- ✓ **Embedding PHP in HTML**
- ✓ **Adding dynamic content**
- ✓ **Accessing form variables**
- ✓ **Identifiers**
- ✓ **User declared variables**
- ✓ **Variable types**
- ✓ **Assigning values to variables**
- ✓ **Constants**
- ✓ **Variable scope**
- ✓ **Operators and precedence**
- ✓ **Expressions**
- ✓ **Variable functions**

1. Embedding PHP in HTML

PHP Tag Styles

XML Style
```
<?php echo '<p>Order processed.</p>'; ?>
```
Short style
```
<? echo '<p>Order processed.</p>'; ?>
```
SCRIPT style
```
<script language='php'> echo '<p>Order processed.</p>'; </script>
```

2. Adding dynamic content

The main reason for using a server-side scripting language is to be able to provide dynamic content to a site's users.This is an important application because content that changes according to a user's needs or over time will keep visitors coming back to a site.

PHP allows us to do this easily.

```
<?php
  echo '<p>Order processed at ';
  echo date('H:i, jS F');
  echo '</p>';
?>
```

3. Accessing form variables

Normally we use
$_GET — HTTP GET Predefined Variables
$_POST — HTTP POST Predefined Variables

Example:-
```
$tireqty // short style
$_POST['tireqty'] // medium style
$HTTP_POST_VARS['tireqty'] // long style
```

4. Identifiers

Identifiers are the names of variables. (The names of functions and classes are also identifiers)

In PHP, identifiers are case sensitive. $tireqty is not the same as $TireQty.Trying to use these interchangeably is a common programming error. Function names are an exception to this rule—their names can be used in any case.
A variable can have the same name as a function.This is confusing however, and should be avoided. Also, you cannot create a function with the same name as another function.

5. **User declared variables**

You can declare and use your own variables in addition to the variables you are passed from the HTML form.

One of the features of PHP is that it does not require you to declare variables before using them. A variable will be created when you first assign a value to it.

6. **Variable types**

A variable's type refers to the kind of data that is stored in it.

PHP's Data Types
PHP supports the following data types:

Integer—Used for whole numbers
Double—Used for real numbers
String—Used for strings of characters
Boolean—Used for true or false values
Array—Used to store multiple data items of the same type
Object—Used for storing instances of classes

7. **Assigning values to variables**

You assign values to variables using the assignment operator, =

Example:-

```
$name="Rohit";
```

8. **Constants**

we can change the value stored in a variable.We can also declare constants. A constant stores a value such as a variable, but its value is set once and then cannot be changed elsewhere in the script.

Example:-

```
define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);
echo TIREPRICE;
```

9. **Variable scope**

The term scope refers to the places within a script where a particular variable is visible.

The four types of scope in PHP are as follows:
Built-in superglobal variables are visible everywhere within a script.
Global variables declared in a script are visible throughout that script, but not inside functions.
Variables used inside functions are local to the function.
Variables used inside functions that are declared as global refer to the global variable of the same

name.

The arrays $_GET and $_POST and some other special variables have their own scope rules.These are known as superglobals and can be seen everywhere, both inside and outside functions

10. Operators and precedence

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

Pre- and Post-Increment and Decrement

The pre- and post- increment (++) and decrement (--) operators are similar to the += and -= operators, but with a couple of twists.

```
$a=4;
echo ++$a;
```

The second line uses the pre-increment operator, so called because the ++ appears before the $a.This has the effect of first, incrementing $a by 1, and second, returning the incremented value. In this case, **$a is incremented to 5 and then the value 5 is returned** and printed.The value of this whole expression is 5. (Notice that the actual value stored in $a is changed:We are not just returning **$a + 1**.)

However, if the ++ is after the $a, we are using the post-increment operator.This has a different effect. Consider the following:

```
$a=4;
echo $a++;
```

In this case, the effects are reversed.That is, first, the value of $a is returned and printed, and second, it is incremented.The value of this whole expression is 4.This is the value that will be

printed. However, the value of $a after this statement is executed is 5.

As you can probably guess, the behavior is similar for the -- operator. However, the value of $a is decremented instead of being incremented.

References
$a = 5;
$b = $a;

These lines of code make a second copy of the value in $a and store it in $b. If we sub-sequently change the value of $a, $b will not change:

$a = 7; // $b will still be 5

You can avoid making a copy by using the reference operator, &. For example,

$a = 5;
$b = &$a;
$a = 7; // $a and $b are now both 7

The Ternary Operator

This operator, ?:, works the same way as it does in C. It takes the form condition ? value if true : value if false

The ternary operator is similar to the expression version of an
is covered later in this chapter.

A simple example is

($grade > 50 ? 'Passed' : 'Failed');

if-else
statement, which
This expression evaluates student grades to
'Passed'
or
'Failed'.
The Error Suppression Operator

The error suppression operator, @, can be used in front of any expression, that is, anything that generates or has a value. For example,
$a = @(57/0);

11. Precedence and Associativity: Evaluating Expressions

The precedence of an operator specifies how "tightly" it binds two expressions together. For example, in the expression 1 + 5 * 3, the answer is 16 and not 18 because the multiplication ("*") operator has a higher precedence than the addition ("+") operator. Parentheses may be used to force precedence, if necessary. For instance: (1 + 5) * 3 evaluates to 18.

When operators have equal precedence, their associativity decides whether they are evaluated starting from the right, or starting from the left - see the examples below.

The following table lists the operators in order of precedence, with the highest-precedence ones at the top. Operators on the same line have equal precedence, in which case associativity decides the order of evaluation.

Operator Precedence

Associativity	Operators	Additional Information
non-associative	clone new	clone and new
left	[	array()
non-associative	++ --	increment/decrement
right	~ - (int) (float) (string) (array) (object) (bool) @	types
non-associative	instanceof	types
right	!	logical
left	* / %	arithmetic
left	+ - .	arithmetic and string
left	<< >>	bitwise
non-associative	< <= > >= <>	comparison
non-associative	== != === !==	comparison
left	&	bitwise and references
left	^	bitwise
left	|	bitwise
left	&&	logical
left	||	logical
left	? :	ternary
right	= += -= *= /= .= %= &= |= ^= <<= >>= =>	assignment
left	and	logical
left	xor	logical
left	or	logical
left	,	many uses

12. Variable functions

Variable handling Functions

Table of Contents