PHP in Web Technologies

PHP (Hypertext Preprocessor) is a widely-used, open-source, server-side scripting language that is especially suited for web development. It is embedded within HTML and enables the creation of dynamic web pages that interact with users, databases, and other web services.

- PHP was created by Rasmus Lerdorf in 1994 and initially stood for "Personal Home Page."
- It has since evolved to stand for "PHP: Hypertext Preprocessor," with a focus on server-side scripting to produce dynamic web content.

Key Features of PHP

- **Server-Side Execution**: PHP runs on the server, making it invisible to users, who only see the processed HTML output.
- **Embedded in HTML**: PHP can be seamlessly embedded within HTML, making it simple to use alongside web page markup.
- **Cross-Platform Compatibility**: PHP is compatible with major operating systems (Windows, Linux, macOS) and web servers (Apache).
- **Free and Open Source**: PHP is free to use, and being open source, it benefits from a large community for support and improvements.
- **Session and Cookie Management**: PHP can manage user sessions and cookies, essential for user authentication and personalization.

Common PHP Functions in Web Development

- **Form Handling**: PHP can handle data submitted through HTML forms using $_POST and $_GET superglobals.
- **Database Operations**: PHP's PDO (PHP Data Objects) allows secure interaction with databases.
- **File Handling**: PHP can read, write, and manage files on the server, useful for file uploads and downloads.
- **Session Management**: PHP sessions allow data persistence across web pages, supporting user login systems and shopping carts.
- Using PHP, we can restrict users to access some pages of your website.
- It can encrypt data.

PHP in Today's Web Development.

- PHP remains widely used in content management systems (CMS) and e-commerce platforms (Magento).
- As new languages and frameworks emerge, PHP continues to evolve with modern updates (such as PHP 8) that improve performance, security, and functionality.

PHP – SYNTAX OVERVIEW

<?php……....?>

Example: <!DOCTYPE html>

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

- In PHP, keywords (e.g. if, else, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.
- echo and print are more or less the same. They are both used to output data to the screen.
- Single-line comments:

```
// This is a single-line comment

# This is also a single-line comment
```

- Multi-lines comments:

```
/* This is a
multi-line comment */
```

PHP – VARIABLE TYPES

- All variables in PHP are denoted with a leading dollar sign ($)
- PHP has three different variable scopes:

1. local
2. global
3. static

PHP has a total of eight data types which we use to construct our variables:

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

"Superglobals" are predefined variables in PHP that are always accessible, independent of scope, and can be accessed from any function, class, or file without requiring any particular actions.

The PHP superglobal variables are:

1. **$GLOBALS**-Global variables are variables that can be accessed from any scope.

```
<?php
$age= 25;

function test() {
echo $GLOBALS['age'];
}

test()
?>
```

2. **$_SERVER**-$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
?>
```

3. **$_REQUEST**-$_REQUEST is a PHP super global variable which contains submitted form data, and all cookie data.
4. **$_POST**-$_POST contains an array of variables received via the HTTP POST method.
5. **$_GET**- $_GET contains an array of variables received via the HTTP GET method.
6. **$_FILES-**A file is uploaded when a HTML form contains an input element with a file type, its enctype attribute set to multipart/form-data, and the method attribute set to HTTP POST method.

```
<form action="home.php" method="POST" enctype="multipart/form-data">
    <p><input type="file" name="file"></p>
    <p><input type ="submit" value="submit"></p>
  </form>
```

7. **$_ENV-**$_ENV is another superglobal associative array in PHP. It stores environment variables available to current script.

8. **$_COOKIE-** The $_COOKIE superglobal variable in PHP is an associative array that stores all the cookies that have been sent to the server from the user's browser. A cookie is often used to identify a user.
9. **$_SESSION-**In PHP, sessions are a way to store data on the server for individual users. Unlike cookies, session data is stored on the server, and only a session ID is stored on the client side in a cookie or passed via URL. This provides a secure way to maintain state across multiple requests or pages.

Key Functions in PHP Sessions:

- Start a Session: session_start();
- Create Session Variables: $_SESSION['username'] = 'Web';
- Access Session Variables: echo $_SESSION['username'];
- Unset a Specific Session Variable: unset($_SESSION['username']);
- Destroy the Entire Session: session_destroy();

Key Differences between Cookies and Sessions:

Aspect	Cookies	Sessions
Storage	Data is stored on the client-side (in the browser).	Data is stored on the server-side.
Data Size	Limited to about 4KB per cookie.	Can store much larger amounts of data (limited by server resources).
Lifetime	Cookies can persist even after the browser is closed (depending on the expiry time).	Sessions typically last only until the browser is closed or the session expires.
Security	Less secure because data is stored on the user's computer and can be tampered with.	More secure because data is stored on the server and only a session ID is shared with the client.
Transmission	Sent with every HTTP request (in headers), potentially increasing bandwidth usage.	Only the session ID is sent with HTTP requests.
Setup	Requires setting cookies manually using setcookie() or headers.	Requires session_start() to initialize. Data is stored in the $_SESSION array.
Expiration	Controlled by the developer using the *expires* or *max-age* attribute.	Sessions expire when the user closes the browser or after a specified time (controlled by the server).
Accessibility	Can be accessed and modified directly by the user, making it less secure.	Users cannot directly access session data stored on the server.
Use Cases	Storing user preferences (e.g., language, theme).	- Managing user login/authentication.

	Keeping users logged in with a "Remember Me" feature. Tracking analytics (e.g., unique user visits).	- Storing temporary data (e.g., shopping cart).

PHP DATE AND TIME

The PHP date () function formats:

Here are 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

Example:

```
<?php
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("l");
echo date("Y"); // Automatic Copyright Year
?>
```

TIME

Here are 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)

Example:

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

PHP Include Files

In PHP, the include and require statements are used to include and execute code from one PHP file into another. They are particularly useful for reusing code, such as including header, footer, or configuration files.

Difference between include and require:

Aspect	include	require
Behavior on Failure	Generates a warning (E_WARNING) and continues script execution if the file is not found.	Generates a fatal error (E_COMPILE_ERROR) and stop the script execution if the file is not found.
Syntax	include '*filename*';	require 'filename';
Error Handling	Ideal for including optional files.	Ideal for including critical files.