PHP BASICS

Basic Language Features

This project is funded by
the European Union

Maharah
Code Your Future

LibyanSpider

Basic Language Features

hello.php

```php
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Basics</title>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
   Hello World!
   <?php
   echo "From PHP.";

   $a = 10; $b = 20;
   $c = $a    +   $b;
   ?>
   <br>
   Sum = <?=$c?>
   <br>
   <?php
   if ($a > $b) {
       echo "a is greater than b";
   }  else {
       echo "b is greater than a";
   }
   ?>
</body>
</html>
```

Basic Language Features

- The PHP file extension must be **.php**
- All statements in PHP must be terminated with a semicolon **;** An exception to this is if the statement happens to be the last statement before a closing tag.
- Whitespace has no semantic meaning in PHP. There is no need to line code up, but most coding standards enforce this to improve code readability.
- Multiple statements are allowed on a single line.
- Code blocks are denoted by the brace symbols { }
- PHP language construct and function names are not case-sensitive, but variable and constant names are.

Inserting PHP into Web Pages

Although PHP is a general scripting language and can be used for other purposes, it is most commonly deployed as a server-side scripting language used for building web pages.

The PHP parser does not parse anything that is not included in the tags that indicate PHP script. Content outside of PHP tags is simply output without inspection. This allows PHP to be embedded in HTML.

Type	Open	Close	Notes
Standard	<?php	?>	
Echo	<?=	?>	
Short	<?	?>	Deprecated

Language Constructs

- Language constructs are different from functions in that they are baked right into the language.

- Language constructs can be understood directly by the parser and do not need to be broken down. Functions, on the other hand, are mapped and simplified to a set of language constructs before they are parsed.

- Language constructs are not functions.They follow rules that are different from functions when it comes to parameters and the use of parentheses.

Language Constructs (2)

Construct	Used For
echo	Outputting a value to stdout.
print	Outputting a value to stdout.
print_r	Displays information about a variable in a way that's readable by humans, especially for arrays.
exit	Optionally outputting a message and terminating the program.
die	This is an alias for exit.
return	Terminates a function and returns control to the calling scope, or if called in the global scope, terminates the program.
empty	Returns a Boolean value depending on whether the variable is empty or not. Empty variables include null variables, empty strings, arrays with no elements, numeric values of 0, a string value of 0, and Boolean values of false.
isset	Check the variable is set

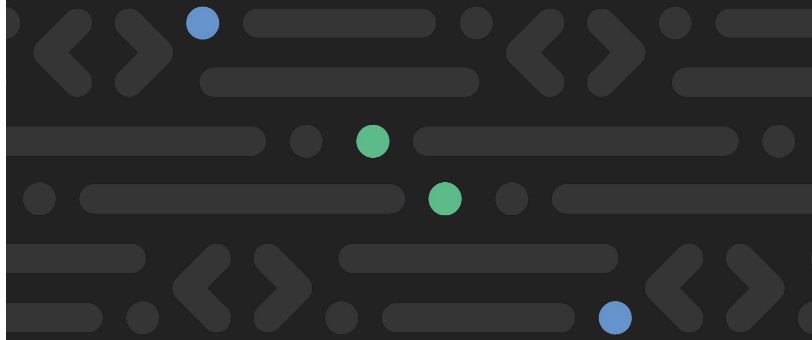
Language Constructs (3)

Construct	Used For
include	Includes a file and evaluates it. PHP will issue a warning if the file is not found or cannot be read.
include_once	If you specify include_once then PHP will make sure that it includes the file only once.
require	PHP will include a file and evaluate it. If the file is not found or cannot be read, then a fatal error will be generated and execution will be stopped.
require_once	As for include_once, but a fatal error will be generated instead of a warning and execution will be stopped..

