PHP Basic Notes

1. Introduction to PHP

PHP (Hypertext Preprocessor) is a widely-used open-source server-side scripting language that is especially suited for web development. It can be embedded into HTML.

Key Features:

- Open-source
- Easy to learn
- Cross-platform
- Server-side execution

2. Setting Up PHP Environment

Installing XAMPP:

- Download from Apache Friends.
- Install and start the Apache and MySQL modules.
- Place your PHP files in the htdocs directory.

Running PHP Files:

- Save your PHP files with a .php extension.
- Access your files via http://localhost/filename.php.

3. PHP Syntax and Variables

Basic Syntax:

- PHP code is executed on the server.
- PHP code starts with .

Example:

php

```
<?php
  echo "Hello, World!";
?>
```

Variables:

- Variables in PHP start with a $ sign followed by the variable name.
- Variables are case-sensitive.

Example:

php

```php
<?php
  $name = "John";
  $age = 25;
  echo $name;
?>
```

4. Data Types

PHP supports various data types:

Scalar Types:

- **String:** A sequence of characters.
- **Integer:** Whole numbers.
- **Float:** Numbers with decimal points.
- **Boolean:** true or false.

Composite Types:

- **Array:** Collection of values.
- **Object:** Instances of classes.

Special Types:

- **NULL:** Represents a variable with no value.

Example:

php

```php
<?php
  $string = "Hello";
  $int = 20;
  $float = 10.5;
  $bool = true;
?>
```

5. Operators

Arithmetic Operators:

- + (Addition)
- - (Subtraction)

- * (Multiplication)
- / (Division)
- % (Modulus)

Example:

php

```
<?php
  $x = 10;
  $y = 6;
  echo $x + $y; // Outputs 16
?>
```

Comparison Operators:

- ==(Equal)
- != (Not equal)
- > (Greater than)
- < (Less than)
- >= (Greater than or equal to)
- <= (Less than or equal to)

Logical Operators:

- && (And)
- || (Or)
- ! (Not)

Example:

php

```
<?php
  $x = 10;
  $y = 20;
  if ($x == $y) {
      echo "Equal";
  } else {
      echo "Not Equal";
  }
?>
```

6. Control Structures

Conditional Statements:

- **if**: Executes a block of code if a specified condition is true.
- **else**: Executes a block of code if the condition in the if statement is false.
- **elseif'**: Executes a block of code if the first if condition is false and the new condition is true.
- **switch**: Selects one of many blocks of code to execute.

Example:

php

```
<?php
  $num = 10;
  if ($num > 0) {
      echo "Positive Number";
  } elseif ($num < 0) {
      echo "Negative Number";
  } else {
      echo "Zero";
  }
?>
```

Loops:

- **for**: Loops through a block of code a specified number of times.
- **while**: Loops through a block of code as long as the specified condition is true.
- **do...while**: Executes a block of code once, then repeats as long as the condition is true.
- **foreach**: Loops through a block of code for each element in an array.

Example: Of `For` Loop !

php

```
<?php
  for ($i = 0; $i < 5; $i++) {
      echo $i;
  }
?>
```

Example : Of `While` Loop !

php

```
$i = 1;
while ($i <= 5) {
    echo $i;
    $i++;
}
```

Example : Of `do..While` Loop !

php

```
$i = 1;
do {
    echo $i;
    $i++;
} while ($i <= 5);
```

Example : Of `Foreach` Loop !

php

```
$numbers = [1, 2, 3, 4, 5];
foreach ($numbers as $number) {
    echo $number;
}
```

7. Functions

Defining and Calling Functions:

- A function is a block of code designed to perform a particular task.
- Functions are defined using the function keyword.

Example:

php

Parameters and Return Values:

- Functions can accept parameters and return values.

Example:

php

8. Superglobals

Superglobals are built-in variables in PHP that are always accessible, regardless of scope.

Common Superglobals:

- *$_GET:* Used to collect form data sent via the GET method.

- *$_POST:* Used to collect form data sent via the POST method.
- *$_SERVER:* Contains information about headers, paths, and script locations.
- *$_SESSION:* Used to store information about a user's session.
- *$_COOKIE:* Used to store information in a user's browser.

Example:

php

9. Handling Forms and User Input

HTML Forms:

- Forms are used to collect user input.
- The form data can be sent using the GET or POST method.

Example:

html

Name: Submit

Handling Form Data in PHP:

php

Validation and Sanitization:

- *Validation*: Ensuring the input meets certain criteria (e.g., correct format).
- *Sanitization*: Cleaning up the input to prevent security issues (e.g., removing HTML tags).

Example:

php

10. Working with Databases

Connecting to MySQL:

- Use mysqli or PDO to connect to MySQL databases.

Example with mysqli:

php

connect_error) { die("Connection failed: " . $conn->connect_error); } echo "Connected successfully"; ?>

CRUD Operations:

- *Create*: INSERT INTO
- *Read*: SELECT

- *Update*: UPDATE
- *Delete*: DELETE

Example:

php

query($sql) === TRUE) { echo "New record created successfully"; } else { echo "Error: " . $sql . "
" . $conn->error; } ?>

11. File Handling

Reading and Writing Files:

- Use fopen(), fwrite(), fread(), and fclose() to handle files.

Example:

php

Uploading Files:

php

Select image to upload: Choose File No file chosen Upload Image

12. Error Handling

Types of Errors:

- *Notice:* Minor error, script continues execution.
- *Warning:* More serious error, but script continues.
- *Fatal Error:* Script halts.

Handling Errors:

- Use try...catch blocks to handle exceptions.

Example:

php

getMessage(), "\n"; } ?>

13. Security Best Practices

SQL Injection Prevention:

- Use prepared statements with mysqli or PDO.

Example:

php

prepare("SELECT * FROM users WHERE email = ?"); $stmt->bind_param("s", $email); $stmt->execute(); ?>

XSS Protection:

- Sanitize user input using htmlspecialchars().

Example:

php

alert('XSS');"); ?>

CSRF Protection:

- Implement CSRF tokens in forms.

Example:

```
<form method="post" action="submit.php">
  <input type="hidden" name="csrf_token" value="<?php echo
$_SESSION['csrf_token']; ?>">
  <input type="submit">
</form>
``
```