 5)
 {
  $length = 5;
 }

 $len = rand(4,$length);

 for ($i=0; $i < $len; $i++) {
  # code...

  $text .= rand(0,9);
 }

 return $text;
}
```

logout.php

```
<?php

session_start();

if(isset($_SESSION['user_id']))
{
 unset($_SESSION['user_id']);

}

header("Location: login.php");
die;
```

index.php

```
<?php
session_start();

 include("connection.php");
 include("functions.php");

 $user_data = check_login($con);

?>

<!DOCTYPE html>
<html>
<head>
 <title>My website</title>
</head>
<body>

 <a href="logout.php">Logout</a>
 <h1>WELCOME LOGIN SUCCESSFULL</h1>

 <br>
 Hello, <?php echo $user_data['user_name']; ?>
</body>
</html>
```

Advanced concepts

INLCUDE & REQUIRE

The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.

Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

- **It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.**
-
- **The include and require statements are identical, except upon failure:**
-
- **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**
- **So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.**
-
- **Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.**

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

A cookie is created with the setcookie() function.

There are three steps involved in identifying returning users −

-
- **Server script sends a set of cookies to the browser. For example name, age, or identification number etc.**
-
- **Browser stores this information on local machine for future use.**
-
- **When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.**

A session is a way to store information (in variables) to be used across multiple pages.

Unlike a cookie, the information is not stored on the users computer.

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

PHP filters are used to validate and sanitize external input.

The PHP filter extension has many of the functions needed for checking user input, and is designed to make data validation easier and quicker.

- **Many web applications receive external input. External input/data can be:**
-
- **User input from a form**
- **Cookies**
- **Web services data**
- **Server variables**
- **Database query results**

You should always validate external data!
Invalid submitted data can lead to security problems and break your webpage!
By using PHP filters you can be sure your application gets the correct input!