Comments

There are three styles to mark comments:

- **#** Perl style comments
- **//** C style comments
- **/***
  Multiline comment
  ***/**

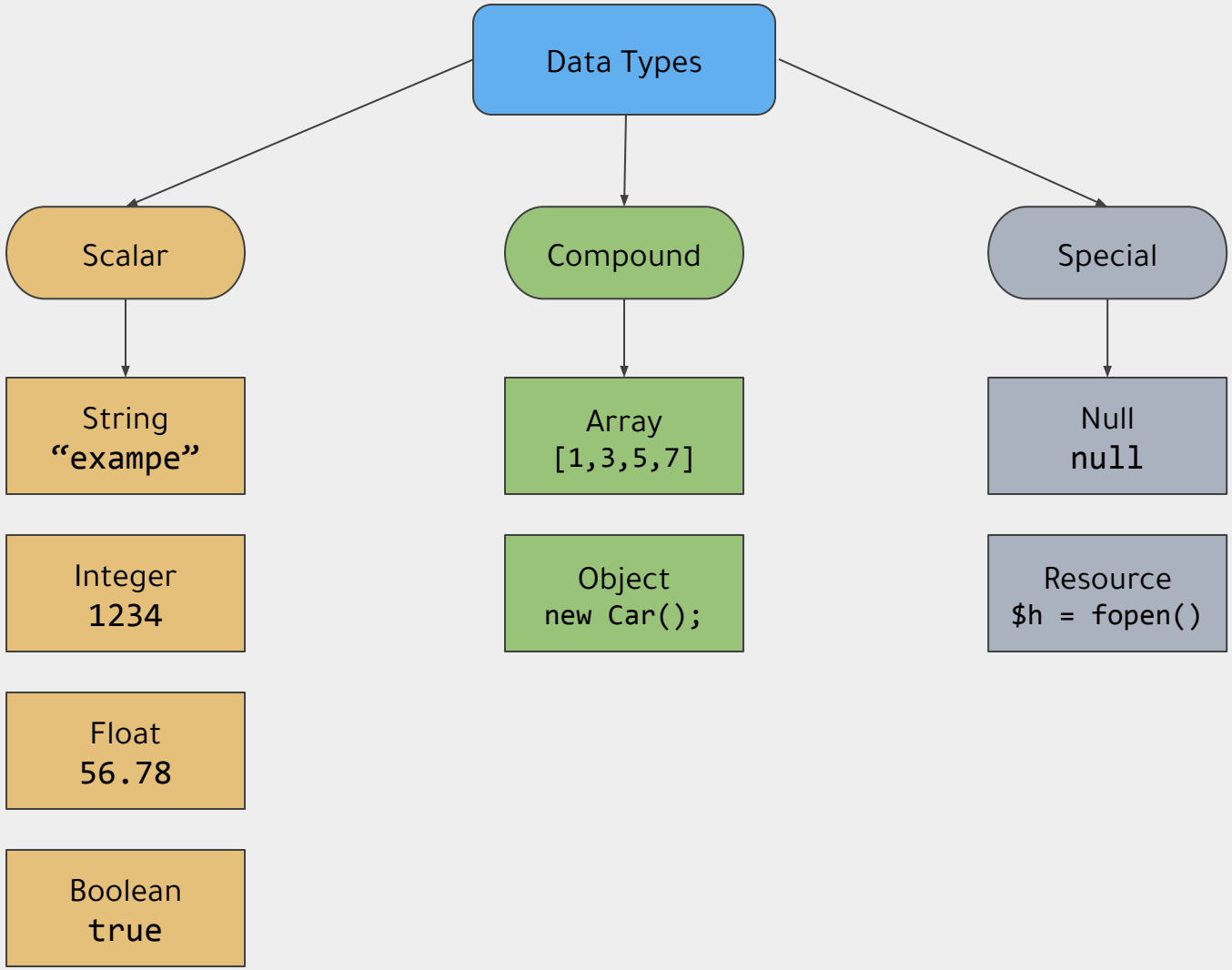


Representing Numbers

There are four ways in which an integer may be expressed in a PHP script:

Notation	Example
Decimal	1234
Binary	0b10011010010
Octal	02322
Hexadecimal	0x4D2

Data Types


Data Types
Scalar
Compound
Special
String
“exampe”
Integer
1234
Float
56.78
Boolean
true
Array
[1,3,5,7]
Object
new Car();
Null
null
Resource
$h = fopen()

Variables

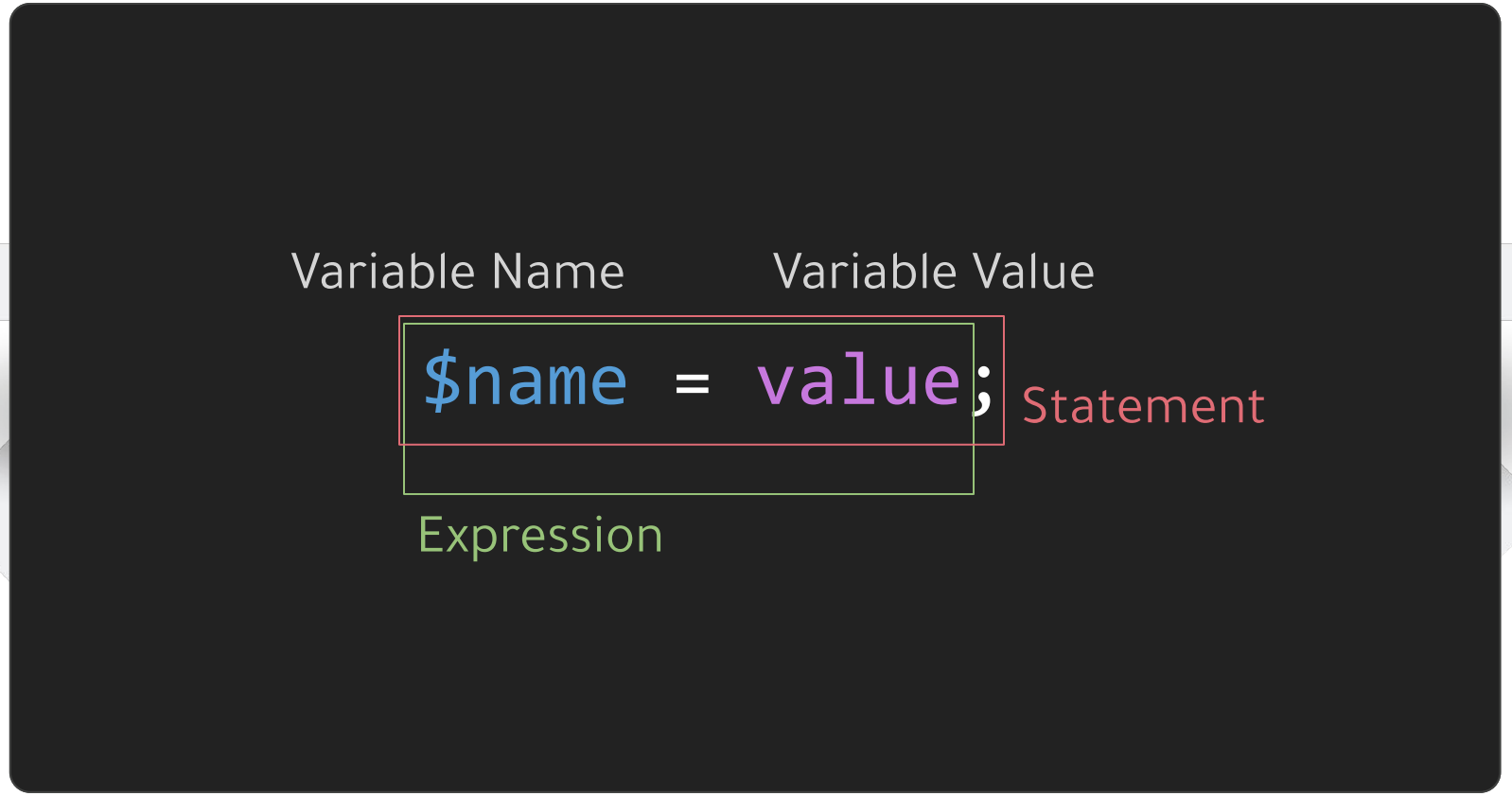
Variables in PHP are used to store and manipulate data. PHP supports various data types, including **strings**, **numbers**, **booleans**, **arrays**, and **objects**. Variables are declared using the **$** sign followed by the variable name.

To declare and assign a value to a variable, use the assignment operator **=** PHP automatically determines the data type based on the assigned value. Variables can be updated and re-assigned throughout the script.

PHP is a loosely typed language. It is important not to think that PHP variables don't have a type. it's just that they may change type during runtime and don't need their type to be declared explicitly when initialized.

```php
<?php
  $name = "Maharah";
  $age = 20;
  $isStudent = true;
  $grades = [80, 85, 90];
  $user = new User();
```

variable


Variable Name
Variable Value
$name = value;
Statement
Expression

Operators and Expressions

PHP offers a wide range of operators for performing arithmetic, comparison, logical, and assignment operations. Expressions combine variables, literals, and operators to perform calculations or generate new values.

```php
<?php
 $num1 = 10;
 $num2 = 5;

 $sum = $num1 + $num2;
 $difference = $num1 - $num2;
 $product = $num1 * $num2;
 $quotient = $num1 / $num2;
```

Naming Variables

PHP variables begin with the dollar symbol **$** and PHP variable names adhere to the following rules:

- Names are case sensitive.
- Names may contain letters, numbers, and the underscore character.
- Names may not begin with a number.

Coding conventions differ on the use of **camelCase**, **StudlyCase**, or **snake_case**, but all of these formats are valid PHP variable name formats.

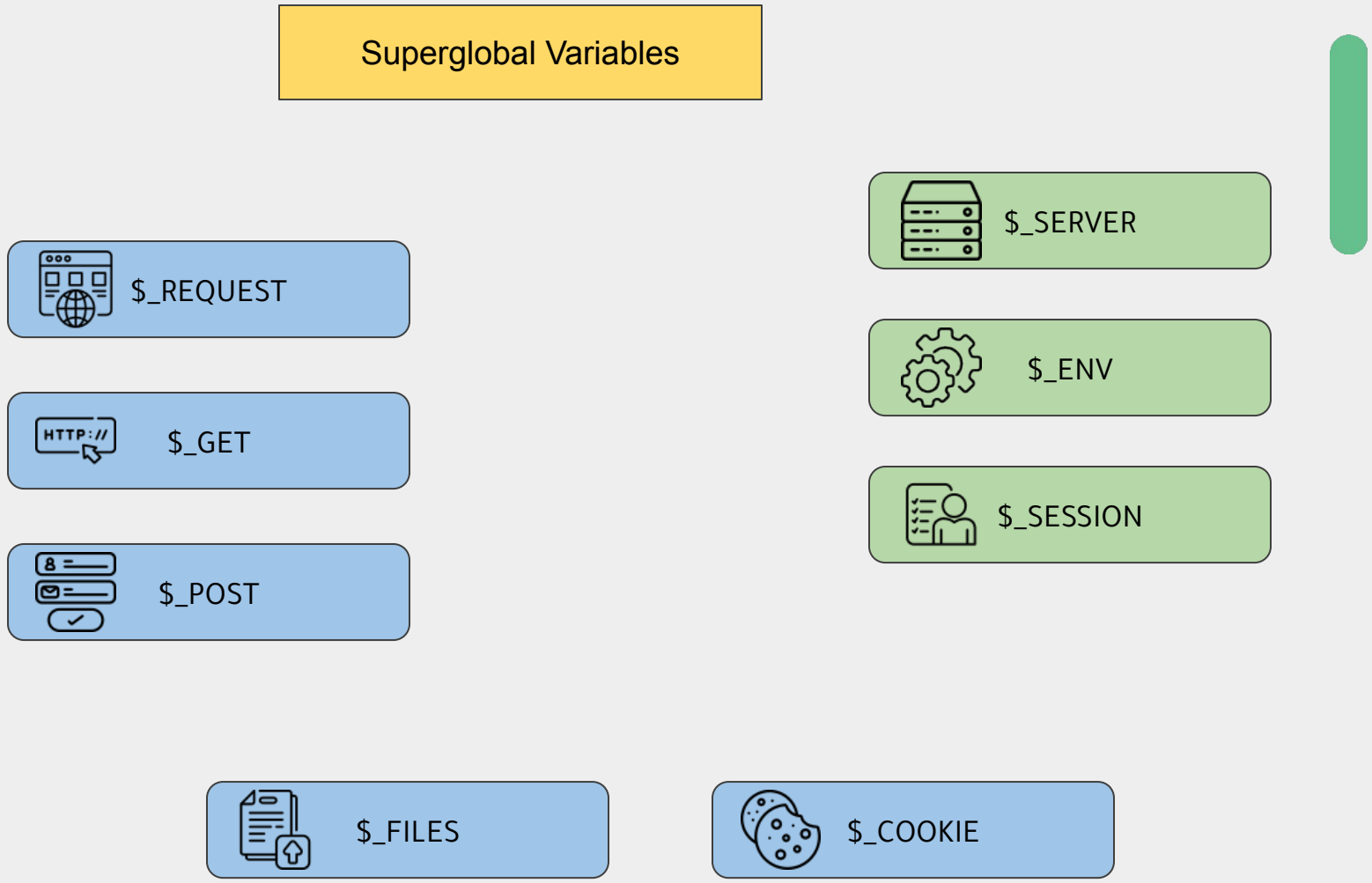
PHP also allows for variable variable names. This is best illustrated by example:

```php
<?php
$a = 'foo';
$$a = 'bar'; // $a is 'foo', so variable $foo is set
echo $foo;   // bar
```

Checking If a Variable Has Been Set?

- The command **isset()** will return true if a variable has been set and false otherwise. It is preferable to use this function instead of checking if the variable is null because it won't cause PHP to generate a warning.
- The command **empty()** will return true if a variable is not set and will not generate a warning. This is not a bulletproof way to test if a variable is set.
- Remember that the string "**0**" is considered empty, but is actually set.
- Variables become unset when they become out of scope and you can use the command **unset()** to manually clear a variable.
- **is_null()** is used to check if a variable has a null

Superglobal Variables



Superglobal Variables

Superglobal variables are predefined variables that are accessible from any part of the PHP script. They provide information about the server, user input, and other useful data. Examples of superglobal variables.

Super Global	Stores
$GLOBALS	References all variables defined in the global scope.
$_SERVER	Server and execution environment information.
$_GET	HTTP GET variables.
$_POST	HTTP POST variables.
$_FILES	Provides information about the uploaded file.
$_SESSION	Session variables.
$_COOKIE	HTTP Cookies.
$_REQUEST	POST, GET, and COOKIE request variables.

Constants

Constants are similar to variables but are immutable. They have the same naming rules as variables, but by convention will have uppercase names.

They can be defined using the **define** function as shown:

- You cannot assign a variable to a constant.
- You can use static scalar values to define a constant, like this:

```
const NAME = "value";

define("NAME", "value");
```

```
<?php
define('NAME', 'Maharah');
define('AGE', 20);
echo NAME . ' ' . AGE;
```

Magic Constants

Magic constants are those which PHP provides automatically to every running script. There are quite a lot of reserved constants and you will need to know the error constants, as well as the commonly used predefined constants.

Construct	Used For
__LINE__	The current line number of the PHP script being executed
__FILE__	The fully resolved (including symlinks) name and path of the file being executed
__CLASS__	The name of the class being executed
__FUNCTION__	The name of the function being executed

Note: the value of these magic constants changes depending on where you use it.

Checking If Constant Has Been Set?

You can check if a constant has been set using the **defined**() function.

```php
<?php
// Define a constant
define('MY_CONSTANT', 'some value');
echo defined('MY_CONSTANT'); // outputs 1
```

Remember, constants in PHP are case-sensitive by default, so ensure that you use the correct case when checking for their existence.

Operators

- **Arithmetic**: Addition, Subtraction, Division, Multiplication, Modulus, Power
  +, -, /, * , %
- **Logic Operators:** PHP uses both symbol and word form logic operators. The symbol form operators are
  and, or, &&, ||, xor
- **Ternary Operator (Short)**: PHP implements the ternary operator in the same format as other C-ancestor languages. The general format is as follows:
  condition ? expression1 : expression2;
- **Null Coalescing Operator**: The null coalescing operator is just a special case of the ternary operator. It allows you to neaten up the syntax used when you're using isset to assign a default value to a variable.
  condition ?? expression;
- **Assignment Operators**: Assignment Operators: PHP uses the = symbol as an assignment operator. The assignment operator can be combined with just about all the binary and arithmetic operators.
- **Comparison Operators:** >, >=, <. <=, <>, ==, ===, !=, !==
- **Reference Operator**: By default, PHP assigns all scalar variables by value. PHP has optimizations to make assignment by value faster than assigning by reference, but if you want to assign by reference, you can use the & operator.

Control Structures

PHP supports if, else, elseif, switch, and ternary conditional structures. If structures look like this:

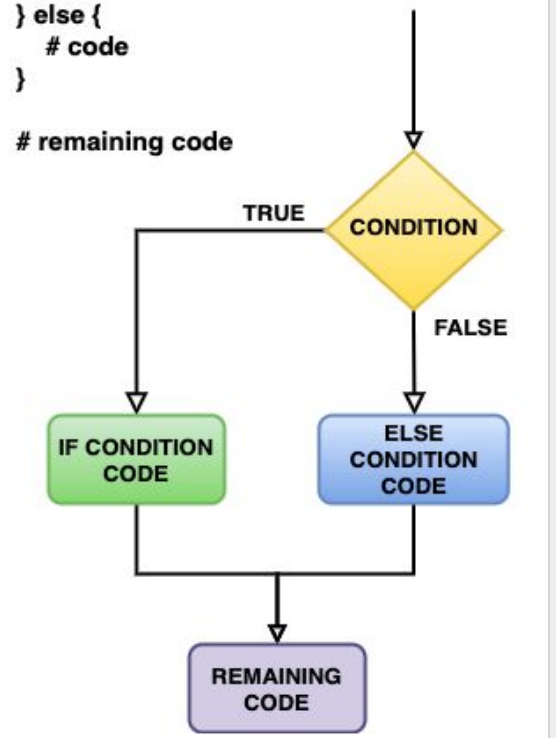
- Conditional statements in PHP allow us to make decisions in our code based on different conditions.
- They enable us to control the flow of our program, executing different code blocks depending on whether specific conditions are met.

- **if-else Statement**
- **switch Statement**
- **match Statement**

if-else Statement

```
if (condition) {
    # code
} else {
    # code
}

# remaining code
```



if-else Statement

The if-else statement is used to execute code blocks conditionally based on a specified condition. If the condition evaluates to true, the code within the if block is executed. Otherwise, the code within the else block (if present) is executed.

```php
<?php
$grade = 75;
if ($grade >= 90) {
   echo "Grade: A";
} elseif ($grade >= 80) {
   echo "Grade: B";
} elseif ($grade >= 70) {
   echo "Grade: C";
} elseif ($grade >= 60) {
   echo "Grade: D";
} else {
   echo "Grade: F";
}
```

switch Statement

The switch statement is used to perform different actions based on different conditions. It simplifies the code when dealing with multiple possible conditions. Each condition is checked against a value, and the corresponding code block is executed.

```php
<?php
$fruit = "apple";
switch ($fruit) {
    case "apple":
        echo "You selected an apple!";
        break;
    case "banana":
        echo "You chose a banana!";
        break;
    case "orange":
        echo "You picked an orange!";
        break;
    default:
        echo "Invalid fruit selection";
}
```

■ Note if you include case statements after the **default** case, they will not be checked.

match Statement

Match expressions are available as of PHP 8.0.0.

The match expression branches evaluation based on an identity check of a value.

Similarly to a switch statement, a match expression has a subject expression that is compared against multiple alternatives. Unlike switch, it will evaluate to a value much like ternary expressions. Unlike switch, the comparison is an identity check (===) rather than a weak equality check (==).

```php
<?php
$food = 'cake';

$return_value = match ($food) {
    'apple' => 'This food is an apple',
    'bar' => 'This food is a bar',
    'cake', 'cookie' => 'This food is a cake',
    default => 'I don\'t know',
};

echo $return_value;
```

Question?

01

What will this script output?

```php
<?php
$number = 1;
$return_value = match ($number) {
  '1' => 'The number is 1',
  '2' => 'The number is 2',
  '3' => 'The number is 3',
  default => 'I don\'t know',
};

echo $return_value;
```